Problem Set3: Least Square Regression

Problem 1

Simple linear regression is to find b_0, b_1 such that they minimize the quadratic loss function(no regularization).

$$L(\beta_0, \beta_1) = \sum(y_i - \beta_0 - \beta_1 xi)^2$$

To find the global minimum, we take the partial derivatives with respect to each parameter of this multivariable($\mathbb{R}^2 \to \mathbb{R}$) function.

$$\frac{\partial L}{\partial \beta_0} = 2 * (-1) * \sum_{i=1}^{n}(y_i - \beta_0 - \beta_1 x_i) = 0 \frac{\partial L}{\partial \beta_1} = 2 * \sum_{i=1}^{n}(-x_i)(y_i - \beta_0 - \beta_1 x_i) = 0$$

so $\sum_{i=1}^{n}(y_i - \beta_0 - \beta_1 x_i) = \frac{0}{-2} = 0$, therefore $\sum_{i=1}^{n} e_i = 0$

Problem 2

a. Becuase a matrix's cross product with its transpose is a symmetric matrix,

$$X^T X = \begin{bmatrix} 30 & 0 & 0 \\ 0 & 10 & 7 \\ 0 & 7 & 15 \end{bmatrix}$$

n = 30(the inner product of the column with the constant term)
b.According to the property of matrix cross product,

$$X^T X = \begin{bmatrix} n & \sum X_i & \sum Z_i \\ 0 & \sum X_i^2 & \sum X_i Z_i \\ 0 & 7 & \sum Z_i^2 \end{bmatrix}$$

and $cor(x, z) = \frac{n \sum X_i Z_i - \sum X_i \sum Z_i}{\sqrt{n \sum X_i^2 - (\sum X_i)^2} \sqrt{n \sum Z_i^2 - (\sum Z_i)^2}}$

so

$$\begin{aligned} cor(x, z) &= \frac{30 * 7 - 0 * 0}{\sqrt{30 * 10 - 0^2} * \sqrt{30 * 15 - 0^2}} \\ &= \frac{210}{\sqrt{300} * \sqrt{450}} \\ &= 0.5715476 \end{aligned}$$

c.
$\bar{y} = -2 + \bar{x} + 2\bar{z} = -2 + \frac{1}{n}(\sum X_i + 2\sum Z_i) = -2 + 0 + 2 * 0 = -2$

d. Given, $R^2 = \frac{||\hat{y}||^2}{||y||^2} = \frac{\sum(\hat{y} - \bar{y})^2}{\sum(\hat{y} - \bar{y})^2 + RSS}$, To find R^2, we have to find $\sum(\hat{y} - \bar{y})^2$

$$
\begin{aligned}
\sum(\hat{y}-\bar{y})^2 &= \sum(-2+x_i+2z_i-(-2))^2 \\
&= \sum(x_i+2z_i)^2 \\
&= \sum(x_i^2+4z_i^2+4x_iz_i) \\
&= \sum x_i^2+4\sum z_i^2+4\sum x_iz_i \\
&= 10+4*15+4*7 \\
&= 98
\end{aligned}
$$

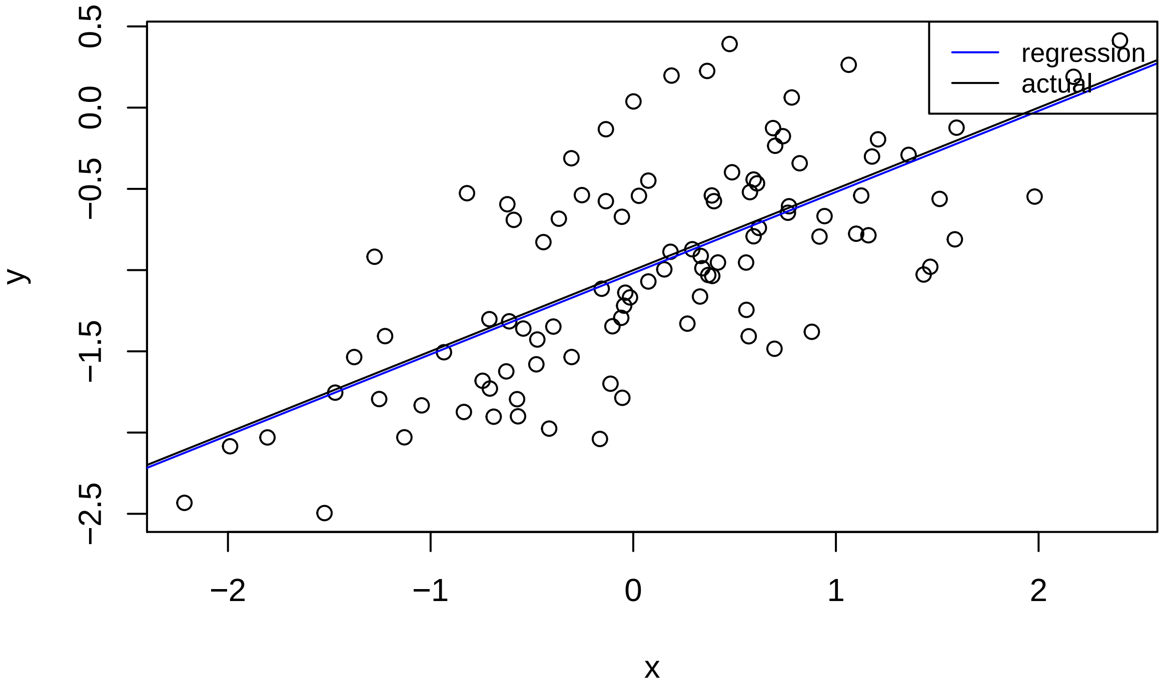
we have $\sum(\hat{y}-\bar{y})^2 = 46$ and $R^2 = \frac{\sum(\hat{y}-\bar{y})^2}{\sum(\hat{y}-\bar{y})^2+RSS} = \frac{98}{98+12} = 0.8909091$

Problem 3

```
set.seed(1)
#a
x <- rnorm(100)
#b
eps <- rnorm(100, mean = 0, sd = 0.5)
#c
y <- -1 + 0.5*x + eps
#d
plot(x, y)
#e
reg <- lm(y~x)
summary(reg)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.93842 -0.30688 -0.06975  0.26970  1.17309
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.01885    0.04849 -21.010  < 2e-16 ***
## x            0.49947    0.05386   9.273 4.58e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4814 on 98 degrees of freedom
## Multiple R-squared:  0.4674, Adjusted R-squared:  0.4619
## F-statistic: 85.99 on 1 and 98 DF,  p-value: 4.583e-15
```

```
#f
plot(x, y)
abline(reg, col = "blue")
abline(a=-1, b=0.5, col = "black")
legend("topright", c("regression", "actual"), col=c("blue", "black"), lty=1, cex=0.8)
```




```r
#g
reg2 <- lm(y ~ poly(x, 2, raw=T))
print("reg1 vs reg2, absolute error vs square error")
```

```
## [1] "reg1 vs reg2, absolute error vs square error"
```

```r
print(paste(sum(abs(reg$residuals)), sum(abs(reg2$residuals))))
```

```
## [1] "37.7109692989015 38.0552098547766"
```

```r
print(paste(sum(reg$residuals * reg$residuals), sum(reg2$residuals*reg2$residuals)))
```

```
## [1] "22.7089022692233 22.257276711411"
```

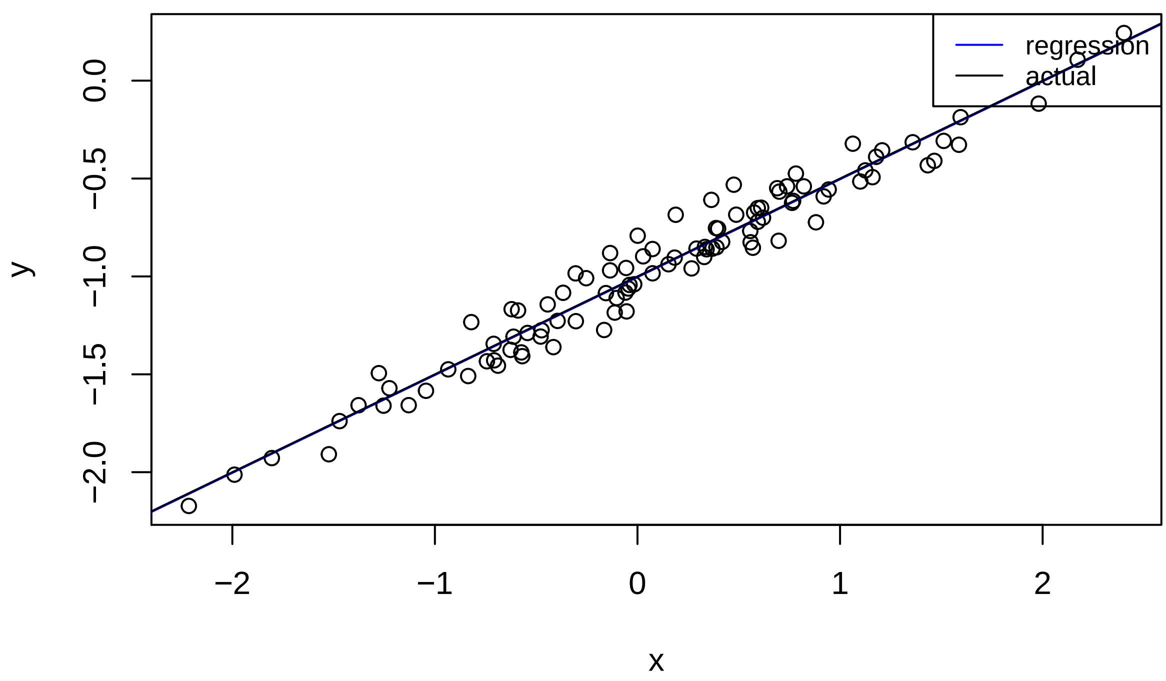
Comment: d: the plot looks decently homoscedastic e: the estimates are very close g: the 2nd degree polynomial has a better(smaller) MSE but higher sum of absolute errors. In summary, the performance of 1degree and 2degree is difficult to tell.

h

```r
set.seed(1)
#a
x <- rnorm(100)
#b
eps <- rnorm(100, mean = 0, sd = 0.1)
#c
y <- -1 + 0.5*x + eps
#d
plot(x, y)
#e
reg <- lm(y~x)
summary(reg)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.18768 -0.06138 -0.01395  0.05394  0.23462
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.003769   0.009699  -103.5   <2e-16 ***
## x            0.499894   0.010773    46.4   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.09628 on 98 degrees of freedom
## Multiple R-squared:  0.9565, Adjusted R-squared:  0.956
## F-statistic:  2153 on 1 and 98 DF,  p-value: < 2.2e-16
```

```r
#f
plot(x, y)
abline(reg, col = "blue")
abline(a=-1, b=0.5, col = "black")
legend("topright", c("regression", "actual"), col=c("blue", "black"), lty=1, cex=0.8)
```



Comment: g.The two lines almost overlap each other. When there is less noise, simple linear regression owns.

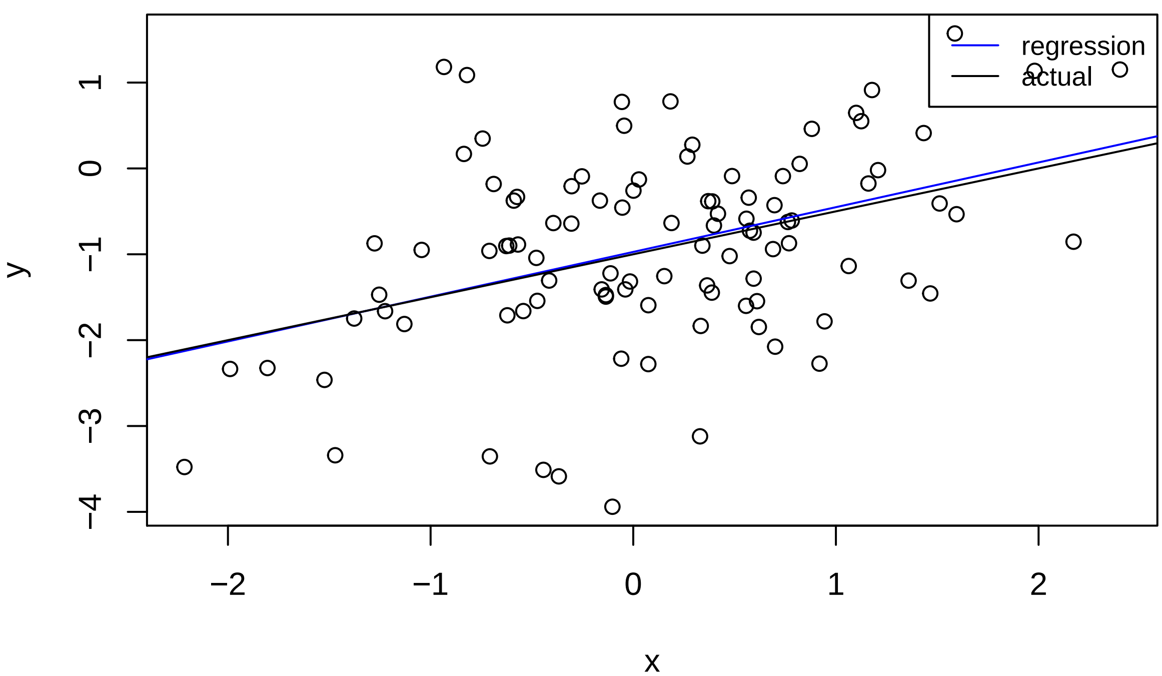
i

```r
#b
eps <- rnorm(100, mean = 0, sd = 1)
```

```r
#c
y <- -1 + 0.5*x + eps
#d
plot(x, y)
#e
reg <- lm(y~x)
summary(reg)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.91411 -0.48230 -0.04533  0.64924  2.64157
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.9726     0.1047  -9.289 4.22e-15 ***
## x             0.5212     0.1163   4.481 2.01e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.039 on 98 degrees of freedom
## Multiple R-squared:  0.1701, Adjusted R-squared:  0.1616
## F-statistic: 20.08 on 1 and 98 DF,  p-value: 2.013e-05
```

```r
#f
plot(x, y)
abline(reg, col = "blue")
abline(a=-1, b=0.5, col = "black")
legend("topright", c("regression", "actual"), col=c("blue", "black"), lty=1, cex=0.8)
```



Comment: g. With more noise(can see from the graph), the regression line tilts wider away from the "true" process, however, it is still relatively close under this level of noise.

Problem4

```r
#refer to the lab result
# ols fit using QR
ols_fit <- function(X, y) {
  # Computer an OLS fit for linear regression using QR, returning multiple aspects of the fit
  #
  #Args:
  # x: a matrix with 1s, and other predictors
  # y: the response variable
  #
  #Returns:
  # A list of information about the ols fit, with attributes
  #   coefficients: intercept, slope1, slope2 and etc.
  #   y_values: y
  #   fitted_values
  #   residuals: y_values - fitted_values
  #   n: number of observations
  #   q: number of parameters

  #using qr to computer cofficients
  qr <- qr(X)
  q <- qr.Q(qr)
  r <- qr.R(qr)
  b <- solve(r, t(q) %*% y)

  #fitted values and residuals
  y.hat <- crossprod(t(X), b)
  resi <- y - y.hat

  return(list(coefficients=b, y_values=y, fitted_values=y.hat, residuals=resi, n=length(y), q=ncol(X)))
}
```

Testing 4

```r
fit <- ols_fit(cbind(1, mtcars$disp, mtcars$hp), mtcars$mpg)
names(fit)
```

```
## [1] "coefficients"  "y_values"      "fitted_values" "residuals"
## [5] "n"             "q"
```

```r
fit$coefficients
```

```
##             [,1]
## [1,] 30.73590425
## [2,] -0.03034628
## [3,] -0.02484008
```

```r
summary(fit$fitted_values)
```

```
##        V1
##  Min.   :11.32
##  1st Qu.:15.16
##  Median :21.64
##  Mean   :20.09
##  3rd Qu.:24.70
##  Max.   :27.15
```

```r
summary(fit$residuals)
```

```
##        V1
##  Min.   :-4.7945
##  1st Qu.:-2.3036
##  Median :-0.8246
##  Mean   : 0.0000
##  3rd Qu.: 1.8582
##  Max.   : 6.9363
```

Problem5

```r
R2 <- function(fit) {
  y.bar <- mean(fit$y_values)
  regss <- sum((fit$fitted_values - y.bar) * (fit$fitted_values - y.bar))
  tss <- sum((fit$y_values - y.bar) * (fit$y_values - y.bar))
  return(regss/tss)
}

RSE <- function(fix) {
  RSS <- sum(fit$residuals * fit$residuals)
  return(sqrt(RSS/(fit$n-fit$q)))
}
```

Testing Problem5

```r
fit <- ols_fit(cbind(1, mtcars$disp, mtcars$hp), mtcars$mpg)
R2(fit)
```

```
## [1] 0.7482402
```

```r
RSE(fit)
```

```
## [1] 3.126601
```

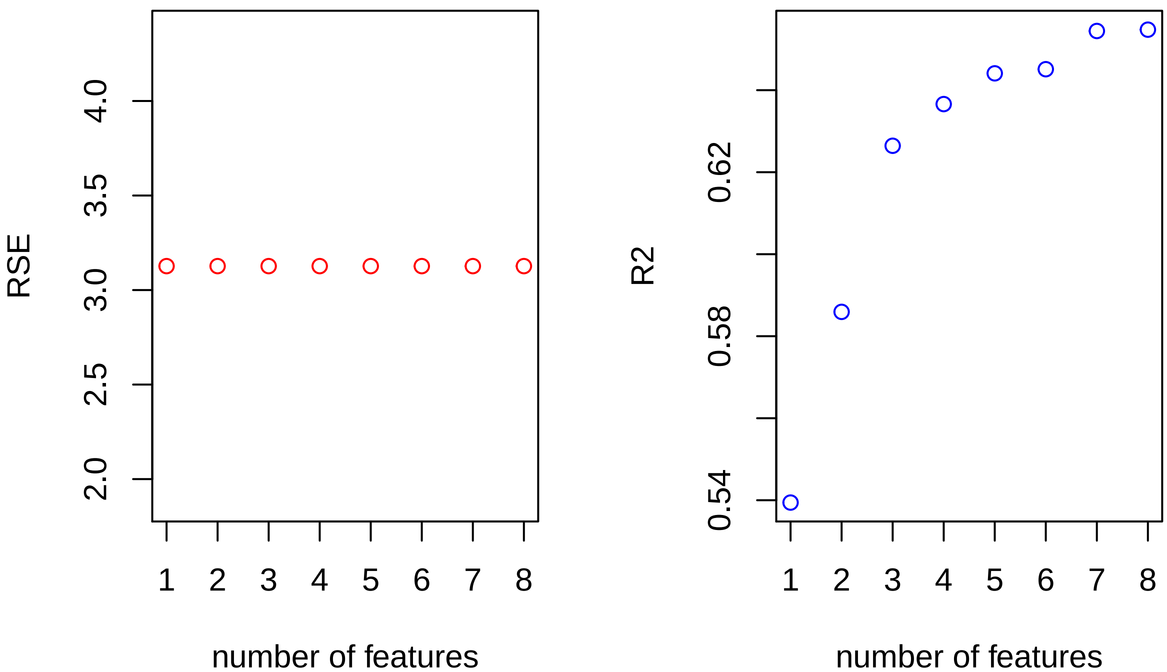
problem6

```r
#gradually adding feature and checking mse and R
prostate <- read.table("~/stat154-fall-2017/data/prostate.txt", row.names = 1)
```

```
y <- prostate$lpsa

performance <- matrix(-1, nrow=ncol(prostate)-1, ncol=2)
features <- c('lcavol', 'lweight', 'svi', 'lbph', 'age', 'lcp', 'pgg45', 'gleason')
for (i in 1:length(features)) {
  X <- cbind(1, prostate[,features[1:i]])
  reg.temp <- ols_fit(X, y)
  performance[i, 1] <- RSE(reg.temp)
  performance[i, 2] <- R2(reg.temp)
}
rownames(performance) <- features
colnames(performance) <- c("RSE", "R2")

par(mfrow=c(1,2))
plot(performance[,1],
     col = "red", xlab = "number of features", ylab = "RSE")
plot(performance[,2],
     col = "blue", xlab = "number of features", ylab = "R2")
```



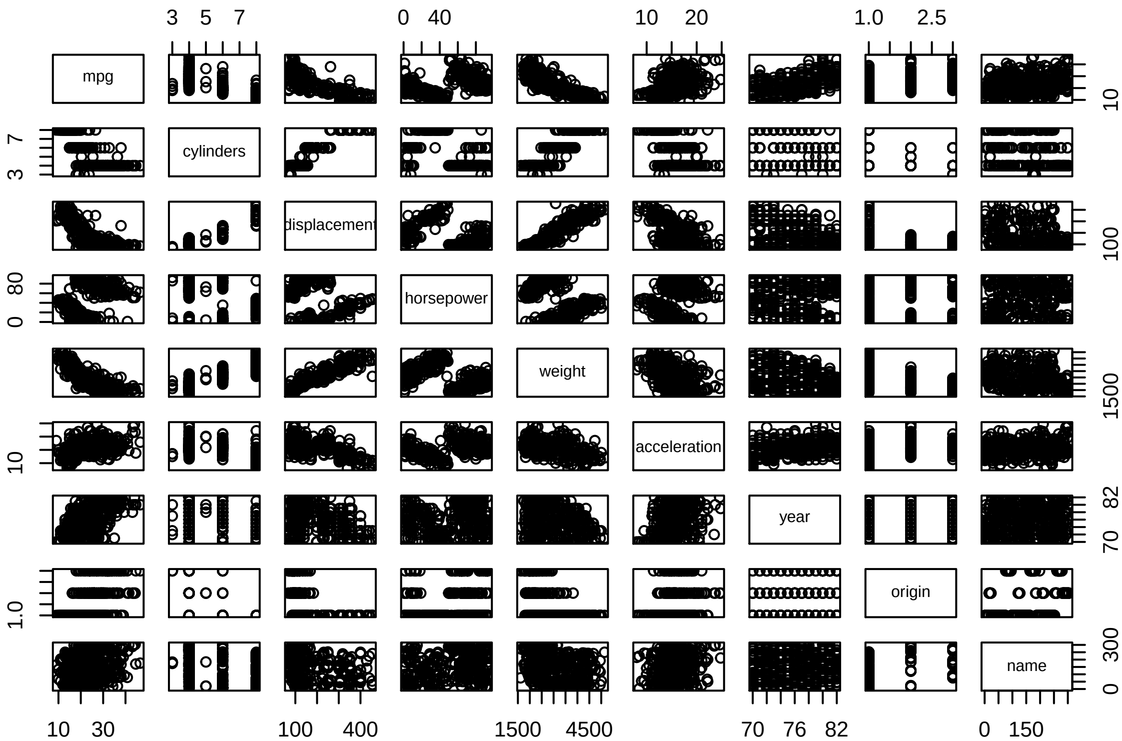
As we add more features, we tend to overfit the model(reducing our errors) but RSE remains the same(the standard deviation of our errors). In other words, as we add more features, we are unable to narrow down the errors of our forecasts.

problem7

```
auto <- read.table("http://www-bcf.usc.edu/~gareth/ISL/Auto.data", header = T)
auto$horsepower <- as.numeric(auto$horsepower)

auto.quan <- dplyr::select(auto, -name)
#a
```

```r
pairs(auto)
```




```r
#b
cor.m <- cor(auto.quan)
cor.m
```

```
##                     mpg  cylinders displacement horsepower     weight
## mpg           1.0000000 -0.7762599   -0.8044430  0.4228227 -0.8317389
## cylinders    -0.7762599  1.0000000    0.9509199 -0.5466585  0.8970169
## displacement -0.8044430  0.9509199    1.0000000 -0.4820705  0.9331044
## horsepower    0.4228227 -0.5466585   -0.4820705  1.0000000 -0.4821507
## weight       -0.8317389  0.8970169    0.9331044 -0.4821507  1.0000000
## acceleration  0.4222974 -0.5040606   -0.5441618  0.2662877 -0.4195023
## year          0.5814695 -0.3467172   -0.3698041  0.1274167 -0.3079004
## origin        0.5636979 -0.5649716   -0.6106643  0.2973734 -0.5812652
##              acceleration       year     origin
## mpg             0.4222974  0.5814695  0.5636979
## cylinders      -0.5040606 -0.3467172 -0.5649716
## displacement   -0.5441618 -0.3698041 -0.6106643
## horsepower      0.2662877  0.1274167  0.2973734
## weight         -0.4195023 -0.3079004 -0.5812652
## acceleration    1.0000000  0.2829009  0.2100836
## year            0.2829009  1.0000000  0.1843141
## origin          0.2100836  0.1843141  1.0000000
```

c.

```r
auto.reg <- lm(mpg~., data=auto.quan)
summary(auto.reg)
```

```
##
## Call:
```

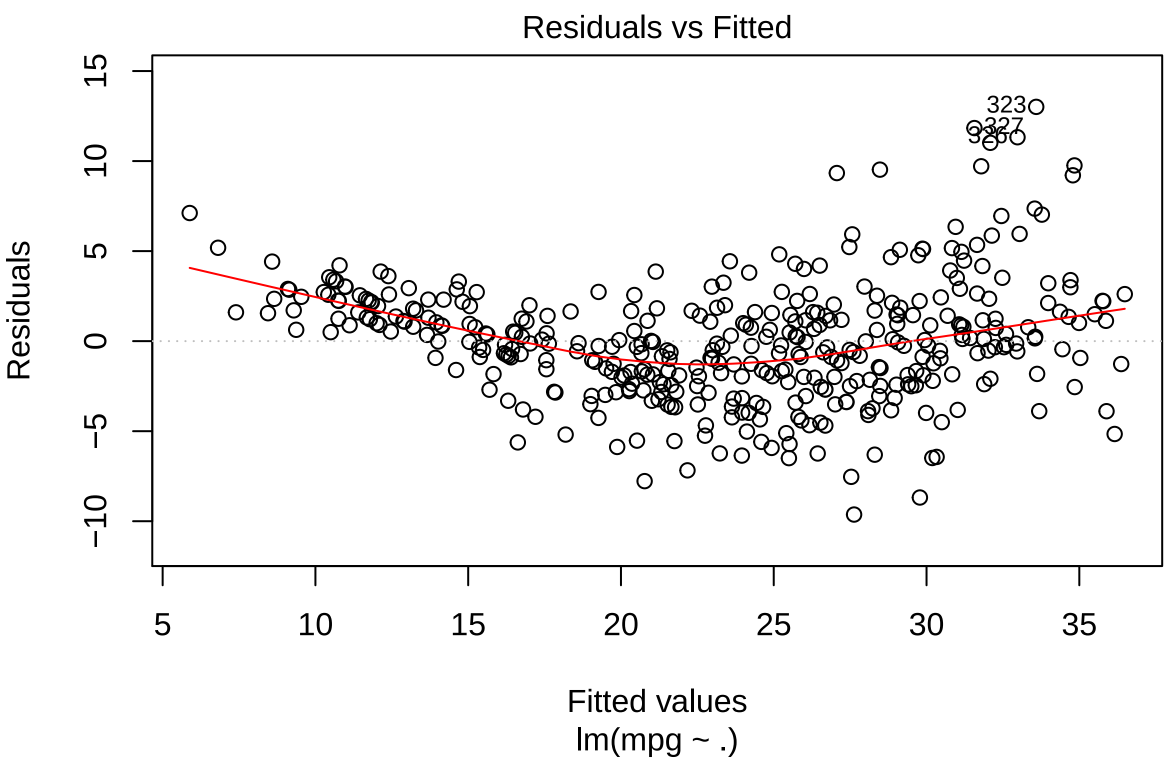
```
## lm(formula = mpg ~ ., data = auto.quan)
##
## Residuals:
##    Min      1Q  Median      3Q     Max
## -9.629 -2.034 -0.046  1.801 13.010
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -2.128e+01  4.259e+00  -4.998 8.78e-07 ***
## cylinders    -2.927e-01  3.382e-01  -0.865   0.3874
## displacement  1.603e-02  7.284e-03   2.201   0.0283 *
## horsepower    7.942e-03  6.809e-03   1.166   0.2442
## weight       -6.870e-03  5.799e-04 -11.846  < 2e-16 ***
## acceleration  1.539e-01  7.750e-02   1.986   0.0477 *
## year          7.734e-01  4.939e-02  15.661  < 2e-16 ***
## origin        1.346e+00  2.691e-01   5.004 8.52e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.331 on 389 degrees of freedom
## Multiple R-squared:  0.822,  Adjusted R-squared:  0.8188
## F-statistic: 256.7 on 7 and 389 DF,  p-value: < 2.2e-16
```

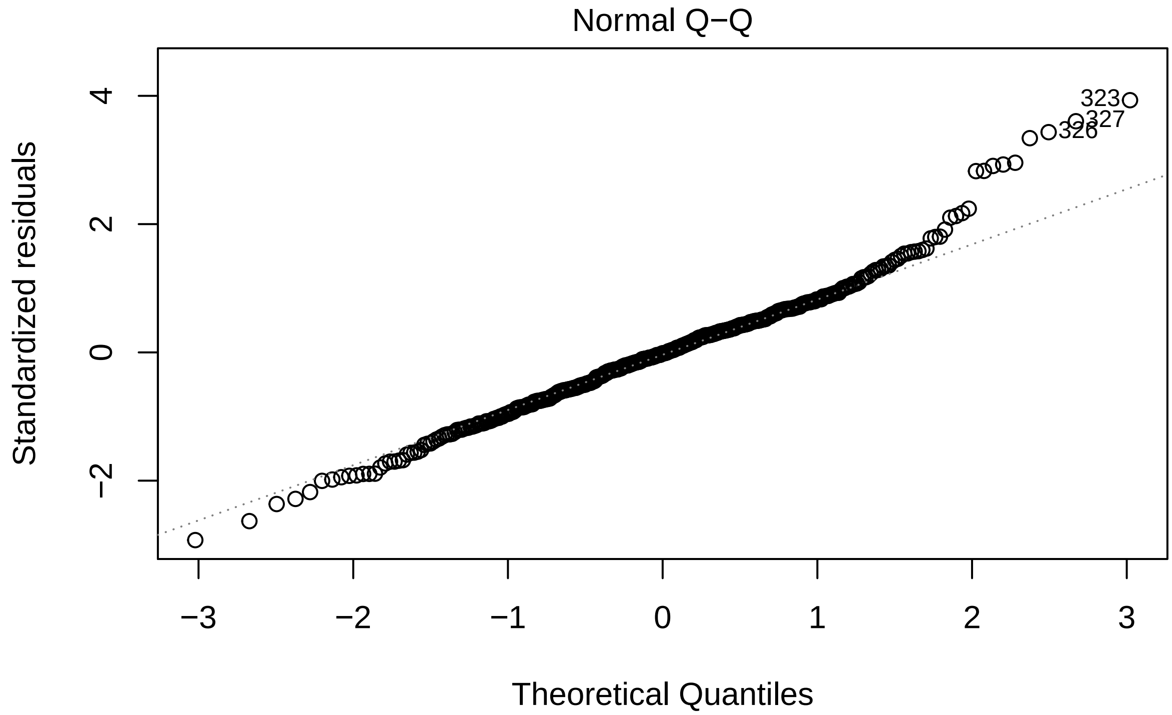
there is a relationship btw the predictors and response because the adjusted Rsquare is pretty high and F-test is statistically significant.

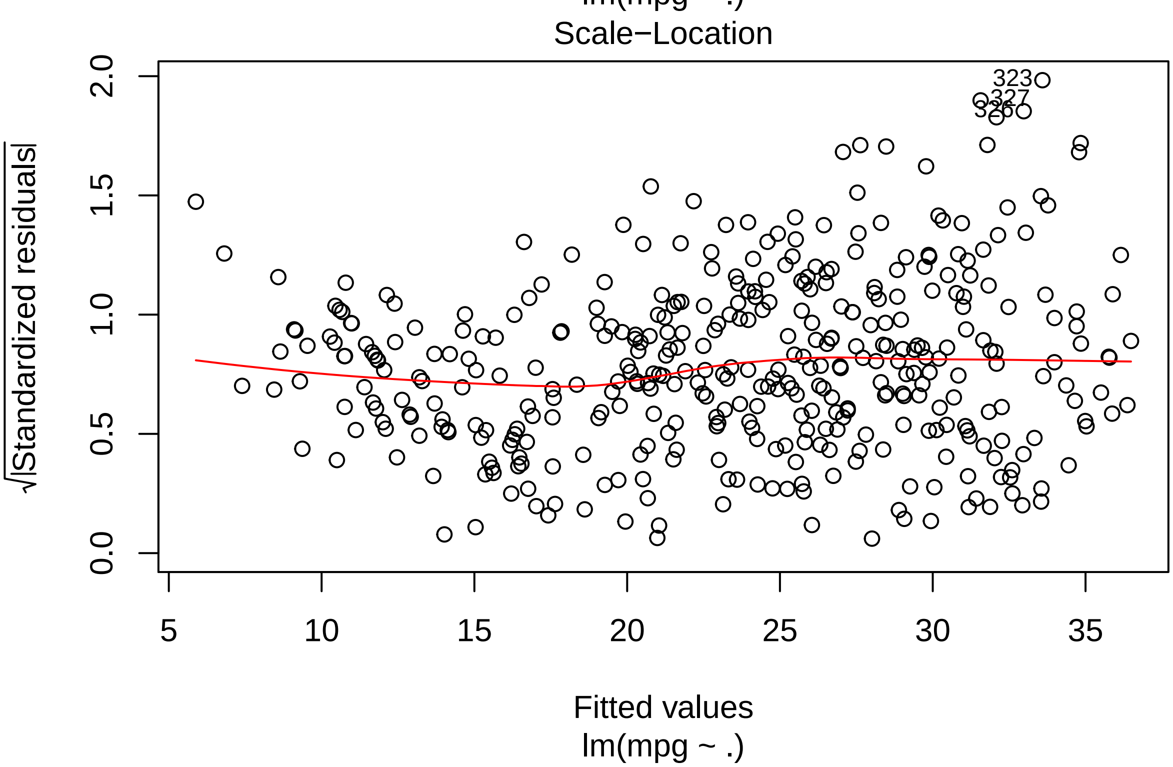
displacement, weight, acceleration, year and origin are statistically significant predictors.

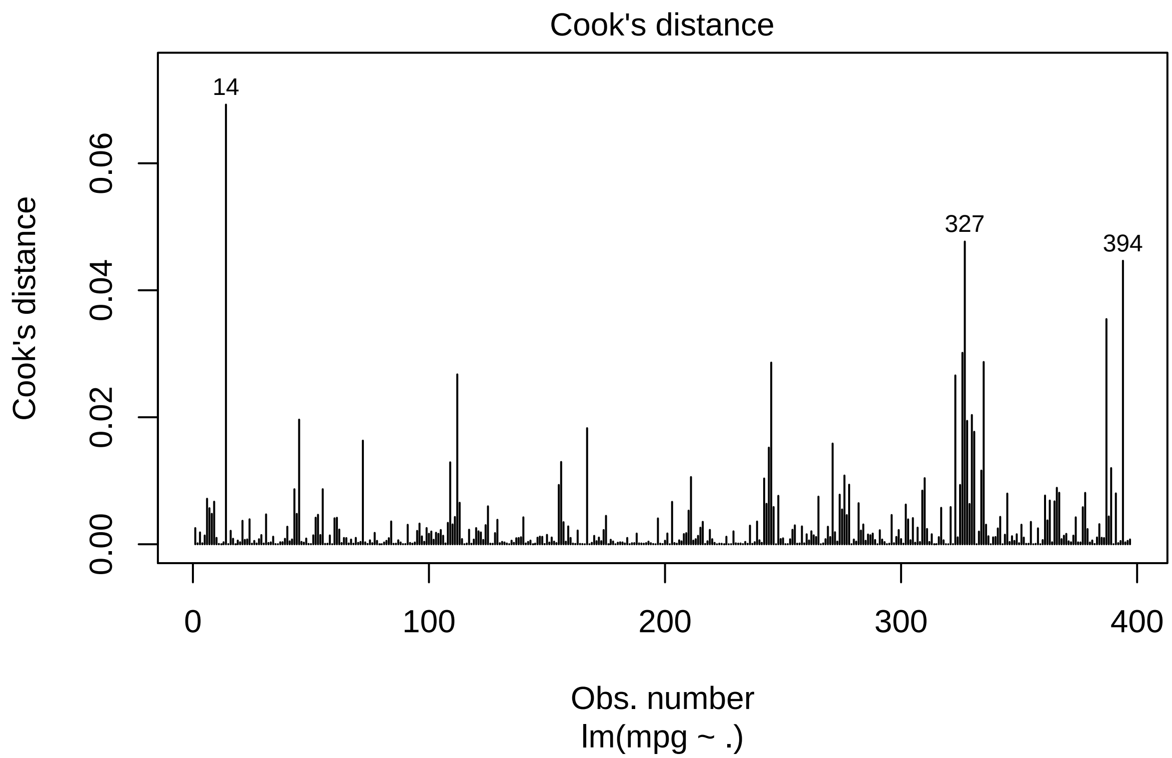
year's coefficient suggests for each increase year(of production), the vehicle's mpg increases by 0.7734.

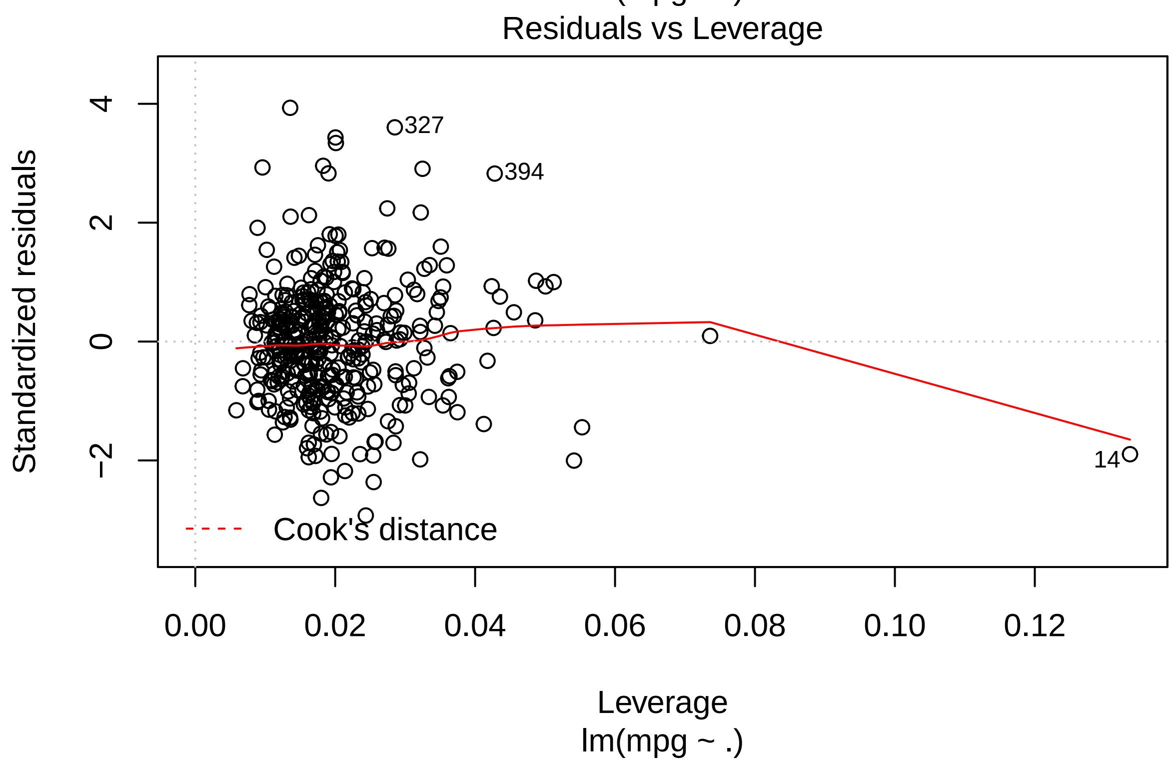
```
#d
plot(auto.reg, which=1:6)
```

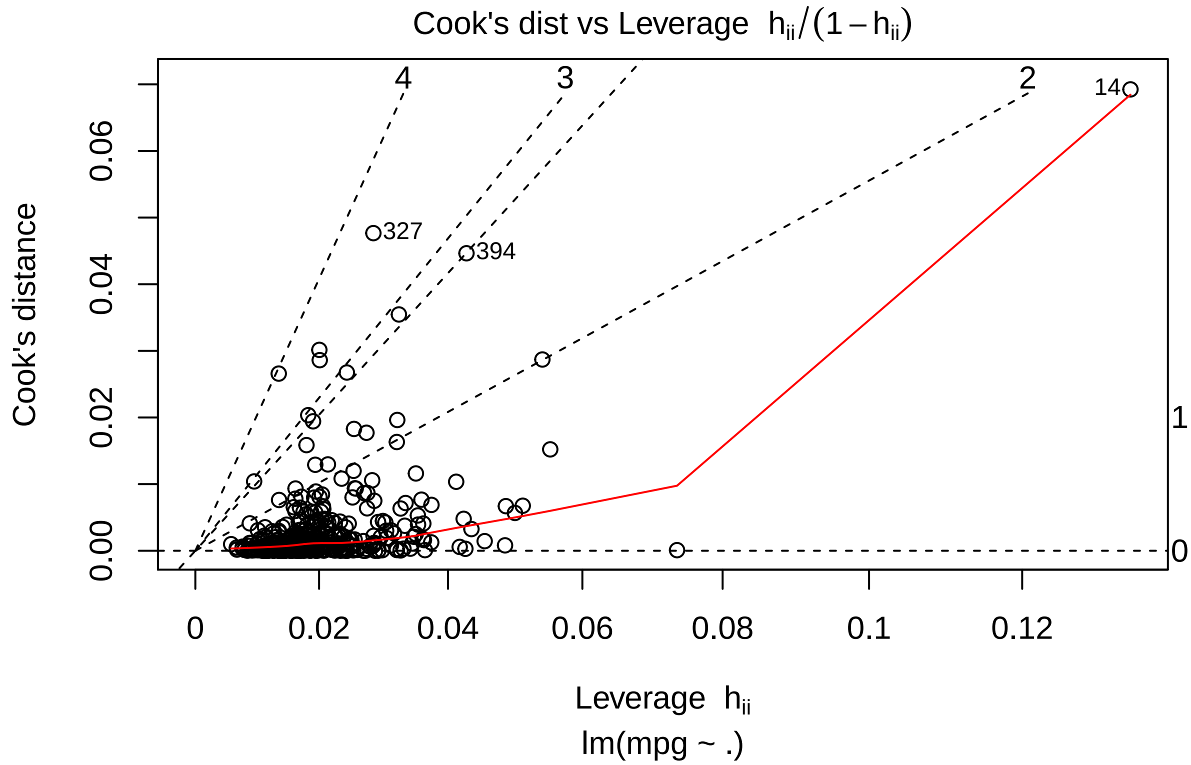



Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(mpg ~ .)
323
327
326
−3
−2
−1
0
1
2
3
4


Scale−Location
√|Standardized residuals|
Fitted values
lm(mpg ~ .)
323
327
326
0.0
0.5
1.0
1.5
2.0
5
10
15
20
25
30
35


Cook's distance
Cook's distance
0.00
0.02
0.04
0.06
14
327
394
0
100
200
300
400
Obs. number
lm(mpg ~ .)


Residuals vs Leverage
Standardized residuals
−2
0
2
4
327
394
14
Cook's distance
0.00
0.02
0.04
0.06
0.08
0.10
0.12
Leverage
lm(mpg ~ .)



Comments: The residual plot suggests observation 323, 326, 327 have high (positive) residuals(and a few unlabeledp points). Among them(obs. 323, 326, 327), 327 and 394 has relatively high cook's distance(influence on the parameters), along with obs. 14, with abnormally high cook's distance and leverage.

Own Obersvations:
High leverage points doesn't ensure high influence on the model(but many of them do).

e. modeling with interaction effect

```
auto.reg.inter <- auto.reg.test <- lm(mpg~. + displacement*weight,
                   data = auto.quan
                   )
summary(auto.reg.inter)
```

```
##
## Call:
## lm(formula = mpg ~ . + displacement * weight, data = auto.quan)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.8561 -1.8167 -0.0141  1.7027 12.1594
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)         -1.130e+01  3.948e+00  -2.863  0.00443 **
## cylinders            2.463e-01  3.079e-01   0.800  0.42424
## displacement        -7.153e-02  1.104e-02  -6.479  2.8e-10 ***
## horsepower           2.114e-03  6.129e-03   0.345  0.73029
## weight              -1.127e-02  6.854e-04 -16.437  < 2e-16 ***
```

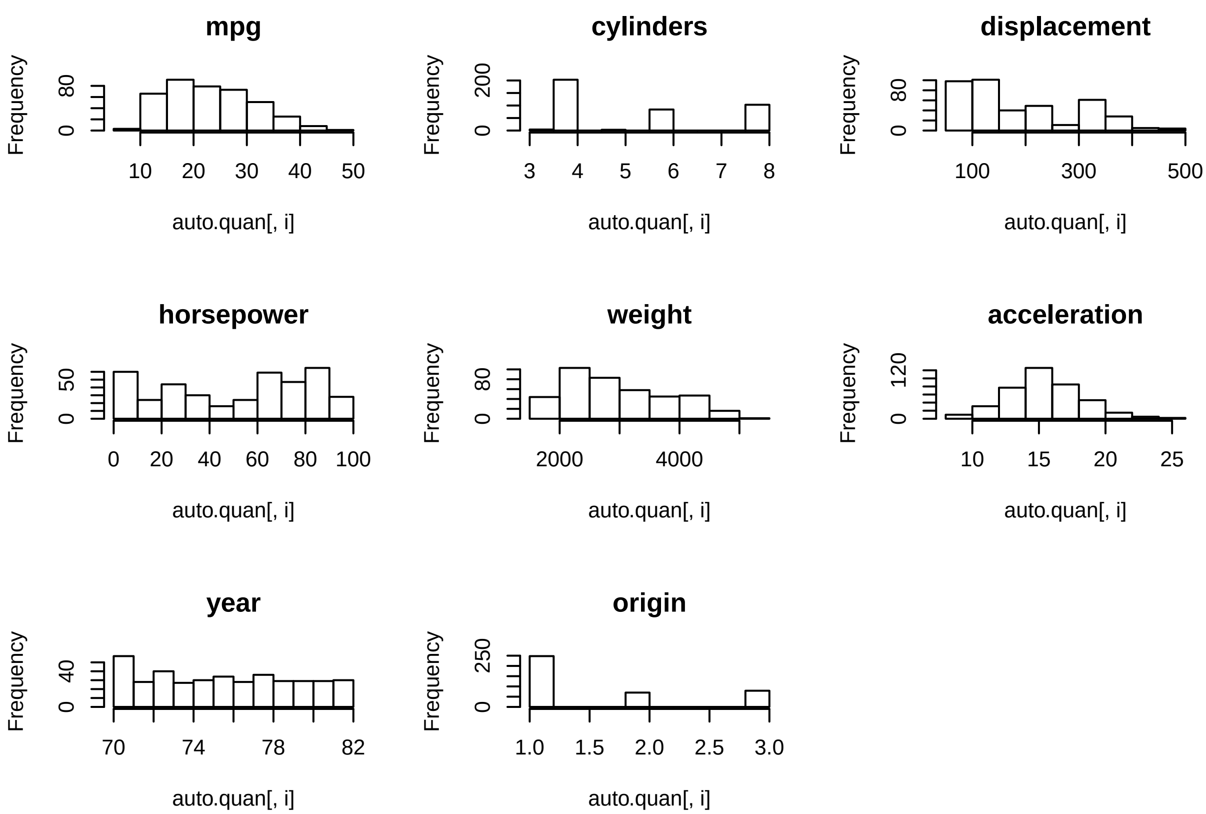
```
## acceleration         2.100e-01  6.966e-02   3.014  0.00275 **
## year                 8.181e-01  4.448e-02  18.394  < 2e-16 ***
## origin               4.428e-01  2.580e-01   1.716  0.08687 .
## displacement:weight  2.212e-05  2.249e-06   9.833  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.984 on 388 degrees of freedom
## Multiple R-squared:  0.8575, Adjusted R-squared:  0.8546
## F-statistic: 291.9 on 8 and 388 DF,  p-value: < 2.2e-16
```

Year, I choose displacement*weight. Things I find:
using the "cateogrical" numerical variable(categorical variable that are numerically encoded) is kind of a mess. I have a 0.4 improvment in RSE and 0.04 improvement in adjusted R2.

f. variable transformation

First check the distribution of the variables

```r
par(mfrow=c(3,3))
for(i in 1:8) {
  hist(auto.quan[,i], main = colnames(auto.quan)[i])
}
```



best results from trying out new things

```r
auto.reg.test <- lm(mpg~.,
                    data = auto.quan %>%
                      #deleting extra features that does not marginally improve performance
                      select(-c(horsepower, displacement, acceleration)) %>%
                      #factorizing the cateogrical variables
                      mutate(cylinders = factor(cylinders)) %>%
                      mutate(origin = factor(origin)) %>%
                      #applying log to the right skewed variable
                      mutate(mpg = log(mpg)) %>%
                      mutate(weight = log(weight)))


summary(auto.reg.test)
```

```
##
## Call:
## lm(formula = mpg ~ ., data = auto.quan %>% select(-c(horsepower,
##     displacement, acceleration)) %>% mutate(cylinders = factor(cylinders)) %>%
##     mutate(origin = factor(origin)) %>% mutate(mpg = log(mpg)) %>%
##     mutate(weight = log(weight)))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.35774 -0.06612  0.00116  0.06183  0.42835
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.655184   0.379617  17.531  < 2e-16 ***
## cylinders4   0.270753   0.057755   4.688 3.83e-06 ***
## cylinders5   0.353132   0.088252   4.001 7.54e-05 ***
## cylinders6   0.206360   0.059956   3.442 0.000641 ***
## cylinders8   0.180068   0.062750   2.870 0.004335 **
## weight      -0.779648   0.045703 -17.059  < 2e-16 ***
## year         0.031631   0.001665  18.992  < 2e-16 ***
## origin2      0.037728   0.018444   2.046 0.041475 *
## origin3      0.038381   0.018203   2.108 0.035629 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1118 on 388 degrees of freedom
## Multiple R-squared:  0.8941, Adjusted R-squared:  0.8919
## F-statistic: 409.3 on 8 and 388 DF,  p-value: < 2.2e-16
```

After few minutes of tweakings, I find that:

1. log(mpg) has better performance than log2(mpg), but log transformation helps in general since the distribution of mpg is right skewed.

2. factorizing categorical variables help in this case
3. agumenting old features may make some features less significant(may due to multicolinearity)
4. scaling a variable doesn't really affect the performance. 5.RSE maybe misleading since log scale will reduce the scale of the error