# HW2

*Ramtin Boustani - SUID# 05999261*

## Problem 1

### Exercise 1 from section 10.7

### (a)

For simiplicity ignoed $\sum$

$(x_{ij} - x_{i\hat{j}})^2$
$= ((x_{ij} - \overline{x_{kj}}) - (x_{i\hat{j}} - \overline{x_{kj}}))^2$
$= ((x_{ij} - \overline{x_{kj}})^2 + (x_{i\hat{j}} - \overline{x_{kj}})^2 - 2(x_{ij} - \overline{x_{kj}})(x_{i\hat{j}} - \overline{x_{kj}}))$
If expanding the above statement and assuming $(x_{ij})^2 = (x_{i\hat{j}})^2$ we will we have the other side statement
$= 0 + 2(x_{ij} - \overline{x_{kj}})^2$

### (b)

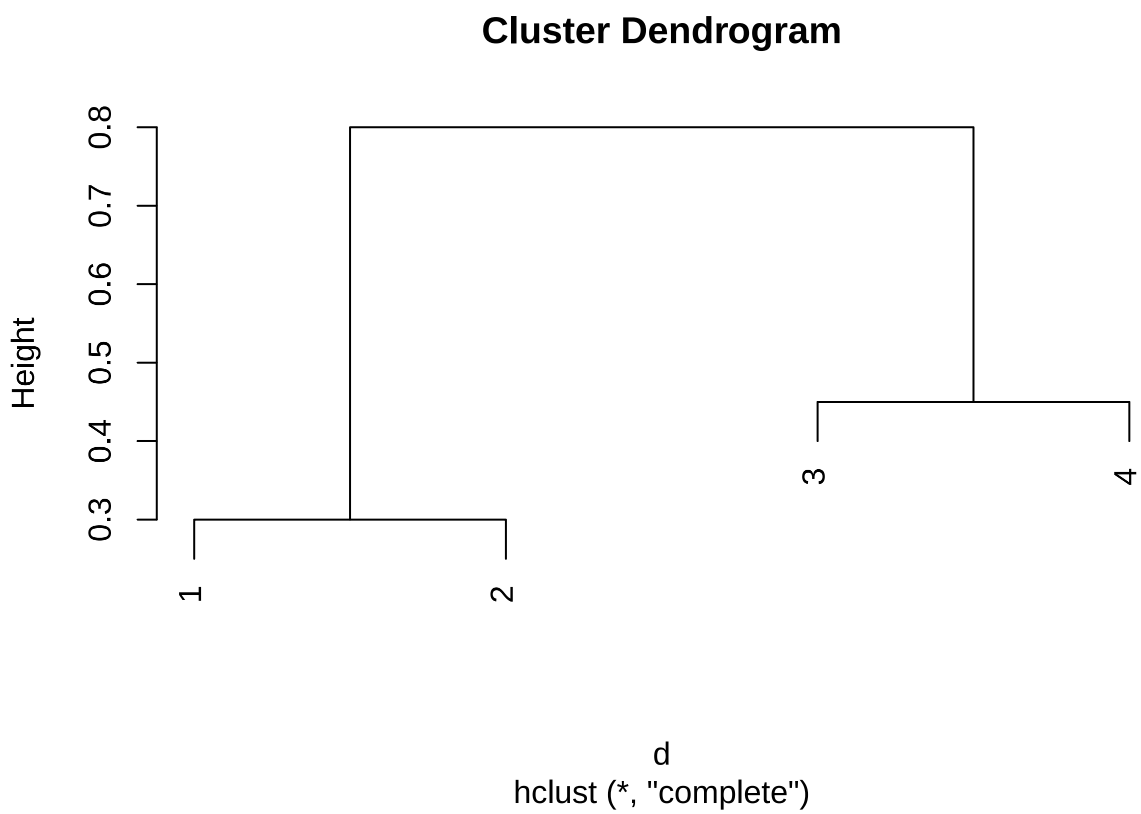
Algorithm 10.1 is a iterative gready algorithm that converge to the local minimum. In every step by assiging ponits to each cluster's centroid we are minmizing inside cluster variance and in genral decreasing distance between clusters.

## Problem 2

### Exercise 2 from section 10.7

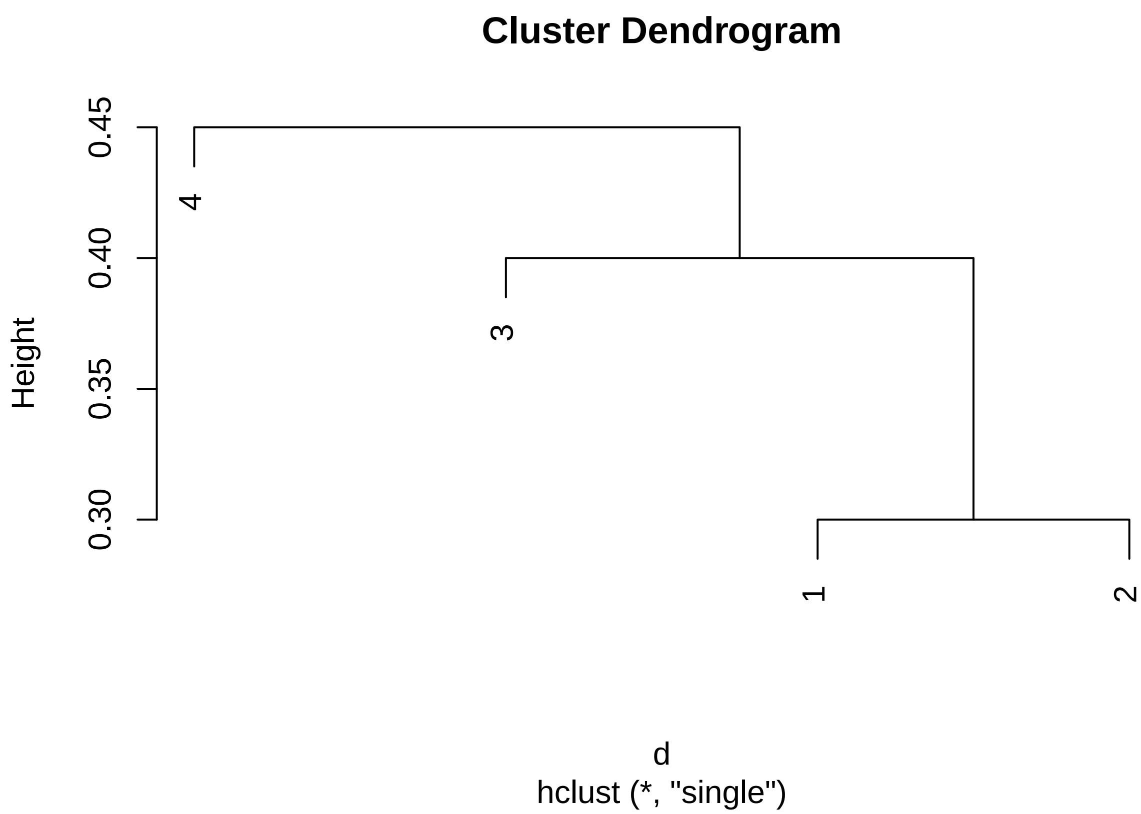
### (a)

```
C = matrix( c(0, 0.3, 0.4, 0.7,
              0.3, 0, 0.5, 0.8,
              0.4, 0.5, 0, 0.45,
              0.7, 0.8, 0.45, 0),
            nrow = 4, ncol = 4)
d = as.dist(C)
hc = hclust(d, method = "complete")
plot(hc)
```




**(b)**

```r
hc = hclust(d, method = "single")
plot(hc)
```




**(c)**

(1,2) –> cluster A
(3,4) –> clustet B

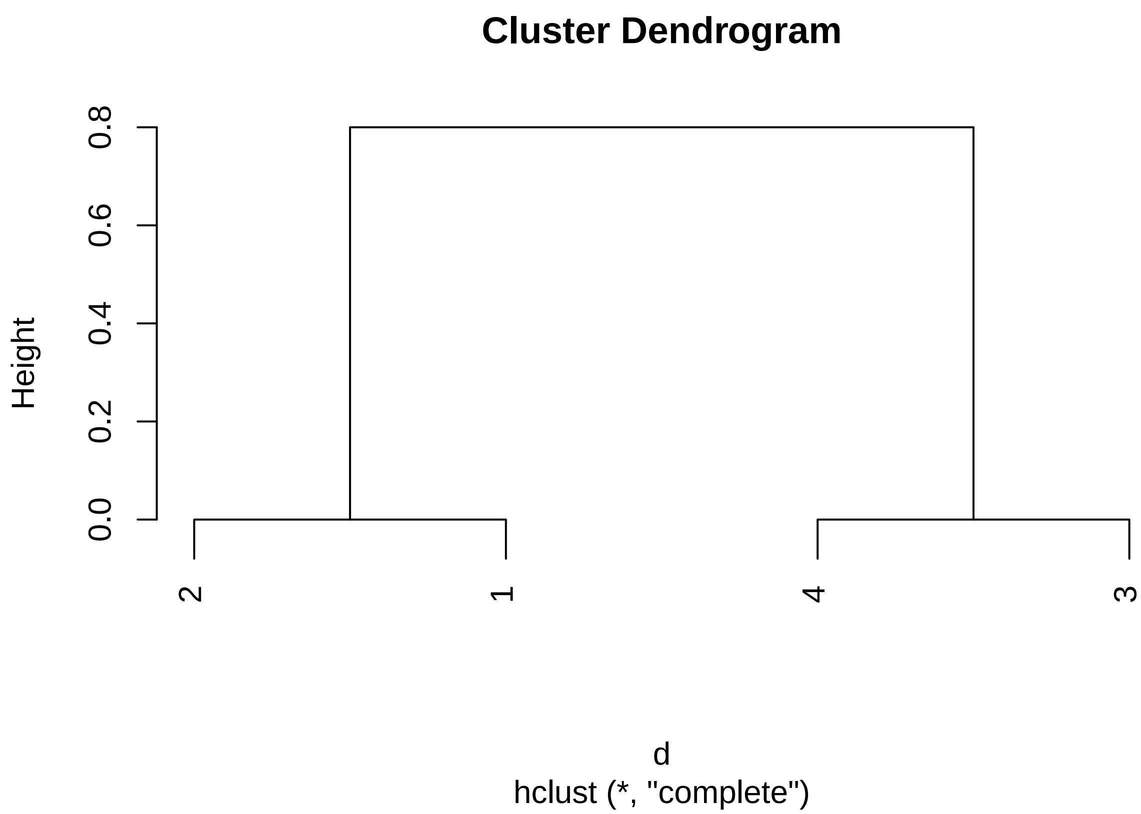
**(d)**

((1,2), 3) –> cluster A
(4) –> clustet B

**(e)**

Swapping columns 1 & 2
Swapping columns 3 & 4

```
C2 = matrix(c(C[,2], C[,1], C[,4], C[,3]), nrow = 4, ncol = 4, byrow = FALSE)
d = as.dist(C2)
hc = hclust(d, method = "complete")
plot(hc, labels=c(2,1,4,3))
```

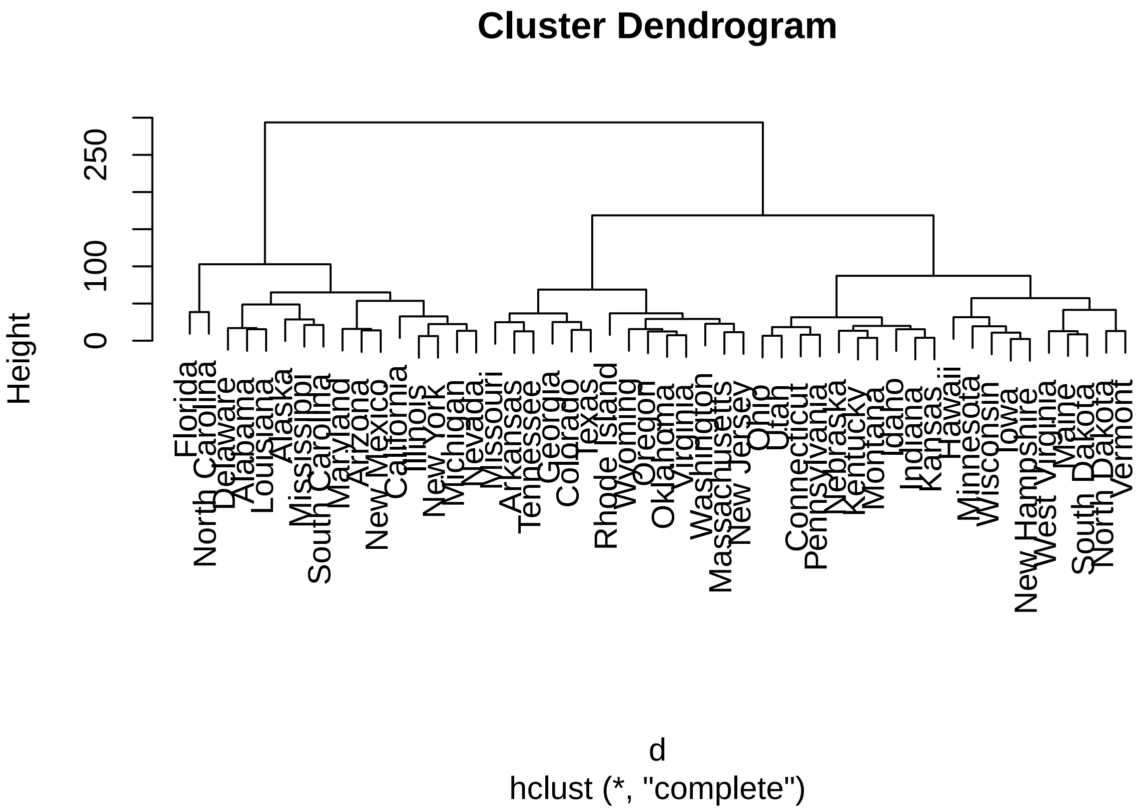



## Problem 3

**Exercise 9 from section 10.7**

**(a)**

```
d = dist(USArrests, method = "euclidean")
hc.complete = hclust(d, method = "complete")
plot(hc.complete)
```



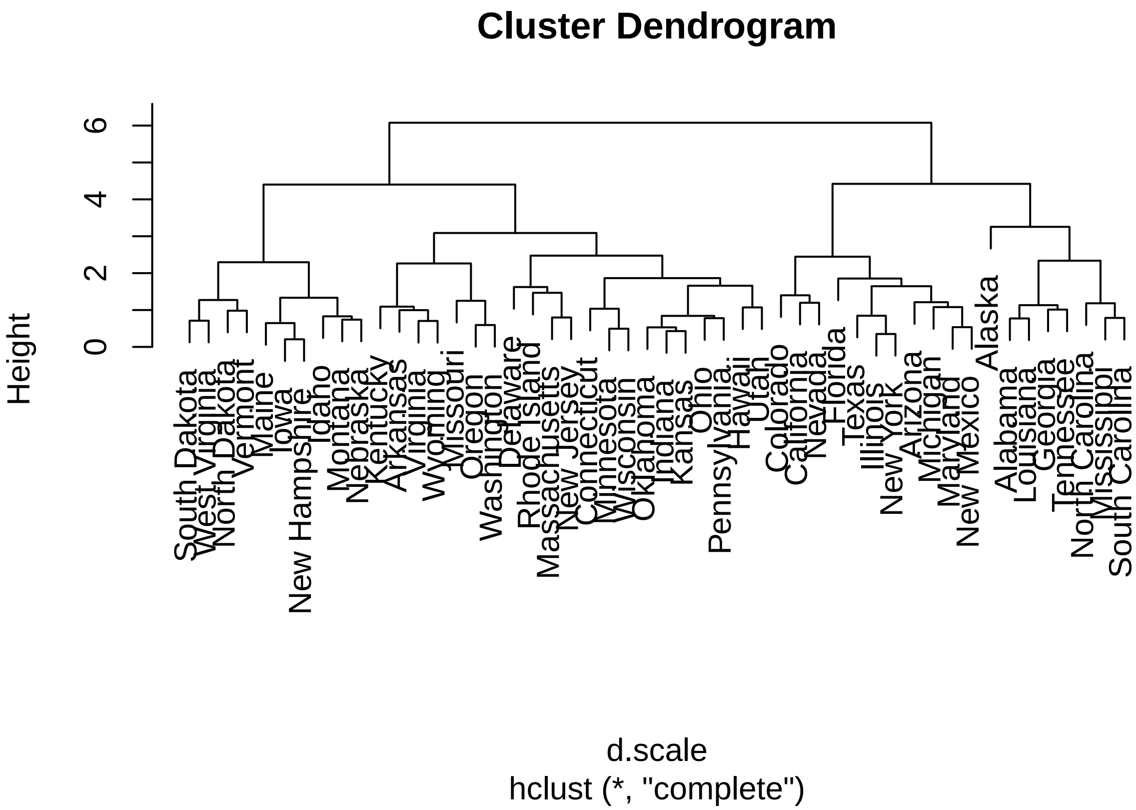

**(b)**

```
cutree(hc.complete, k=3)
```

```
##        Alabama         Alaska        Arizona       Arkansas     California
##              1              1              1              2              1
##       Colorado    Connecticut       Delaware        Florida        Georgia
##              2              3              1              1              2
##         Hawaii          Idaho       Illinois        Indiana           Iowa
##              3              3              1              3              3
##         Kansas       Kentucky      Louisiana          Maine       Maryland
##              3              3              1              3              1
##  Massachusetts       Michigan      Minnesota    Mississippi       Missouri
##              2              1              3              1              2
##        Montana       Nebraska         Nevada  New Hampshire     New Jersey
##              3              3              1              3              2
##     New Mexico       New York North Carolina   North Dakota           Ohio
##              1              1              1              3              3
##       Oklahoma         Oregon   Pennsylvania   Rhode Island South Carolina
##              2              2              3              2              1
##   South Dakota      Tennessee          Texas           Utah        Vermont
##              3              2              2              3              3
##       Virginia     Washington  West Virginia      Wisconsin        Wyoming
##              2              2              3              3              2
```

**(c)**

```r
d.scale = dist(scale(USArrests), method = "euclidean")
hc.complete.scale = hclust(d.scale, method = "complete")
plot(hc.complete.scale)
```




**(d)**

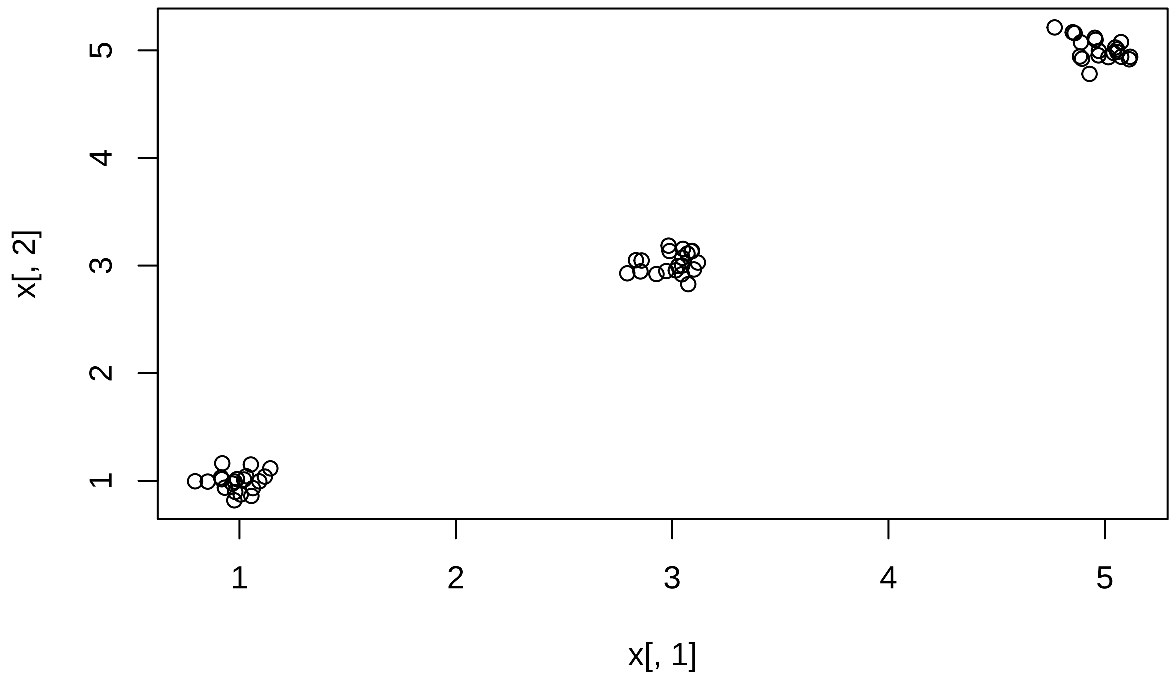
Since UrbanPop column has a different unit with other columns, scaling variables is a good idea to standardizing data. After scaling the height changed from 300 to 60 and states that have been clustered together also changed!

## Problem 4

**Exercise 10 from section 10.7**

**(a)**

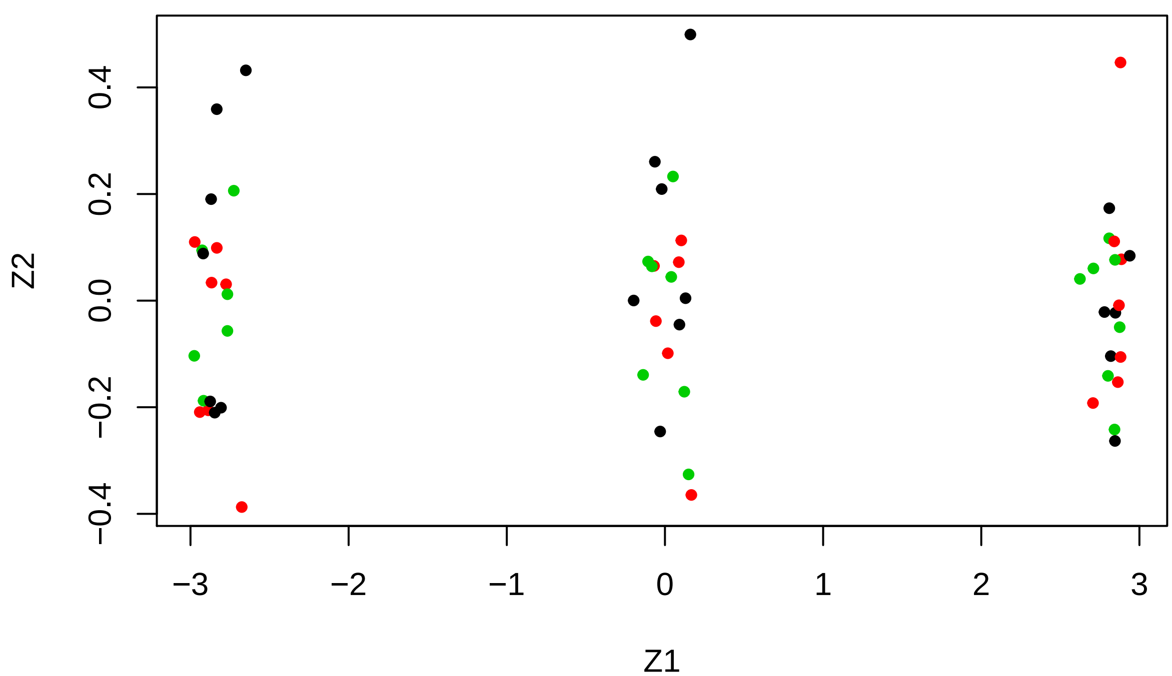
```r
set.seed(101)
x = matrix(rnorm(60*50, mean=0, sd=0.1), 60,50)
x[1:20,1]=1+x[1:20,1]
x[1:20,2]=1+x[1:20,2]
x[21:40,1]=3+x[21:40,1]
x[21:40,2]=3+x[21:40,2]
x[41:60,1]=5+x[41:60,1]
x[41:60,2]=5+x[41:60,2]
plot(x[,1],x[,2])
```




```
true.labels <- c(rep(1, 20), rep(2, 20), rep(3, 20))
```

**(b)**

```
pr.out = prcomp(x)
plot(pr.out$x[,1:2], col = 1:3, xlab = "Z1", ylab = "Z2", pch=20)
```



**(c)**

```r
km.out=kmeans(x,3, nstart=20)
table(true.labels, km.out$cluster)
```

```
##
## true.labels  1  2  3
##           1  0  0 20
##           2  0 20  0
##           3 20  0  0
```

**(d)**

```r
km.out=kmeans(x,2, nstart=20)
km.out$cluster
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [36] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

```r
table(true.labels, km.out$cluster)
```

```
##
## true.labels  1  2
##           1 20  0
##           2  0 20
##           3  0 20
```

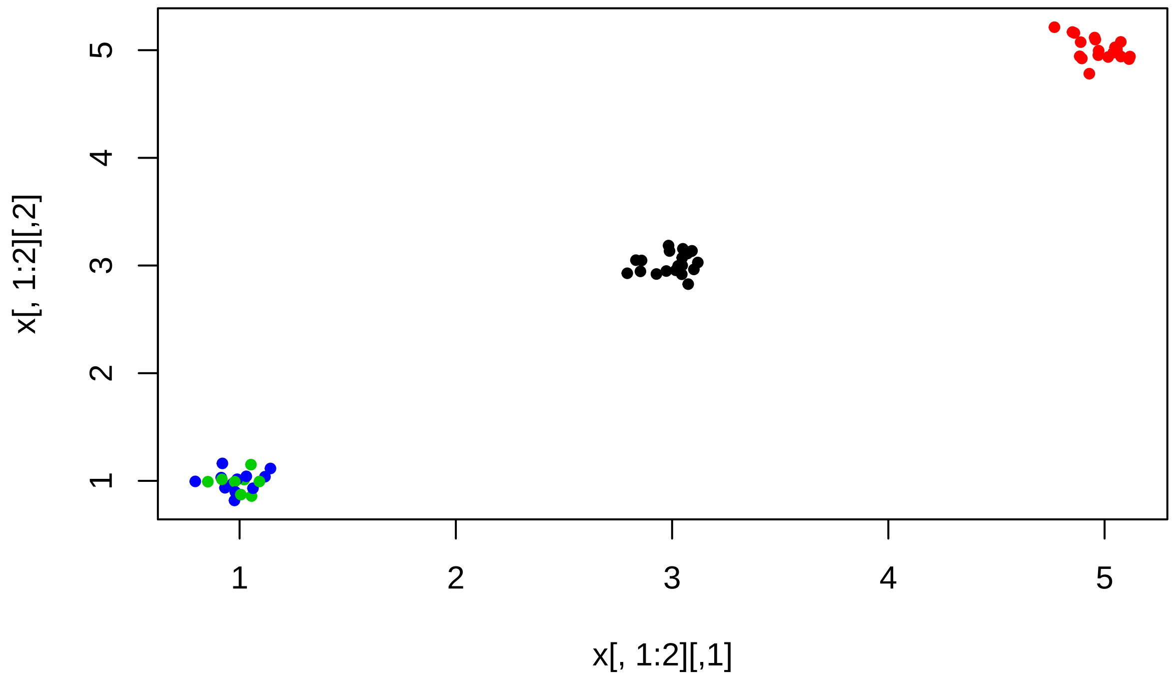
clusters of 2 & 3 are merged!

**(e)**

```r
km.out=kmeans(x,4, nstart=20)
km.out$cluster
```

```
##  [1] 4 3 4 3 4 4 4 4 3 3 3 4 4 3 4 4 4 3 3 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [36] 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

```r
plot(x[,1:2], col= km.out$cluster, pch=20)
```



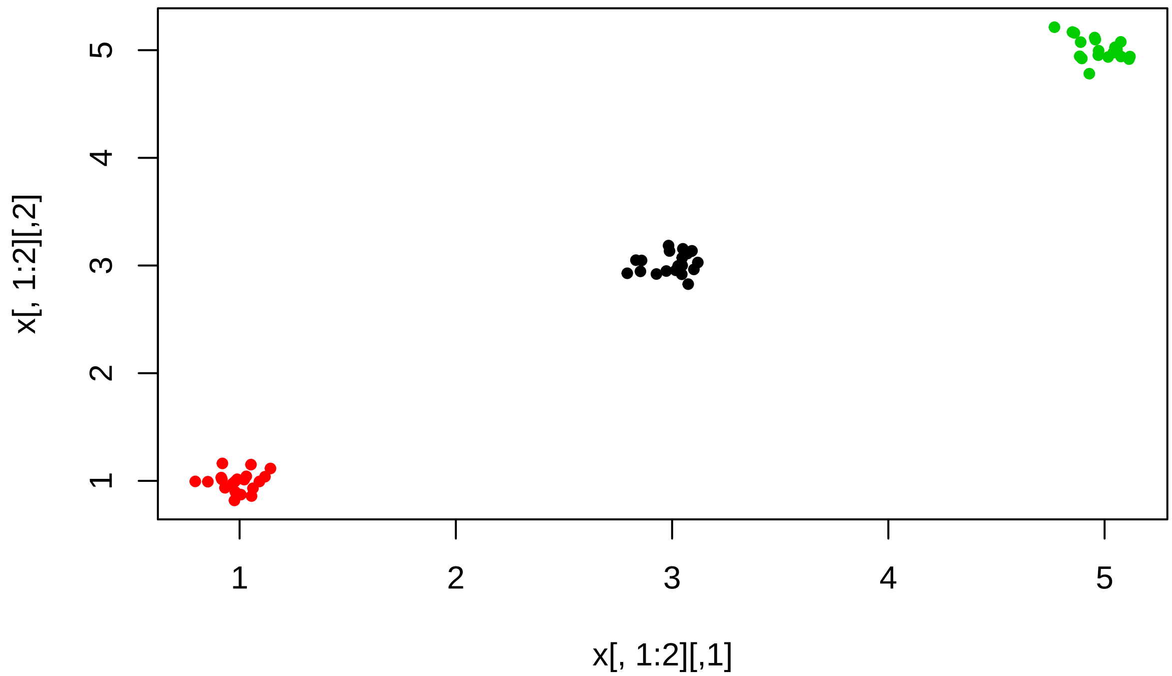

Cluster 1 splitted in 2 clusters of 1 & 4

**(f)**

```
km.out = kmeans(pr.out$x[,1:2], 3, nstart = 20)
table(true.labels, km.out$cluster)
```

```
##
## true.labels  1  2  3
##           1  0 20  0
##           2 20  0  0
##           3  0  0 20
```

```
plot(x[,1:2], col= km.out$cluster, pch=20)
```

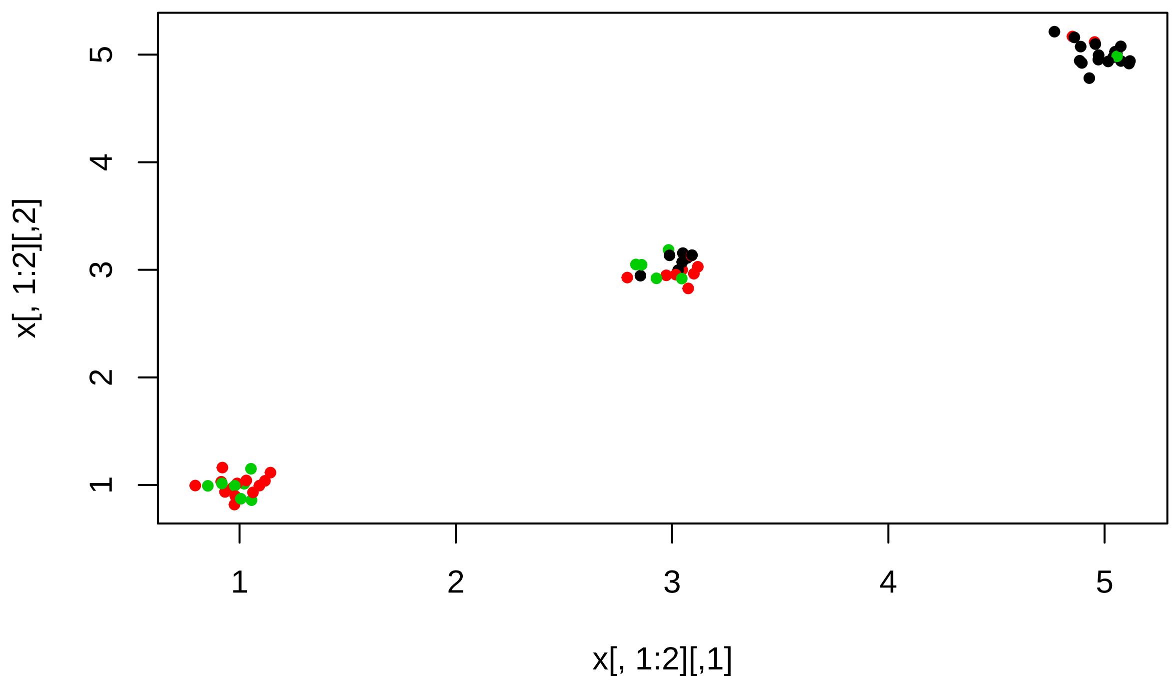



(g)

```
km.out = kmeans(scale(x) , 3, nstart = 20)
table(true.labels, km.out$cluster)
```

```
##
## true.labels  1  2  3
##           1  0 13  7
##           2  7  8  5
##           3 17  2  1
```

```
plot(x[,1:2], col= km.out$cluster, pch=20)
```




Not good clutering result due to minmizing distance between observations

## Problem 5

### Exercise 4 from section 3.7

#### (a)

Cubic regression has a lower traing RSS because it has more parameters to fit the line

#### (b)

Linear regression has a lower testing RSS because cubic regression has overfit issue using training data

#### (c)

Cubic regression has a lower traing RSS (same to (a)) because it has more parameter can can fit the line better

#### (d)

Not enough inforamtion it really depends on true relation betwen Xs and Y.

## Problem 6

### Exercise 9, parts (a)-(d) only

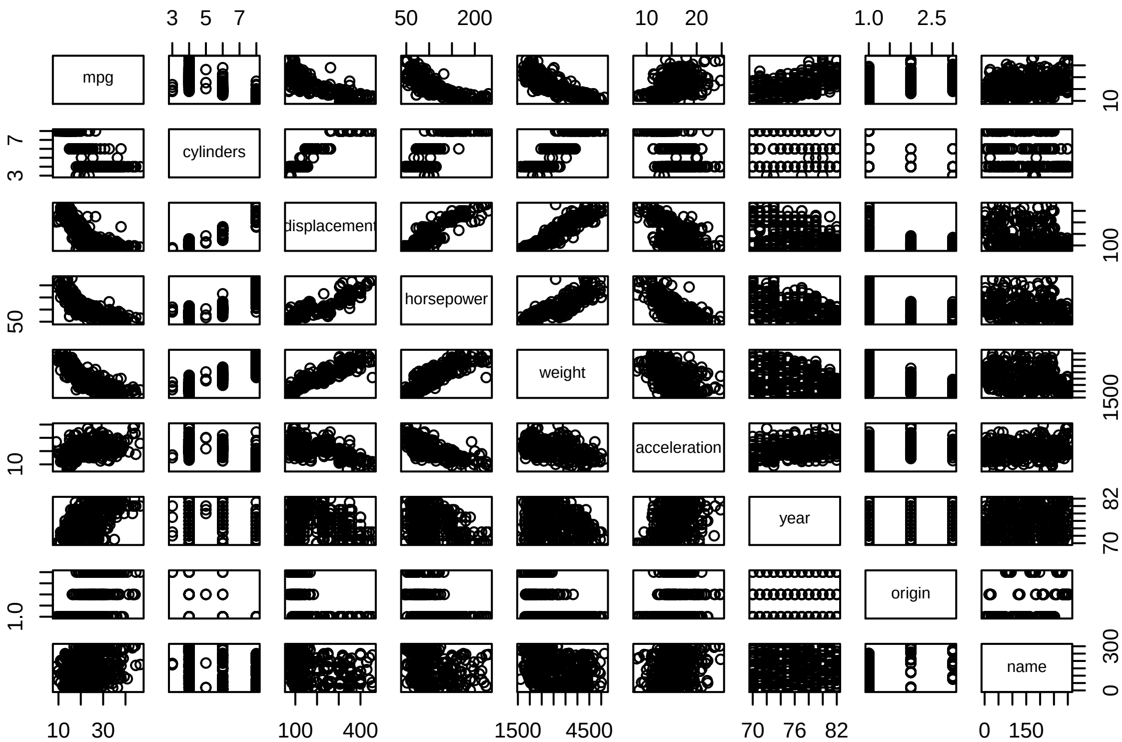
```r
install.packages("ISLR", repos = "http://cran.us.r-project.org")
```

```
##
## The downloaded binary packages are in
##  /var/folders/nm/g09p009s0qb231f3_hsjz7r40011sl/T//Rtmpo9Gk2J/downloaded_packages
```

```r
library(ISLR)
auto = ISLR::Auto
```

**(a)**

```r
pairs(auto)
```




**ba)**

```r
cor(auto[0:8])
```

```
##                     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)

*mpg* ∼ . mpg is Y and . means all columns -*name* : means remove name from data

```r
result = lm( formula = mpg ~ . -name , data=auto)
summary(result)
```

```
##
## 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
```

##### i.

$H_0$ = all parameters are zero
Big F-statistic and small p-value are showing null hypothesis is not correct so there is relationship

##### ii.

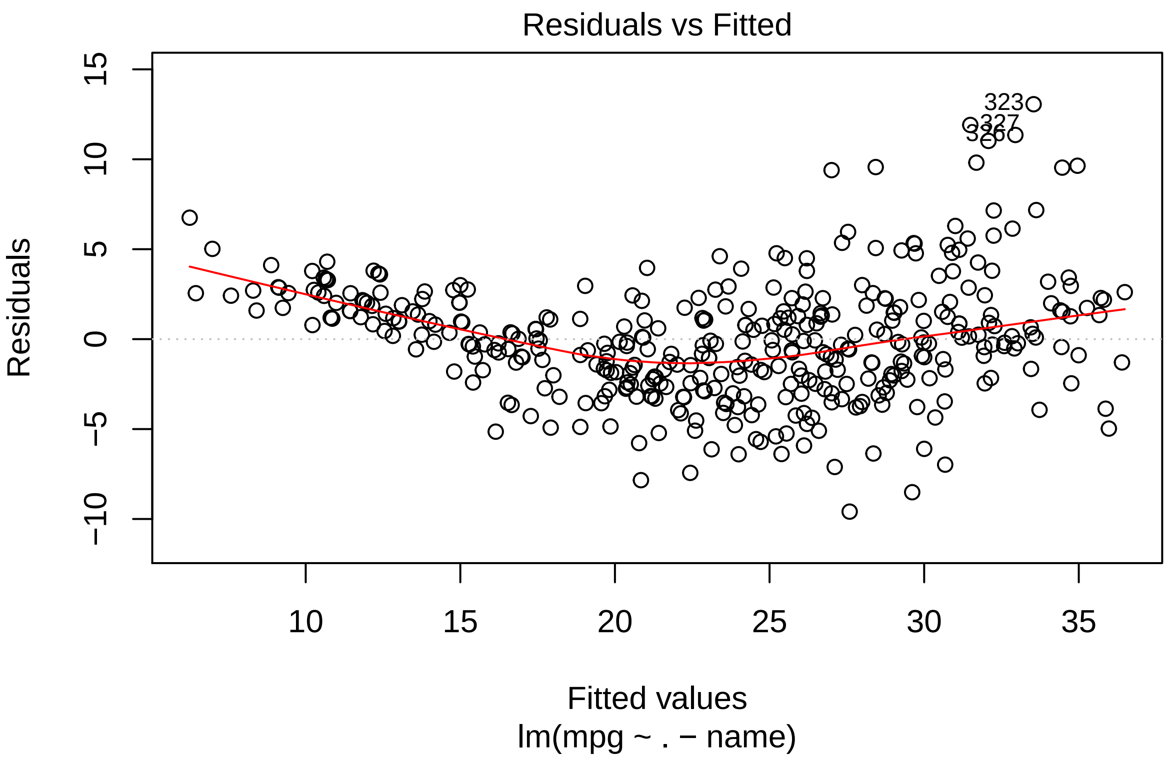
Significant relationship: year, weight, origin, displacement Weak relationship: acceleration, cylinders, horsepower

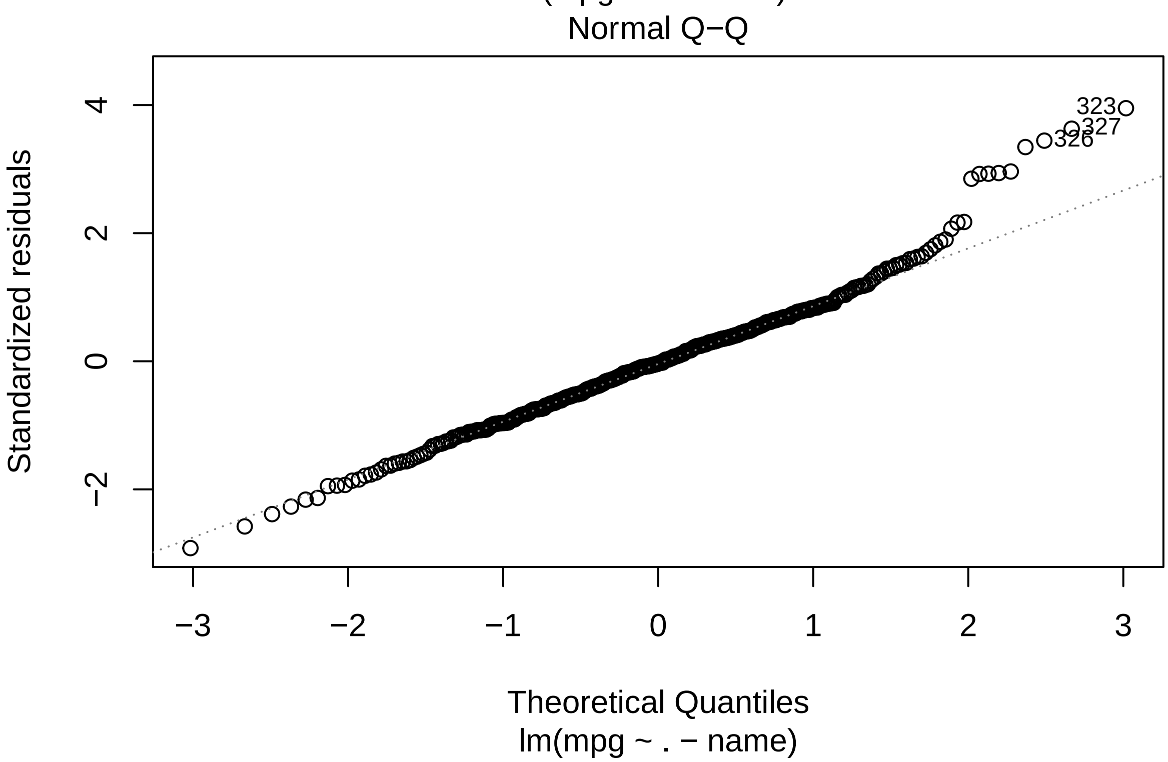
##### iii.

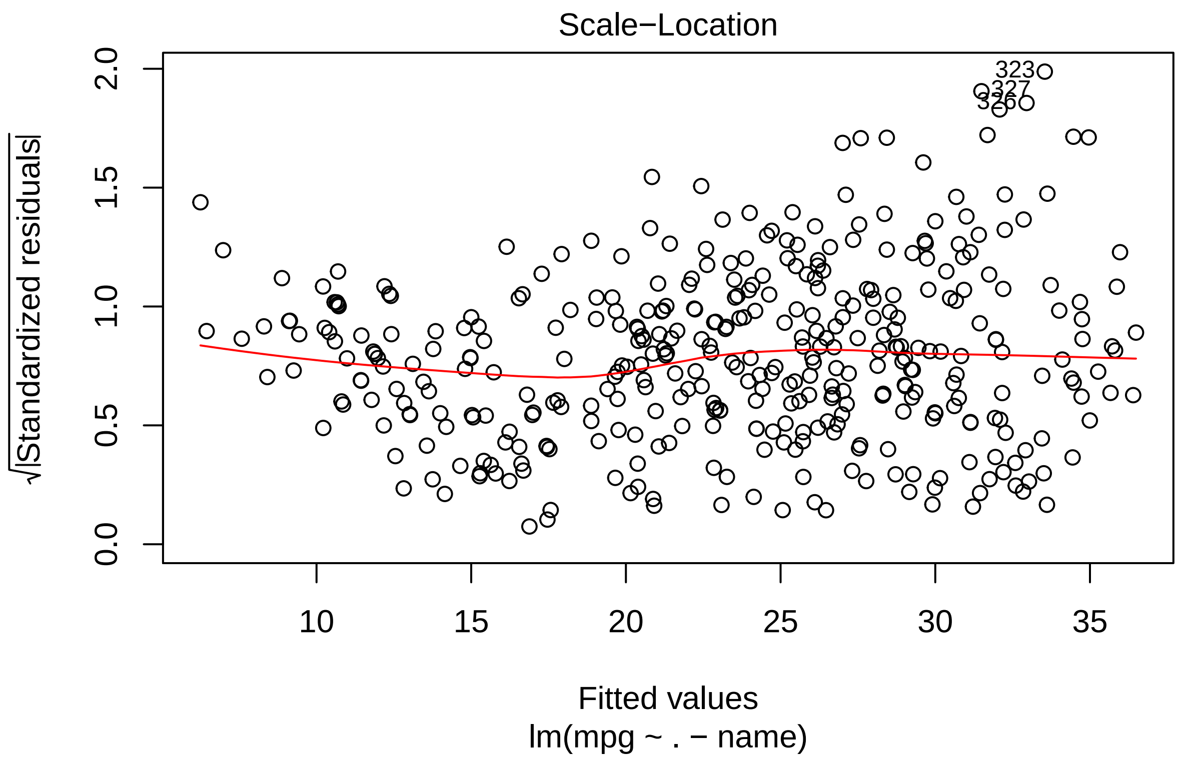
Increase of one year is an increase of 0.750773 in Y (mpg)

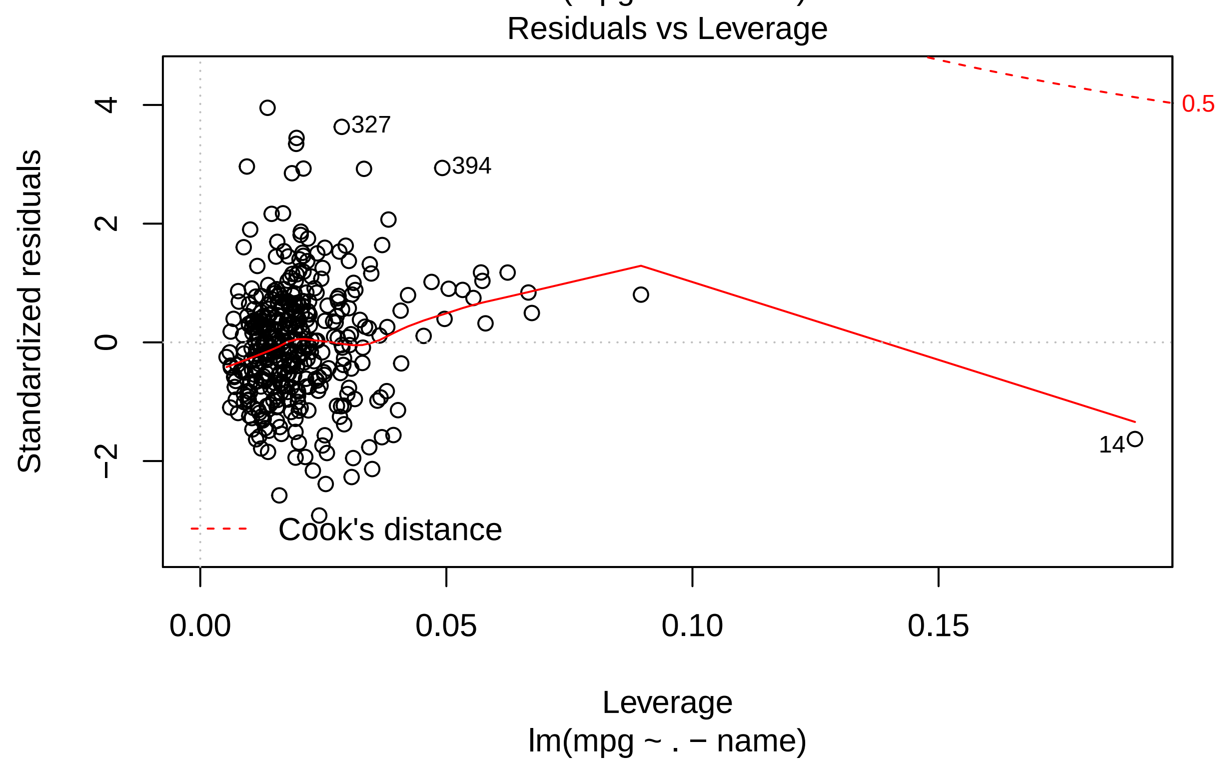
#### (d)

```r
plot(result)
```


Residuals vs Fitted
Residuals
Fitted values
lm(mpg ~ . − name)
323
327
326


Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(mpg ~ . − name)
323
327
326




## Problem 7

**Exercise 14 from section 3.7**

**(a)**

```
set.seed(1)
x1=runif(100)
x2=0.5*x1+rnorm(100)/10
y =2+2*x1+0.3*x2+rnorm(100)
```

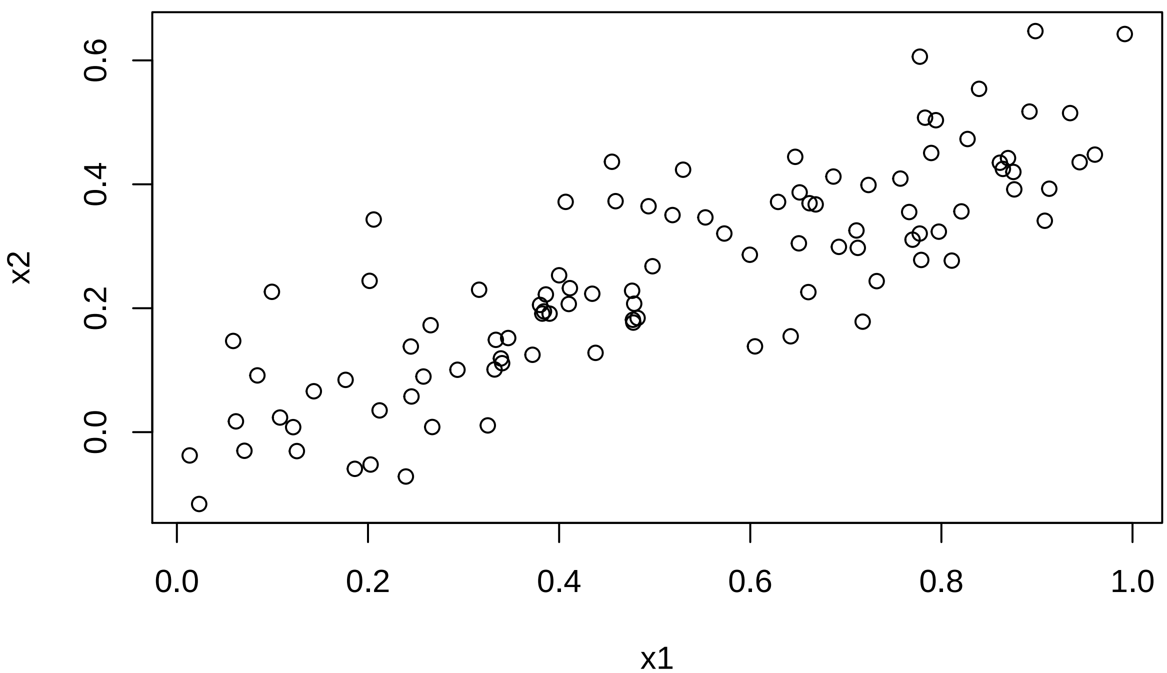
$\beta_0 = 2$
$\beta_1 = 2$
$\beta_2 = 0.3$

**(b)**

```
cor(x1,x2)
```

```
## [1] 0.8351212
```

```
plot(x1,x2)
```




Linear correlation, X1 and X2 increases together

**(c)**

```
lm.out = lm( formula = y ~ . ,data = as.data.frame(matrix(c(x1,x2),100,2, byrow = FALSE)) )
summary(lm.out)
```

```
##
## Call:
## lm(formula = y ~ ., data = as.data.frame(matrix(c(x1, x2), 100,
##     2, byrow = FALSE)))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.8311 -0.7273 -0.0537  0.6338  2.3359
```

```
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.1305     0.2319   9.188 7.61e-15 ***
## V1            1.4396     0.7212   1.996   0.0487 *
## V2            1.0097     1.1337   0.891   0.3754
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.056 on 97 degrees of freedom
## Multiple R-squared:  0.2088, Adjusted R-squared:  0.1925
## F-statistic:  12.8 on 2 and 97 DF,  p-value: 1.164e-05
```

$\hat{\beta}_0 = 2.1305$
$\hat{\beta}_1 = 1.4396$
$\hat{\beta}_2 = 1.0097$

Only $\hat{\beta}_0$ is close to $\beta_0$
For V1: We can reject $H_0$ because p-value is below 5%
For V1: We cannot reject $H_0$ because p-value is above 5%

### (d)

```r
lm.out = lm( formula = y ~ . ,data = as.data.frame(matrix(c(x1),100,1, byrow = FALSE)) )
summary(lm.out)
```

```
##
## Call:
## lm(formula = y ~ ., data = as.data.frame(matrix(c(x1), 100, 1,
##     byrow = FALSE)))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.89495 -0.66874 -0.07785  0.59221  2.45560
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.1124     0.2307   9.155 8.27e-15 ***
## V1            1.9759     0.3963   4.986 2.66e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.055 on 98 degrees of freedom
## Multiple R-squared:  0.2024, Adjusted R-squared:  0.1942
## F-statistic: 24.86 on 1 and 98 DF,  p-value: 2.661e-06
```

Using only X1 the coeffient is 1.9759. It is higher comapre to using both X1 and X2 that was 1.4396 and we can reject $H_0$ for its very low p-value.

### (e)

```r
lm.out = lm( formula = y ~ . ,data = as.data.frame(matrix(c(x2),100,1, byrow = FALSE)) )
summary(lm.out)
```

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

```
## lm(formula = y ~ ., data = as.data.frame(matrix(c(x2), 100, 1,
##     byrow = FALSE)))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.62687 -0.75156 -0.03598  0.72383  2.44890
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.3899     0.1949   12.26  < 2e-16 ***
## V1            2.8996     0.6330    4.58 1.37e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.072 on 98 degrees of freedom
## Multiple R-squared:  0.1763, Adjusted R-squared:  0.1679
## F-statistic: 20.98 on 1 and 98 DF,  p-value: 1.366e-05
```

Using only X2 the coeffient is 2.8996 It is much higher comapre to using both X1 and X2 that was 1.0097 and we can reject $H_0$ for its very low p-value.

### (f)

No, both result are not against each other. X1 and X2 have collinearity and it is hard distinguish effects of both together.

### (g)

```r
x1=c(x1,0.1)
x2=c(x2,0.8)
y=c(y,6)
lm.out1 = lm(y ~ x1 + x2 )
summary(lm.out)
```

```
##
## Call:
## lm(formula = y ~ ., data = as.data.frame(matrix(c(x2), 100, 1,
##     byrow = FALSE)))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.62687 -0.75156 -0.03598  0.72383  2.44890
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.3899     0.1949   12.26  < 2e-16 ***
## V1            2.8996     0.6330    4.58 1.37e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.072 on 98 degrees of freedom
## Multiple R-squared:  0.1763, Adjusted R-squared:  0.1679
## F-statistic: 20.98 on 1 and 98 DF,  p-value: 1.366e-05
```

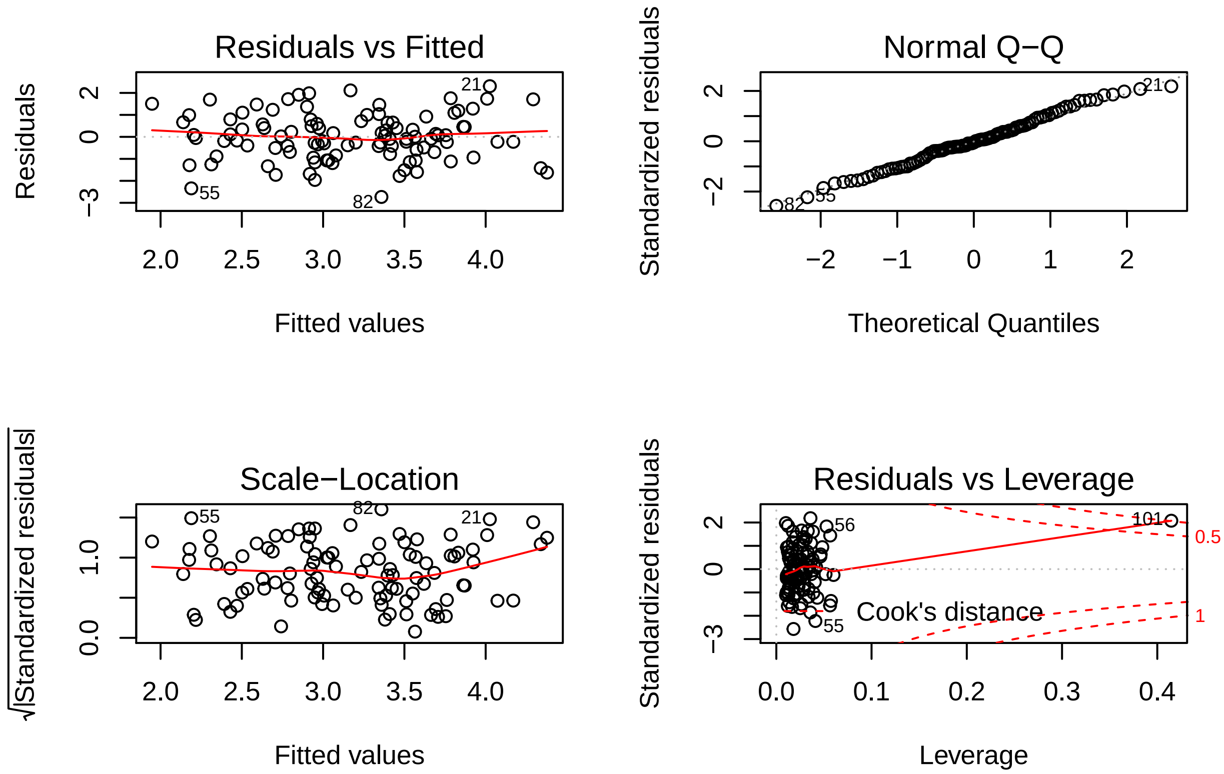
```r
lm.out2 = lm(y ~ x1)
summary(lm.out)
```

```
##
## Call:
## lm(formula = y ~ ., data = as.data.frame(matrix(c(x2), 100, 1,
##     byrow = FALSE)))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.62687 -0.75156 -0.03598  0.72383  2.44890
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.3899     0.1949   12.26  < 2e-16 ***
## V1            2.8996     0.6330    4.58 1.37e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.072 on 98 degrees of freedom
## Multiple R-squared:  0.1763, Adjusted R-squared:  0.1679
## F-statistic: 20.98 on 1 and 98 DF,  p-value: 1.366e-05
```

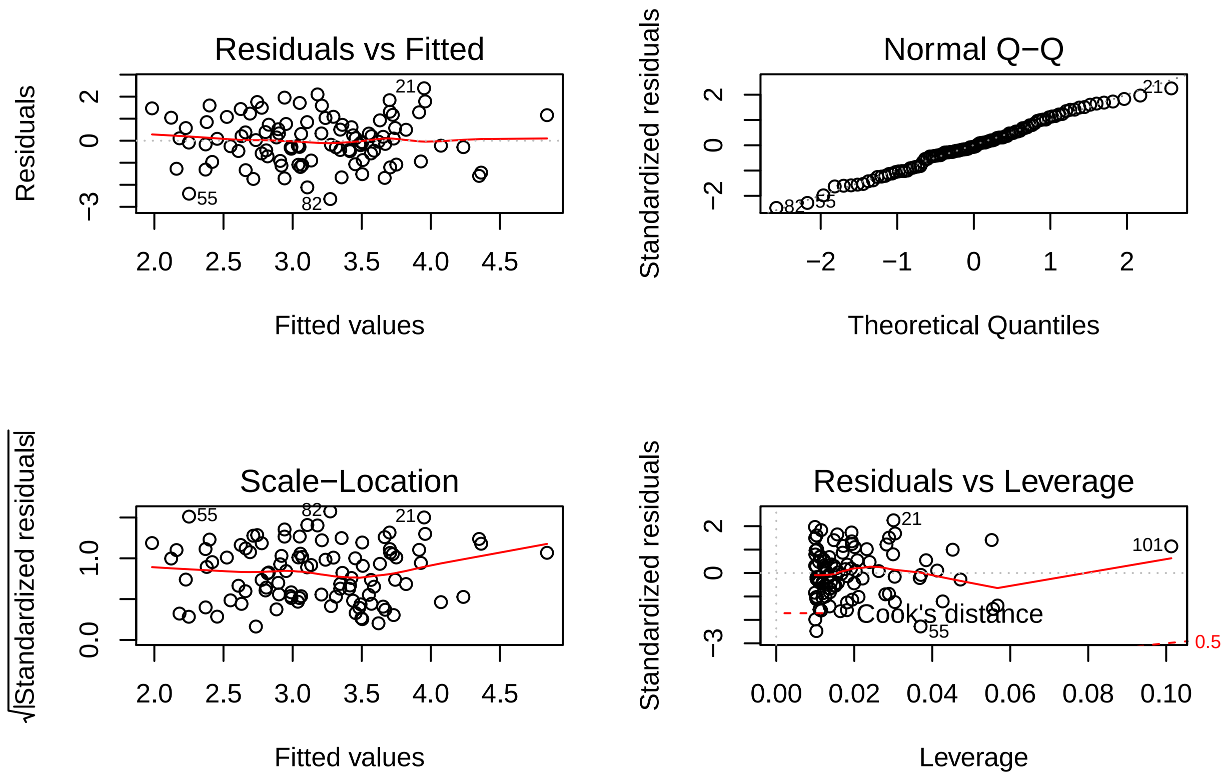
```r
lm.out3 = lm(y ~ x2)
summary(lm.out)
```

```
##
## Call:
## lm(formula = y ~ ., data = as.data.frame(matrix(c(x2), 100, 1,
##     byrow = FALSE)))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.62687 -0.75156 -0.03598  0.72383  2.44890
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.3899     0.1949   12.26  < 2e-16 ***
## V1            2.8996     0.6330    4.58 1.37e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.072 on 98 degrees of freedom
## Multiple R-squared:  0.1763, Adjusted R-squared:  0.1679
## F-statistic: 20.98 on 1 and 98 DF,  p-value: 1.366e-05
```

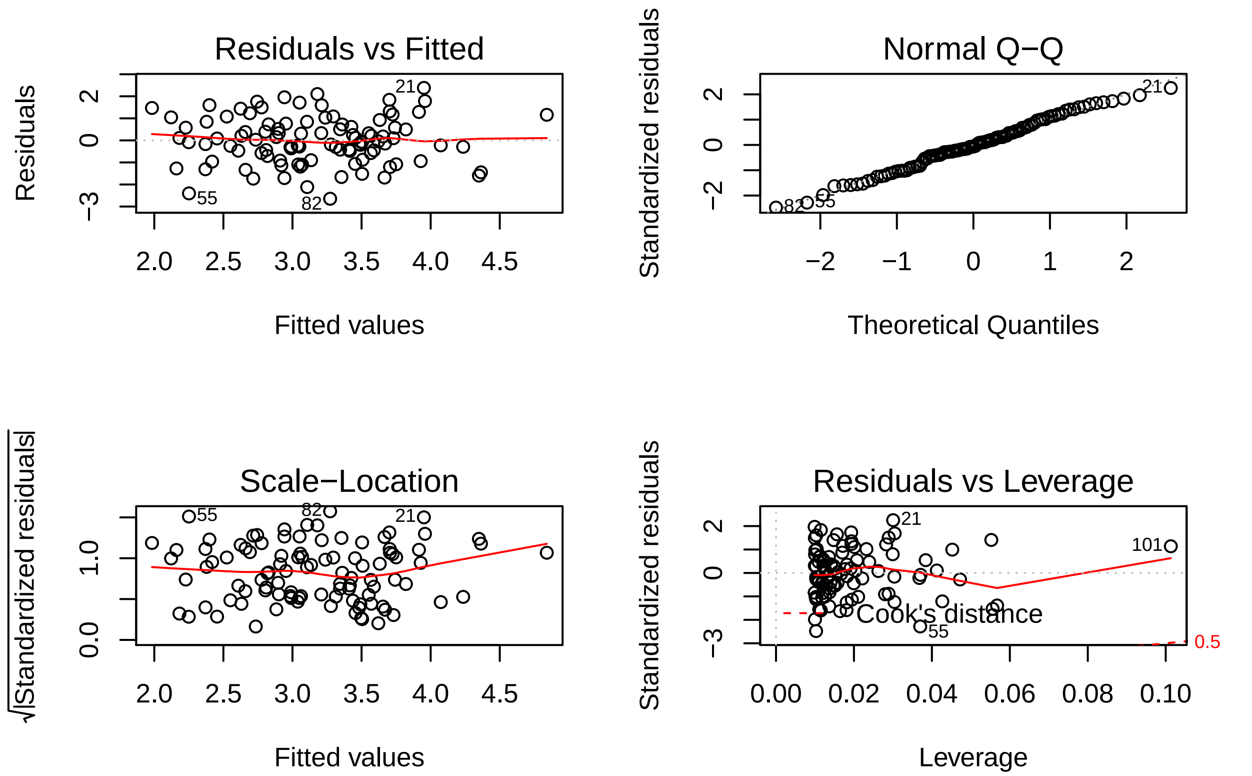
```r
par(mfrow=c(2,2))
plot(lm.out1)
```




```r
par(mfrow=c(2,2))
plot(lm.out3)
```




```r
par(mfrow=c(2,2))
plot(lm.out3)
```




# Problem 8

### Exercise 15 from section 3.7

```
install.packages("MASS", repos = "http://cran.us.r-project.org")
```

```
##
## The downloaded binary packages are in
##  /var/folders/nm/g09p009s0qb231f3_hsjz7r40011sl/T//Rtmpo9Gk2J/downloaded_packages
```

```
library(MASS)
boston = MASS::Boston
```

### (a)

```
lm.zn = lm(crim ~ zn, data = boston)
summary(lm.zn)
```

```
##
## Call:
## lm(formula = crim ~ zn, data = boston)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -4.429 -4.222 -2.620  1.250 84.523
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)  4.45369    0.41722  10.675  < 2e-16 ***
## zn          -0.07393    0.01609  -4.594 5.51e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.435 on 504 degrees of freedom
## Multiple R-squared:  0.04019,    Adjusted R-squared:  0.03828
## F-statistic:  21.1 on 1 and 504 DF,  p-value: 5.506e-06
```

```
lm.indus = lm(crim ~ indus, data = boston)
summary(lm.indus)
```

```
##
## Call:
## lm(formula = crim ~ indus, data = boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -11.972  -2.698  -0.736   0.712  81.813
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.06374    0.66723  -3.093  0.00209 **
## indus        0.50978    0.05102   9.991  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.866 on 504 degrees of freedom
## Multiple R-squared:  0.1653, Adjusted R-squared:  0.1637
## F-statistic: 99.82 on 1 and 504 DF,  p-value: < 2.2e-16
```

```
lm.chas = lm(crim ~ chas, data = boston)
summary(lm.chas)
```

```
##
## Call:
## lm(formula = crim ~ chas, data = boston)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -3.738 -3.661 -3.435  0.018 85.232
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.7444     0.3961   9.453   <2e-16 ***
## chas         -1.8928     1.5061  -1.257    0.209
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.597 on 504 degrees of freedom
## Multiple R-squared:  0.003124,   Adjusted R-squared:  0.001146
## F-statistic: 1.579 on 1 and 504 DF,  p-value: 0.2094
```

```
lm.nox = lm(crim ~ nox, data = boston)
summary(lm.nox)
```

```
##
## Call:
## lm(formula = crim ~ nox, data = boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -12.371  -2.738  -0.974   0.559  81.728
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -13.720      1.699  -8.073 5.08e-15 ***
## nox           31.249      2.999  10.419  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.81 on 504 degrees of freedom
## Multiple R-squared:  0.1772, Adjusted R-squared:  0.1756
## F-statistic: 108.6 on 1 and 504 DF,  p-value: < 2.2e-16
```

```r
lm.rm = lm(crim ~ rm, data = boston)
summary(lm.rm)
```

```
##
## Call:
## lm(formula = crim ~ rm, data = boston)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -6.604 -3.952 -2.654  0.989 87.197
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   20.482      3.365   6.088 2.27e-09 ***
## rm            -2.684      0.532  -5.045 6.35e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.401 on 504 degrees of freedom
## Multiple R-squared:  0.04807,    Adjusted R-squared:  0.04618
## F-statistic: 25.45 on 1 and 504 DF,  p-value: 6.347e-07
```

```r
lm.age = lm(crim ~ age, data = boston)
summary(lm.age)
```

```
##
## Call:
## lm(formula = crim ~ age, data = boston)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -6.789 -4.257 -1.230  1.527 82.849
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.77791    0.94398  -4.002 7.22e-05 ***
```

```
## age          0.10779    0.01274   8.463 2.85e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.057 on 504 degrees of freedom
## Multiple R-squared:  0.1244, Adjusted R-squared:  0.1227
## F-statistic: 71.62 on 1 and 504 DF,  p-value: 2.855e-16
```

```r
lm.dis = lm(crim ~ dis, data = boston)
summary(lm.dis)
```

```
##
## Call:
## lm(formula = crim ~ dis, data = boston)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -6.708 -4.134 -1.527  1.516 81.674
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   9.4993     0.7304  13.006   <2e-16 ***
## dis          -1.5509     0.1683  -9.213   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.965 on 504 degrees of freedom
## Multiple R-squared:  0.1441, Adjusted R-squared:  0.1425
## F-statistic: 84.89 on 1 and 504 DF,  p-value: < 2.2e-16
```

```r
lm.rad = lm(crim ~ rad, data = boston)
summary(lm.rad)
```

```
##
## Call:
## lm(formula = crim ~ rad, data = boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -10.164  -1.381  -0.141   0.660  76.433
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.28716    0.44348  -5.157 3.61e-07 ***
## rad          0.61791    0.03433  17.998  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.718 on 504 degrees of freedom
## Multiple R-squared:  0.3913, Adjusted R-squared:   0.39
## F-statistic: 323.9 on 1 and 504 DF,  p-value: < 2.2e-16
```

```r
lm.tax = lm(crim ~ tax, data = boston)
summary(lm.tax)
```

```
##
```

```
## Call:
## lm(formula = crim ~ tax, data = boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -12.513  -2.738  -0.194   1.065  77.696
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -8.528369   0.815809  -10.45   <2e-16 ***
## tax          0.029742   0.001847   16.10   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.997 on 504 degrees of freedom
## Multiple R-squared:  0.3396, Adjusted R-squared:  0.3383
## F-statistic: 259.2 on 1 and 504 DF,  p-value: < 2.2e-16
```

```
lm.ptratio = lm(crim ~ ptratio, data = boston)
summary(lm.ptratio)
```

```
##
## Call:
## lm(formula = crim ~ ptratio, data = boston)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -7.654 -3.985 -1.912  1.825 83.353
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -17.6469     3.1473  -5.607 3.40e-08 ***
## ptratio       1.1520     0.1694   6.801 2.94e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.24 on 504 degrees of freedom
## Multiple R-squared:  0.08407,    Adjusted R-squared:  0.08225
## F-statistic: 46.26 on 1 and 504 DF,  p-value: 2.943e-11
```

```
lm.black = lm(crim ~ black, data = boston)
summary(lm.black)
```

```
##
## Call:
## lm(formula = crim ~ black, data = boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -13.756  -2.299  -2.095  -1.296  86.822
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 16.553529   1.425903  11.609   <2e-16 ***
## black       -0.036280   0.003873  -9.367   <2e-16 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.946 on 504 degrees of freedom
## Multiple R-squared:  0.1483, Adjusted R-squared:  0.1466
## F-statistic: 87.74 on 1 and 504 DF,  p-value: < 2.2e-16
```

```r
lm.lstat = lm(crim ~ lstat, data = boston)
summary(lm.lstat)
```

```
##
## Call:
## lm(formula = crim ~ lstat, data = boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -13.925  -2.822  -0.664   1.079  82.862
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.33054    0.69376  -4.801 2.09e-06 ***
## lstat        0.54880    0.04776  11.491  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.664 on 504 degrees of freedom
## Multiple R-squared:  0.2076, Adjusted R-squared:  0.206
## F-statistic:   132 on 1 and 504 DF,  p-value: < 2.2e-16
```

```r
lm.medv = lm(crim ~ medv, data = boston)
summary(lm.medv)
```

```
##
## Call:
## lm(formula = crim ~ medv, data = boston)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -9.071 -4.022 -2.343  1.298 80.957
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 11.79654    0.93419   12.63   <2e-16 ***
## medv        -0.36316    0.03839   -9.46   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.934 on 504 degrees of freedom
## Multiple R-squared:  0.1508, Adjusted R-squared:  0.1491
## F-statistic: 89.49 on 1 and 504 DF,  p-value: < 2.2e-16
```

All have p-value less than 0.05 except *"chas"* with p-value=0.2094 so there is significant association between all predictors and the response except for *"chas"*.

### (b)

```r
lm.out = lm(crim ~ ., data = boston)
summary(lm.out)
```

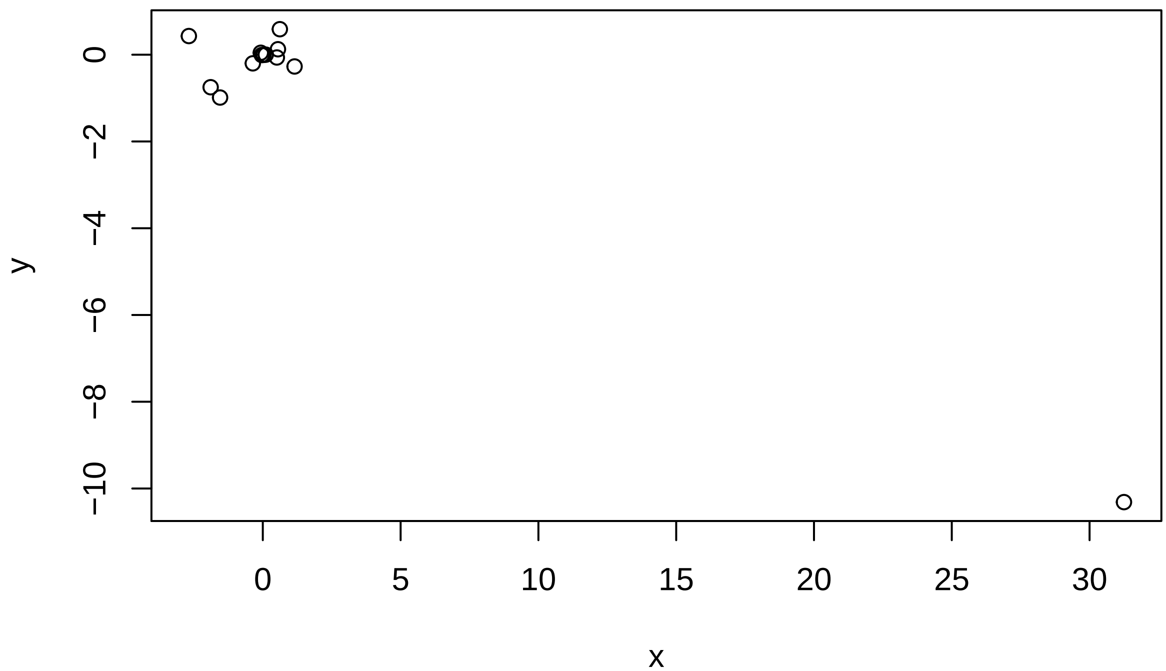
```
##
## Call:
## lm(formula = crim ~ ., data = boston)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -9.924 -2.120 -0.353  1.019 75.051
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  17.033228   7.234903   2.354 0.018949 *
## zn            0.044855   0.018734   2.394 0.017025 *
## indus        -0.063855   0.083407  -0.766 0.444294
## chas         -0.749134   1.180147  -0.635 0.525867
## nox         -10.313535   5.275536  -1.955 0.051152 .
## rm            0.430131   0.612830   0.702 0.483089
## age           0.001452   0.017925   0.081 0.935488
## dis          -0.987176   0.281817  -3.503 0.000502 ***
## rad           0.588209   0.088049   6.680 6.46e-11 ***
## tax          -0.003780   0.005156  -0.733 0.463793
## ptratio      -0.271081   0.186450  -1.454 0.146611
## black        -0.007538   0.003673  -2.052 0.040702 *
## lstat         0.126211   0.075725   1.667 0.096208 .
## medv         -0.198887   0.060516  -3.287 0.001087 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.439 on 492 degrees of freedom
## Multiple R-squared:  0.454,  Adjusted R-squared:  0.4396
## F-statistic: 31.47 on 13 and 492 DF,  p-value: < 2.2e-16
```

We cab reject $H_0$ for zn, dis, rad, black, medv.

### (c)

```r
x = c(coefficients(lm.zn)[2],
      coefficients(lm.indus)[2],
      coefficients(lm.chas)[2],
      coefficients(lm.nox)[2],
      coefficients(lm.rm)[2],
      coefficients(lm.age)[2],
      coefficients(lm.dis)[2],
      coefficients(lm.rad)[2],
      coefficients(lm.tax)[2],
      coefficients(lm.ptratio)[2],
      coefficients(lm.black)[2],
      coefficients(lm.lstat)[2],
      coefficients(lm.medv)[2])
y = coefficients(lm.out)[2:14]
```

```r
plot(x, y)
```




**(d)**

```r
lm.ployzn = lm(crim ~ poly(zn,3), data=boston)
summary(lm.ployzn)
```

```
##
## Call:
## lm(formula = crim ~ poly(zn, 3), data = boston)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -4.821 -4.614 -1.294  0.473 84.130
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)      3.6135     0.3722   9.709  < 2e-16 ***
## poly(zn, 3)1 -38.7498     8.3722  -4.628  4.7e-06 ***
## poly(zn, 3)2  23.9398     8.3722   2.859  0.00442 **
## poly(zn, 3)3 -10.0719     8.3722  -1.203  0.22954
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.372 on 502 degrees of freedom
## Multiple R-squared:  0.05824,    Adjusted R-squared:  0.05261
## F-statistic: 10.35 on 3 and 502 DF,  p-value: 1.281e-06
```

```r
lm.ployindus = lm(crim ~ poly(indus,3), data=boston)
summary(lm.ployindus)
```

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

```
## lm(formula = crim ~ poly(indus, 3), data = boston)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -8.278 -2.514  0.054  0.764 79.713
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)        3.614      0.330  10.950  < 2e-16 ***
## poly(indus, 3)1   78.591      7.423  10.587  < 2e-16 ***
## poly(indus, 3)2  -24.395      7.423  -3.286  0.00109 **
## poly(indus, 3)3  -54.130      7.423  -7.292  1.2e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.423 on 502 degrees of freedom
## Multiple R-squared:  0.2597, Adjusted R-squared:  0.2552
## F-statistic: 58.69 on 3 and 502 DF,  p-value: < 2.2e-16
```

```r
lm.ploynox = lm(crim ~ poly(nox,3), data=boston)
summary(lm.ploynox)
```

```
##
## Call:
## lm(formula = crim ~ poly(nox, 3), data = boston)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -9.110 -2.068 -0.255  0.739 78.302
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)     3.6135     0.3216  11.237  < 2e-16 ***
## poly(nox, 3)1  81.3720     7.2336  11.249  < 2e-16 ***
## poly(nox, 3)2 -28.8286     7.2336  -3.985 7.74e-05 ***
## poly(nox, 3)3 -60.3619     7.2336  -8.345 6.96e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.234 on 502 degrees of freedom
## Multiple R-squared:  0.297,  Adjusted R-squared:  0.2928
## F-statistic: 70.69 on 3 and 502 DF,  p-value: < 2.2e-16
```

```r
lm.ployrm = lm(crim ~ poly(rm,3), data=boston)
summary(lm.ployrm)
```

```
##
## Call:
## lm(formula = crim ~ poly(rm, 3), data = boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -18.485  -3.468  -2.221  -0.015  87.219
##
## Coefficients:
```

```
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    3.6135     0.3703   9.758  < 2e-16 ***
## poly(rm, 3)1 -42.3794     8.3297  -5.088 5.13e-07 ***
## poly(rm, 3)2  26.5768     8.3297   3.191  0.00151 **
## poly(rm, 3)3  -5.5103     8.3297  -0.662  0.50858
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.33 on 502 degrees of freedom
## Multiple R-squared:  0.06779,    Adjusted R-squared:  0.06222
## F-statistic: 12.17 on 3 and 502 DF,  p-value: 1.067e-07
```

```r
lm.ployage = lm(crim ~ poly(age,3), data=boston)
summary(lm.ployage)
```

```
##
## Call:
## lm(formula = crim ~ poly(age, 3), data = boston)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -9.762 -2.673 -0.516  0.019 82.842
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)     3.6135     0.3485  10.368  < 2e-16 ***
## poly(age, 3)1  68.1820     7.8397   8.697  < 2e-16 ***
## poly(age, 3)2  37.4845     7.8397   4.781 2.29e-06 ***
## poly(age, 3)3  21.3532     7.8397   2.724  0.00668 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.84 on 502 degrees of freedom
## Multiple R-squared:  0.1742, Adjusted R-squared:  0.1693
## F-statistic: 35.31 on 3 and 502 DF,  p-value: < 2.2e-16
```

```r
lm.ploydis = lm(crim ~ poly(dis,3), data=boston)
summary(lm.ploydis)
```

```
##
## Call:
## lm(formula = crim ~ poly(dis, 3), data = boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -10.757  -2.588   0.031   1.267  76.378
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)     3.6135     0.3259  11.087  < 2e-16 ***
## poly(dis, 3)1 -73.3886     7.3315 -10.010  < 2e-16 ***
## poly(dis, 3)2  56.3730     7.3315   7.689 7.87e-14 ***
## poly(dis, 3)3 -42.6219     7.3315  -5.814 1.09e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 7.331 on 502 degrees of freedom
## Multiple R-squared:  0.2778, Adjusted R-squared:  0.2735
## F-statistic: 64.37 on 3 and 502 DF,  p-value: < 2.2e-16
```

```
lm.ployrad = lm(crim ~ poly(rad,3), data=boston)
summary(lm.ployrad)
```

```
##
## Call:
## lm(formula = crim ~ poly(rad, 3), data = boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -10.381  -0.412  -0.269   0.179  76.217
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)     3.6135     0.2971  12.164  < 2e-16 ***
## poly(rad, 3)1 120.9074     6.6824  18.093  < 2e-16 ***
## poly(rad, 3)2  17.4923     6.6824   2.618  0.00912 **
## poly(rad, 3)3   4.6985     6.6824   0.703  0.48231
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.682 on 502 degrees of freedom
## Multiple R-squared:    0.4,  Adjusted R-squared:  0.3965
## F-statistic: 111.6 on 3 and 502 DF,  p-value: < 2.2e-16
```

```
lm.ployptratio = lm(crim ~ poly(ptratio,3), data=boston)
summary(lm.ployptratio)
```

```
##
## Call:
## lm(formula = crim ~ poly(ptratio, 3), data = boston)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -6.833 -4.146 -1.655  1.408 82.697
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)          3.614      0.361  10.008  < 2e-16 ***
## poly(ptratio, 3)1   56.045      8.122   6.901 1.57e-11 ***
## poly(ptratio, 3)2   24.775      8.122   3.050  0.00241 **
## poly(ptratio, 3)3  -22.280      8.122  -2.743  0.00630 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.122 on 502 degrees of freedom
## Multiple R-squared:  0.1138, Adjusted R-squared:  0.1085
## F-statistic: 21.48 on 3 and 502 DF,  p-value: 4.171e-13
```

```
lm.ployblack = lm(crim ~ poly(black,3), data=boston)
summary(lm.ployblack)
```

```
##
## Call:
## lm(formula = crim ~ poly(black, 3), data = boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -13.096  -2.343  -2.128  -1.439  86.790
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)       3.6135     0.3536  10.218   <2e-16 ***
## poly(black, 3)1 -74.4312     7.9546  -9.357   <2e-16 ***
## poly(black, 3)2   5.9264     7.9546   0.745    0.457
## poly(black, 3)3  -4.8346     7.9546  -0.608    0.544
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.955 on 502 degrees of freedom
## Multiple R-squared:  0.1498, Adjusted R-squared:  0.1448
## F-statistic: 29.49 on 3 and 502 DF,  p-value: < 2.2e-16
```

```r
lm.ploylstat = lm(crim ~ poly(lstat,3), data=boston)
summary(lm.ploylstat)
```

```
##
## Call:
## lm(formula = crim ~ poly(lstat, 3), data = boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.234  -2.151  -0.486   0.066  83.353
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)       3.6135     0.3392  10.654   <2e-16 ***
## poly(lstat, 3)1  88.0697     7.6294  11.543   <2e-16 ***
## poly(lstat, 3)2  15.8882     7.6294   2.082   0.0378 *
## poly(lstat, 3)3 -11.5740     7.6294  -1.517   0.1299
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.629 on 502 degrees of freedom
## Multiple R-squared:  0.2179, Adjusted R-squared:  0.2133
## F-statistic: 46.63 on 3 and 502 DF,  p-value: < 2.2e-16
```

```r
lm.ploymedv = lm(crim ~ poly(medv,3), data=boston)
summary(lm.ploymedv)
```

```
##
## Call:
## lm(formula = crim ~ poly(medv, 3), data = boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -24.427  -1.976  -0.437   0.439  73.655
```

```
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)       3.614      0.292  12.374  < 2e-16 ***
## poly(medv, 3)1  -75.058      6.569 -11.426  < 2e-16 ***
## poly(medv, 3)2   88.086      6.569  13.409  < 2e-16 ***
## poly(medv, 3)3  -48.033      6.569  -7.312 1.05e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.569 on 502 degrees of freedom
## Multiple R-squared:  0.4202, Adjusted R-squared:  0.4167
## F-statistic: 121.3 on 3 and 502 DF,  p-value: < 2.2e-16
```

For zn, rm, rad, tax, lstat based on p-values the cubic coefficient is not good enough.