Statistical Learning and Linear Regression

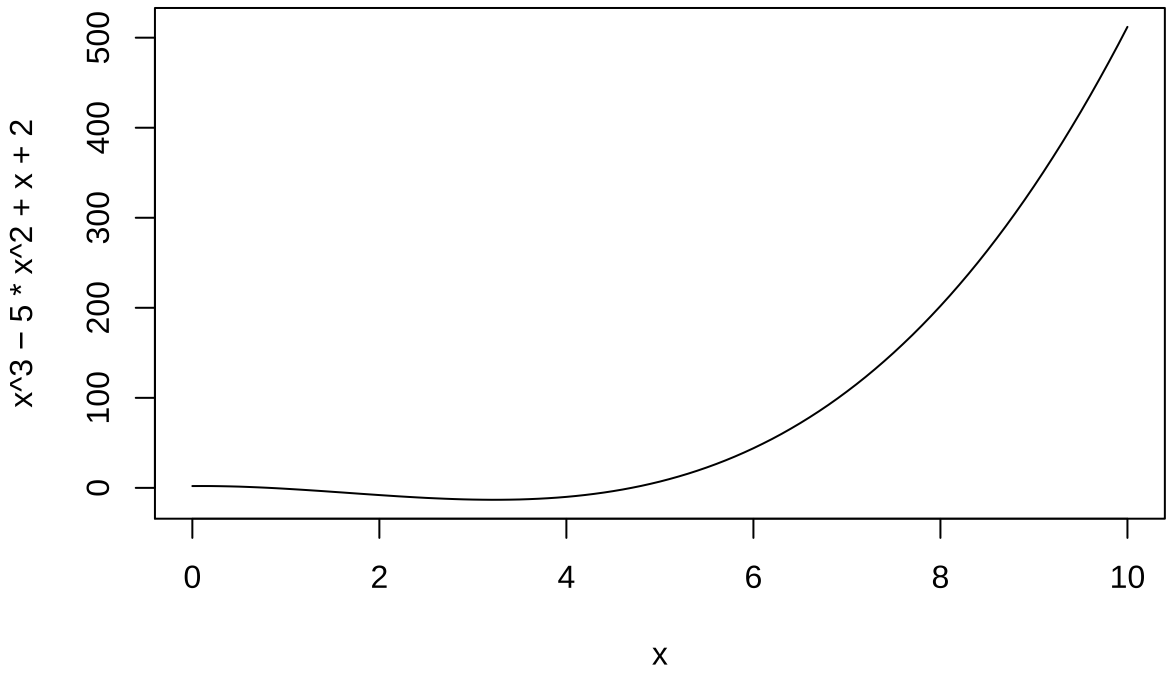
Naeem Chowdhury

2/14/2020

1. Reproduce slide 18, using R markdown.

- Let $f(x) = x^3 - 5x^2 + x + 2$. That's the truth. Draw $f(x)$ in range $x \in (0, 10)$.

```
curve(x^3 - 5*x^2 + x + 2, from = 0, to = 10)
```



- Make 10 training sets. To make each training set, pick 15 random values of x in the range. Generate 15 responses $f(x) + \epsilon$, with $\epsilon \sim \mathbb{N}(0, 2)$.

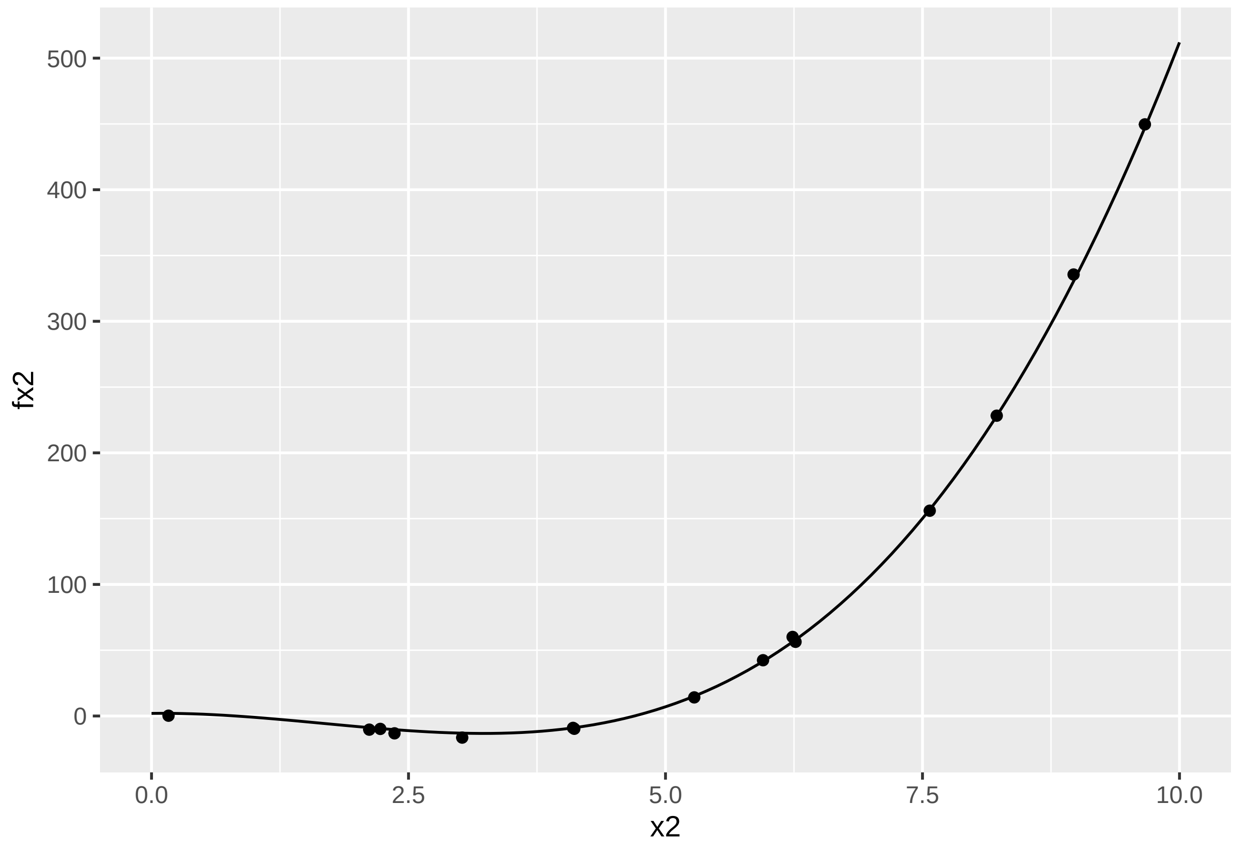
Produce training data, and resulting function values.

```
vecG <- c(1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1)
trainingData <- data.frame(id = vecG, stringsAsFactors=TRUE)
# Creating each traing seet
for (i in (1:10)){
  vecFx <- vector() # vector of random variables
  vecX <- vector() # vector of randomly generated x
  randEpsilon <- rnorm(15, 0, 2)
  randX <- runif(15, 0, 10)
  for(j in (1:15)){


    fx <- ((randX[j])^3) - (5*((randX[j])^2)) + randX[j] + 2 + randEpsilon[j]

    vecX[j] <- randX[j]
    vecFx[j] <- fx
  }
  trainingData <- cbind(trainingData, data.frame(name = vecX))
  trainingData <- cbind(trainingData, data.frame(name2 = vecFx))
}
colnames(trainingData) <- c("dummy", "x1", "fx1", "x2", "fx2", "x3", "fx3", "x4", "fx4", "x5", "fx5", "
trainingData <- trainingData[,-1]

fun.1 <- function(x) x^3 - 5*x^2 + x + 2
```

```
## Creating Testing Sets
vecG <- c(1,1,1,1,1,1,1,1,1,1,1,1,1,1,1)
testingData <- data.frame(id = vecG, stringsAsFactors=TRUE)
# Creating each testing set
for (i in (1:10)){

  vecFx <- vector() # vector of random variables
  vecX <- vector() # vector of randomly generated x
  randEpsilon <- rnorm(15,0,2)
  randX <- runif(15,0,10)
  for(j in (1:15)){

    fx <- ((randX[j])^3) - (5*((randX[j])^2)) + randX[j] + 2 + randEpsilon[j]

    vecX[j] <- randX[j]
    vecFx[j] <- fx
  }
  testingData <- cbind(testingData, data.frame(name = vecX))
  testingData <- cbind(testingData, data.frame(name2 = vecFx))
}
colnames(testingData) <- c("dummy", "x1", "fx1", "x2", "fx2", "x3", "fx3", "x4", "fx4", "x5", "fx5", "x
testingData <- testingData[,-1]
```

Showing a sample plot

```
fun.1 <- function(x) x^3 - 5*x^2 + x + 2

ggplot(data = trainingData, mapping = aes(x = x2, y = fx2)) +
  geom_point() +
  stat_function(fun = fun.1) + xlim(0,10)
```



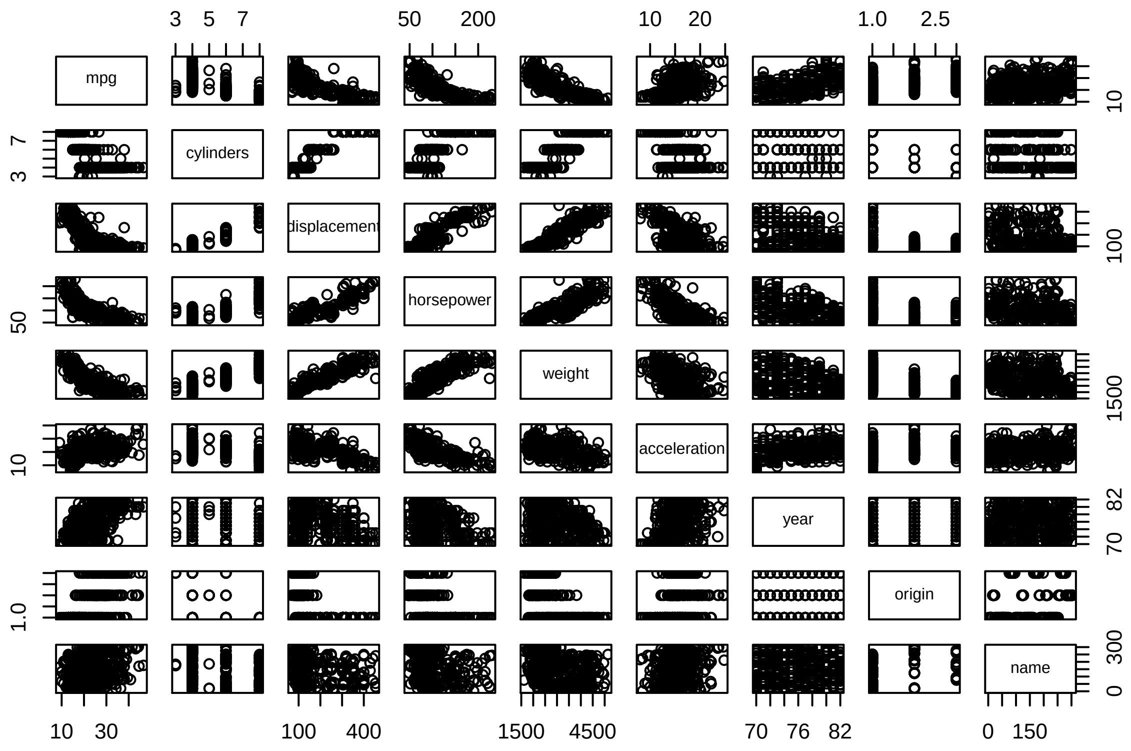
```r
## Creating Models and Calculating MSE for training and test
dummyVec <- c(1,2,3,4,5)
dummyMatrix <- matrix(0, ncol = 10, nrow = 5)
testMSE.df <- data.frame(dummyMatrix)
## Polynomial 1
polyMSEs <- data.frame(id = dummyVec, stringsAsFactors=TRUE)

for (i in (1:10)){ # For each training set


  xName <- paste("x",i, sep="")
  fxName <- paste("fx",i, sep="")
  trainingMSEs <- c()

  for (k in (1:5)){
    lm.obj <- lm(trainingData[,(i*2)] ~ poly(trainingData[,i*2-1],k))


    traingMSE <- mean((trainingData[,(i*2)] - lm.obj$fitted.values)^2)
    trainingMSEs <- append(trainingMSEs, traingMSE)

    testMSEs <- c()


    for (j in (1:10)){
      prediction <- as.vector( predict(lm.obj, as.data.frame( testingData[, (2*j-1)]  ) ) )
      testMSE <- mean(( testingData[,(2*j)] - prediction)^2)
      testMSEs <- append(testMSEs, testMSE)

    }

    avgtestMSE <- mean(testMSEs)

    testMSE.df[k,i] <- avgtestMSE
  }

  polyMSEs <- cbind(polyMSEs, trainingMSEs)

}

colnames(polyMSEs) <- c('degree','trainMSE1','trainMSE2','trainMSE3','trainMSE4','trainMSE5','trainMSE6

polyMSEs <- polyMSEs %>% mutate(avgMSE = Reduce("+",.)/length(.))


colnames(testMSE.df) <- c('testMSE1','testMSE2','testMSE3','testMSE4','testMSE5','testMSE6','testMSE7',

testMSE.df <- testMSE.df %>% mutate(avg = Reduce("+",.)/length(.))
```

Problem 3

```r
library(ISLR)
View(Auto)
```

a. Produce a scatterplot matrix which includes all of the variables in the data set

```r
pairs(Auto)
```



b. Compute the matrix of correlations between the variables using the function cor(). You will need to exclude the "name" variable, which is qualitative.

```
cor(Auto[,names(Auto) !="name"])
```

```
##                     mpg  cylinders displacement horsepower     weight
## mpg           1.0000000 -0.7776175   -0.8051269 -0.7784268 -0.8322442
## cylinders    -0.7776175  1.0000000    0.9508233  0.8429834  0.8975273
## displacement -0.8051269  0.9508233    1.0000000  0.8972570  0.9329944
## horsepower   -0.7784268  0.8429834    0.8972570  1.0000000  0.8645377
## weight       -0.8322442  0.8975273    0.9329944  0.8645377  1.0000000
## acceleration  0.4233285 -0.5046834   -0.5438005 -0.6891955 -0.4168392
## year          0.5805410 -0.3456474   -0.3698552 -0.4163615 -0.3091199
## origin        0.5652088 -0.5689316   -0.6145351 -0.4551715 -0.5850054
##              acceleration       year     origin
## mpg             0.4233285  0.5805410  0.5652088
## cylinders      -0.5046834 -0.3456474 -0.5689316
## displacement   -0.5438005 -0.3698552 -0.6145351
## horsepower     -0.6891955 -0.4163615 -0.4551715
## weight         -0.4168392 -0.3091199 -0.5850054
## acceleration    1.0000000  0.2903161  0.2127458
## year            0.2903161  1.0000000  0.1815277
## origin          0.2127458  0.1815277  1.0000000
```

c. Perform a multiple linear regression with mpg as the response and all other variables except name as the predictors. Use the summary() function to print the results.

```r
regression.model = lm(mpg ~. -name, data = Auto)
summary(regression.model)
```

```
##
## Call:
## lm(formula = mpg ~ . - name, data = Auto)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.5903 -2.1565 -0.1169  1.8690 13.0604
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -17.218435   4.644294  -3.707  0.00024 ***
## cylinders     -0.493376   0.323282  -1.526  0.12780
## displacement   0.019896   0.007515   2.647  0.00844 **
## horsepower    -0.016951   0.013787  -1.230  0.21963
## weight        -0.006474   0.000652  -9.929  < 2e-16 ***
## acceleration   0.080576   0.098845   0.815  0.41548
## year           0.750773   0.050973  14.729  < 2e-16 ***
## origin         1.426141   0.278136   5.127 4.67e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.328 on 384 degrees of freedom
## Multiple R-squared:  0.8215, Adjusted R-squared:  0.8182
## F-statistic: 252.4 on 7 and 384 DF,  p-value: < 2.2e-16
```

-Is there a relationship between the predictors and the response?

The predictors have a nonzero response with the relationship, but some of the relationships are not statistically significant. The adjusted R^2 value of 0.82 tell us that about 82% of the variances in *mpg* are explained by the predictors.

-Which predictors appear to have a statistically significant relationship to the response?

We can say that the predictors with a p-value of less than 0.05 (under significance level of 95%) have a statistically significant relationship to the response. (displacement, origin, year, weight)

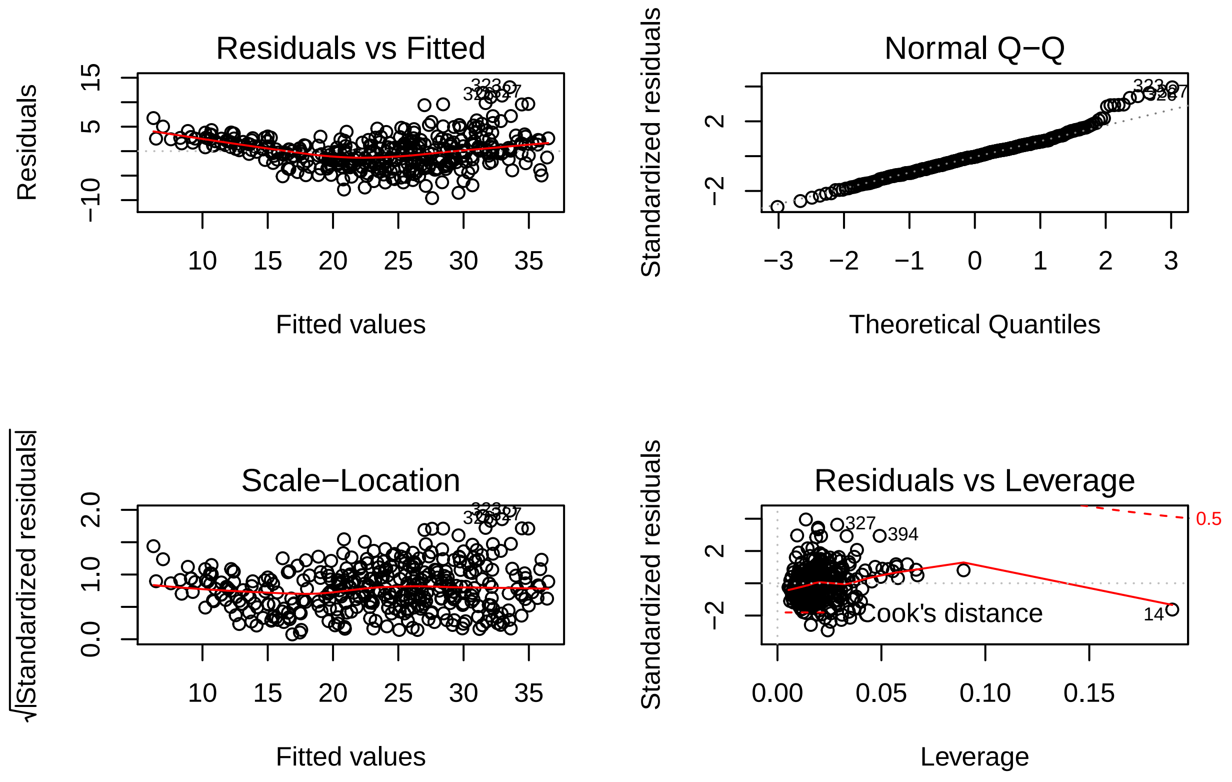
-What does the coefficient for the year variable suggest?

A coefficient of 0.75 suggests that the average effect of the passing of a year is 0.75 increase in *mpg* when all other predictors are fixed.

d. Use the plot() function to produce diagnostic plots of the linear regression fit. Comment on any problems you see with the fit. Do the residual plots suggest any unusually large outliers? Does the leverage plots identify any observations with unusually high leverages?

In the first graph, we can see that the relationship between the response and the predictors is non-linear. The residuals seem normally distributed and right-skewed. The leverage graph shows no high leverage points except for point 14.

```r
par(mfrow=c(2,2))
plot(regression.model)
```



e. Use the * and : symbols to fit linear regression models with interaction effects.

```r
regression.model = lm(mpg ~.-name-cylinders-acceleration+year:origin+displacement:weight+
                  displacement:weight+acceleration:horsepower+acceleration:weight, data=Auto)
summary(regression.model)
```

```
##
## Call:
## lm(formula = mpg ~ . - name - cylinders - acceleration + year:origin +
##     displacement:weight + displacement:weight + acceleration:horsepower +
##     acceleration:weight, data = Auto)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.5074 -1.6324  0.0599  1.4577 12.7376
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)             1.868e+01  7.796e+00   2.396 0.017051 *
## displacement           -7.794e-02  9.026e-03  -8.636  < 2e-16 ***
## horsepower              8.719e-02  3.167e-02   2.753 0.006183 **
## weight                 -1.350e-02  1.287e-03 -10.490  < 2e-16 ***
## year                    4.911e-01  9.825e-02   4.998 8.83e-07 ***
## origin                 -1.262e+01  4.109e+00  -3.071 0.002288 **
## year:origin             1.686e-01  5.277e-02   3.195 0.001516 **
## displacement:weight     2.253e-05  2.184e-06  10.312  < 2e-16 ***
## horsepower:acceleration -9.164e-03  2.222e-03  -4.125 4.56e-05 ***
## weight:acceleration      2.784e-04  7.087e-05   3.929 0.000101 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.861 on 382 degrees of freedom
## Multiple R-squared:  0.8687, Adjusted R-squared:  0.8656
## F-statistic: 280.8 on 9 and 382 DF,  p-value: < 2.2e-16
```

displacement, origin, weight, and year are used due to their statistical significance. The summary confirms.

f. Try a few different transformations of the variables. Comment on the results.

```r
# Model with horsepower interacting with weight.
lm.fit = lm(mpg ~ origin+weight+year+displacement+(horsepower*weight), data=Auto)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = mpg ~ origin + weight + year + displacement + (horsepower *
##     weight), data = Auto)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.4091 -1.7734 -0.1386  1.5039 11.9502
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)        7.012e-01  3.966e+00   0.177 0.859757
## origin             8.399e-01  2.488e-01   3.376 0.000809 ***
## weight            -1.142e-02  6.891e-04 -16.571  < 2e-16 ***
## year               7.727e-01  4.476e-02  17.263  < 2e-16 ***
## displacement       6.504e-03  4.884e-03   1.331 0.183822
## horsepower        -2.202e-01  2.076e-02 -10.605  < 2e-16 ***
## weight:horsepower  5.438e-05  5.068e-06  10.731  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.928 on 385 degrees of freedom
## Multiple R-squared:  0.8615, Adjusted R-squared:  0.8593
## F-statistic:   399 on 6 and 385 DF,  p-value: < 2.2e-16
```

All of the coefficients are statistically significant except for displacement, with a higher p-value of 0.18. Horsepower is now significant when interacting with weight.

```r
# linear combination of logs
lm.fit = lm(mpg ~ log(cylinders)+log(origin)+log(weight)+log(year)+log(displacement)+log(acceleration),
summary(lm.fit)
```

```
##
## Call:
## lm(formula = mpg ~ log(cylinders) + log(origin) + log(weight) +
##     log(year) + log(displacement) + log(acceleration), data = Auto)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.7712 -2.0217 -0.0756  1.6291 13.0172
##
```

```
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       -80.6285    17.6409  -4.571 6.56e-06 ***
## log(cylinders)      1.4934     1.6993   0.879   0.3801
## log(origin)         1.3100     0.5169   2.534   0.0117 *
## log(weight)       -18.6839     1.7994 -10.383  < 2e-16 ***
## log(year)          58.7365     3.5395  16.594  < 2e-16 ***
## log(displacement)  -1.0393     1.5760  -0.659   0.5100
## log(acceleration)   0.3333     1.1094   0.300   0.7640
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.143 on 385 degrees of freedom
## Multiple R-squared:  0.8403, Adjusted R-squared:  0.8378
## F-statistic: 337.6 on 6 and 385 DF,  p-value: < 2.2e-16
```

In this model, only origin, weight, and year are significant.