

471

Jiajian Huang

1/17/2019

HW 1

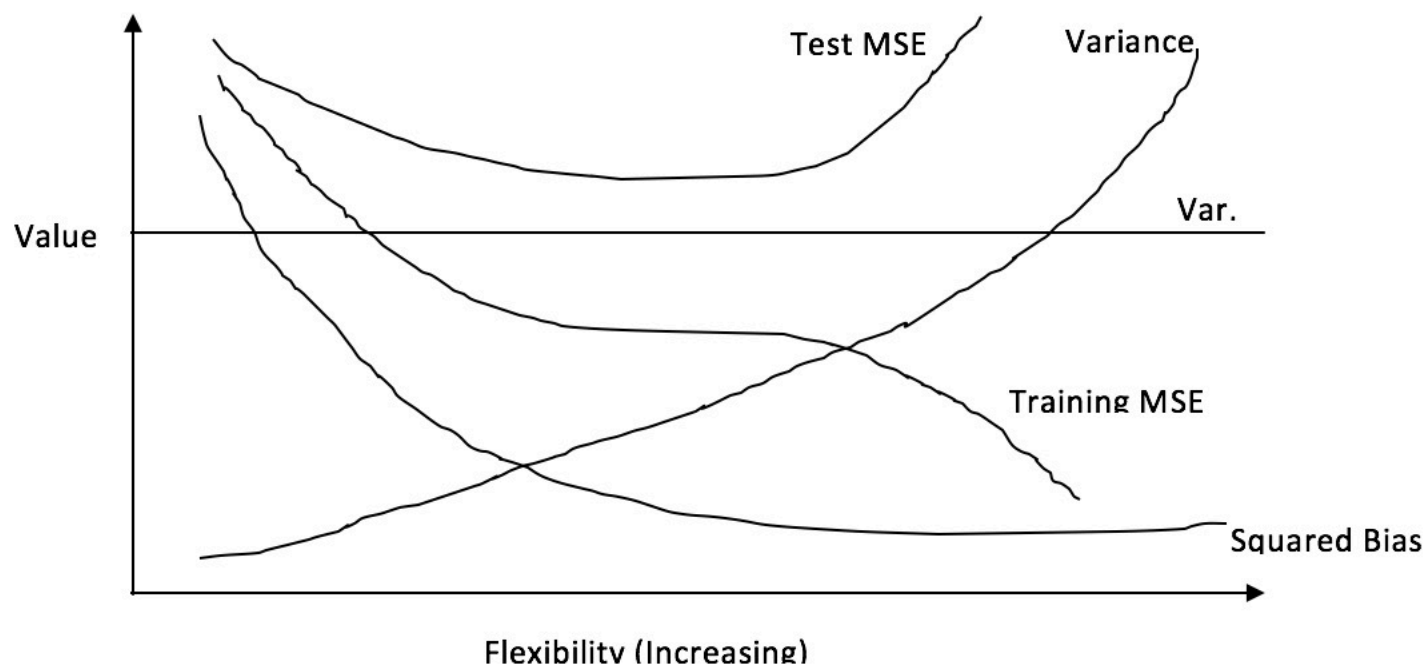
Q1

- a. The given experiment has extremely large sample size of n , and the number of predictors p is very small, so the flexible statistical learning can be used, since the large number of parameters that are present in the model could be estimated, due to large number of the sample size.
- b. The given experiment has small sample size, that is small n , and large number of parameters p , hence the flexible statistical method can't be used, as there already are a large number of parameters, and now more parameters will include a large error in the model, since the sample size is small, and all can't be obtained.
- c. When the relationship between the predictors and the response is highly non-linear, then it is better to use the flexible statistical learning, as it will fit a curve for the given model, which maybe better able to picture the model, since the relationship is not linear.
- d. If the variance of the error terms is extremely high, then if the flexible method is used, due to the risk of over-estimation, the error may increase all the more, due to addition of noise. Hence it is better to avoid the use of flexible statistical learning in all those cases.

Q2

a.

```
knitr::include_graphics("/Users/huangjiajian/Desktop/471/WechatIMG1.jpeg")
```



- b. The squared bias for the given experiment, keeps on decreasing monotonically along with the increase in the flexibility of the experiment, since more the flexible is the experiment, less will be the bias, since the model is better able to describe the relationship.

The variance keeps on increasing along with increasing in the flexibility, since with the inclusion of more variables, the variance of the errors will increase, since the risk of over-estimation increase all the more.

The model gets a much better fit than the previous situation, but only up to an optimal level, where the model obtained is the best one to depict the nature of the response, and the variable, after which over-estimation leads to larger error variance, and deviation from the original model. Thus, the test MSE decrease, reaches the minimum and then increases, after attaining the optimum position.

Q3

a.

```
write.table(College, file = "College.csv", row.names=F, sep = ",")
```

```
college <- read.csv("college.csv")
```

b.

```
head(college[,1:4])
```

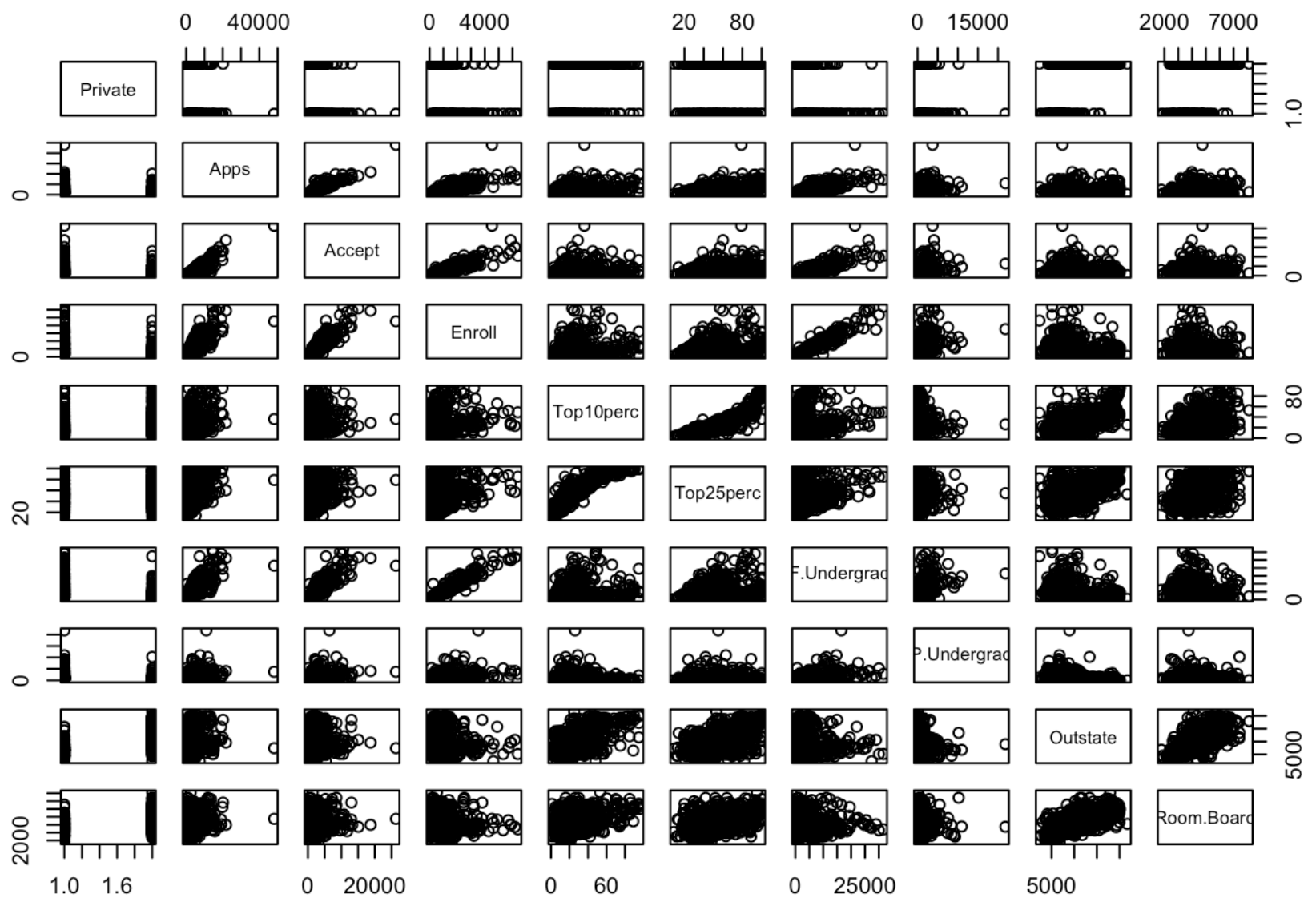
```
## Private Apps Accept Enroll
## 1 Yes 1660 1232 721
## 2 Yes 2186 1924 512
## 3 Yes 1428 1097 336
## 4 Yes 417 349 137
## 5 Yes 193 146 55
## 6 Yes 587 479 158
```

C.

```
summary(college)
```

```
## Private           Apps           Accept           Enroll           Top10perc
## No :212   Min.      :   81   Min.      :   72   Min.      :   35   Min.      : 1.00
## Yes:565   1st Qu.:  776   1st Qu.:  604   1st Qu.:  242   1st Qu.:15.00
##           Median : 1558   Median : 1110   Median :  434   Median :23.00
##           Mean    : 3002   Mean    : 2019   Mean    :  780   Mean    :27.56
##           3rd Qu.: 3624   3rd Qu.: 2424   3rd Qu.:  902   3rd Qu.:35.00
##           Max.    :48094   Max.    :26330   Max.    :6392   Max.    :96.00
## Top25perc       F.Undergrad       P.Undergrad           Outstate
## Min.      :  9.0   Min.      :  139   Min.      :    1.0   Min.      : 2340
## 1st Qu.: 41.0   1st Qu.:  992   1st Qu.:   95.0   1st Qu.: 7320
## Median : 54.0   Median : 1707   Median :  353.0   Median : 9990
## Mean     : 55.8   Mean     : 3700   Mean     :  855.3   Mean     :10441
## 3rd Qu.: 69.0   3rd Qu.: 4005   3rd Qu.:  967.0   3rd Qu.:12925
## Max.     :100.0   Max.     :31643   Max.     :21836.0   Max.     :21700
## Room.Board       Books           Personal           PhD
## Min.      :1780   Min.      :  96.0   Min.      :  250   Min.      :  8.00
## 1st Qu.:3597   1st Qu.: 470.0   1st Qu.:  850   1st Qu.: 62.00
## Median :4200   Median : 500.0   Median :1200   Median : 75.00
## Mean     :4358   Mean     : 549.4   Mean     :1341   Mean     : 72.66
## 3rd Qu.:5050   3rd Qu.: 600.0   3rd Qu.:1700   3rd Qu.: 85.00
## Max.     :8124   Max.     :2340.0   Max.     :6800   Max.     :103.00
## Terminal         S.F.Ratio       perc.alumni       Expend
## Min.      : 24.0   Min.      :  2.50   Min.      :  0.00   Min.      : 3186
## 1st Qu.: 71.0   1st Qu.:11.50   1st Qu.:13.00   1st Qu.: 6751
## Median : 82.0   Median :13.60   Median :21.00   Median : 8377
## Mean     : 79.7   Mean     :14.09   Mean     :22.74   Mean     : 9660
## 3rd Qu.: 92.0   3rd Qu.:16.50   3rd Qu.:31.00   3rd Qu.:10830
## Max.     :100.0   Max.     :39.80   Max.     :64.00   Max.     :56233
## Grad.Rate
## Min.      : 10.00
## 1st Qu.: 53.00
## Median : 65.00
## Mean     : 65.46
## 3rd Qu.: 78.00
## Max.     :118.00
```

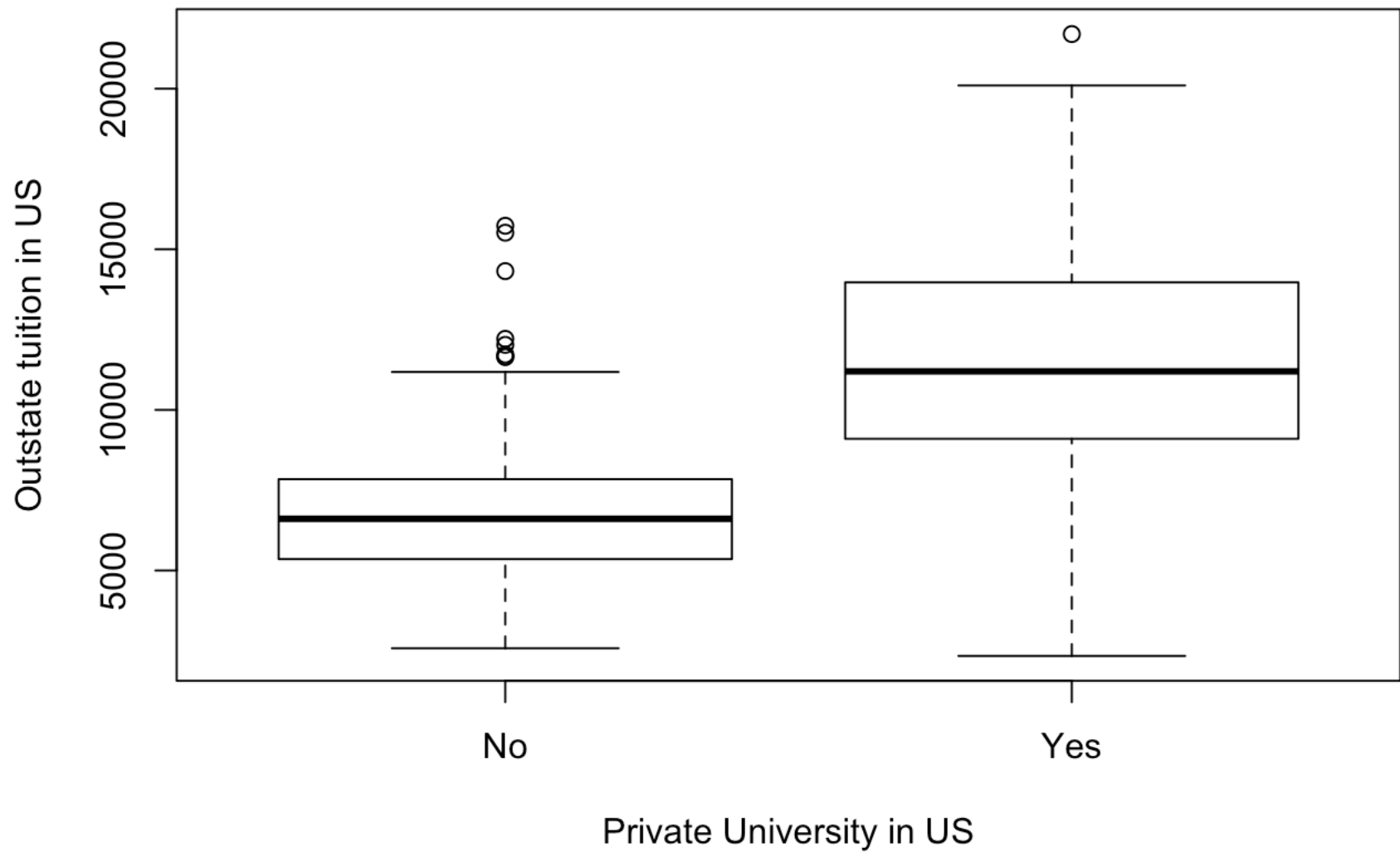
```
pairs(college[,1:10])
```



Now the scatter plot for each of the variables, in the same window is obtained by using the below given code, and the result thus obtained is also given above.

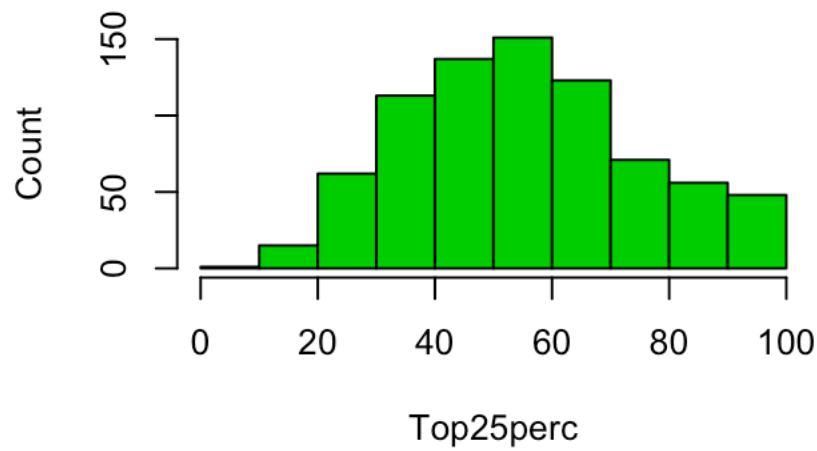
```
plot(college$Private,college$Outstate, xlab = "Private University in US", ylab = "Out
state tuition in US", main = "Outstate Plot")
```

Outstate Plot

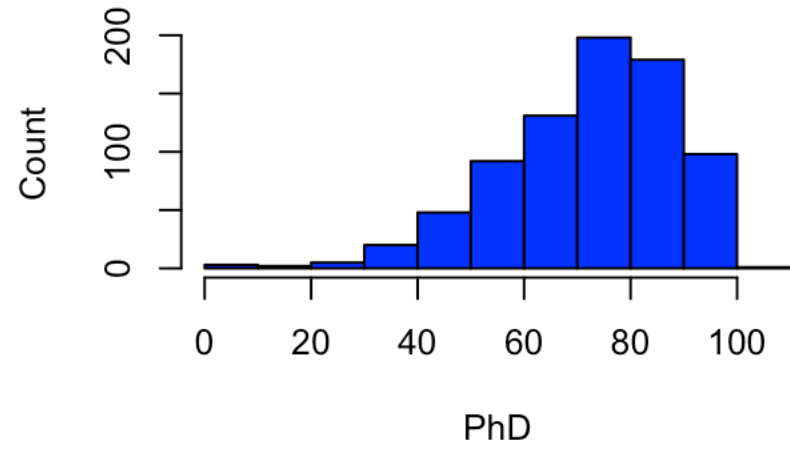


```
par(mfrow = c(2,2))
hist(college$Top25perc, col = 3, xlab = "Top25perc", ylab = "Count")
hist(college$PhD, col = 4, xlab = "PhD", ylab = "Count")
hist(college$Grad.Rate, col = 5, xlab = "Grade rate", ylab = "Count")
hist(college$Expend, col = 2, xlab = "Expend", ylab = "Count")
```

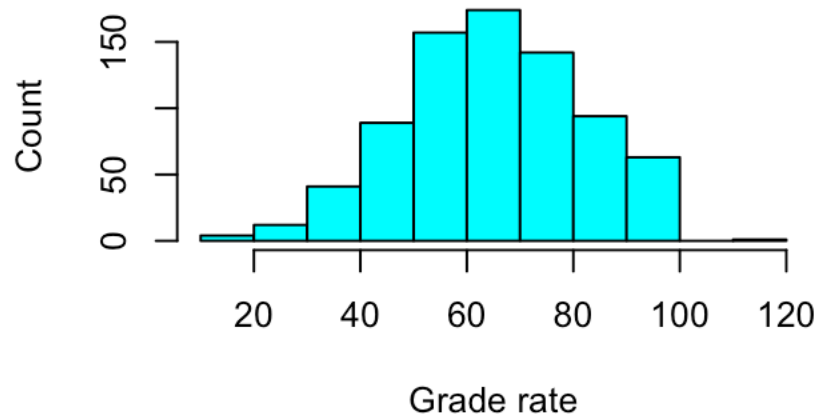
Histogram of college\$Top25perc



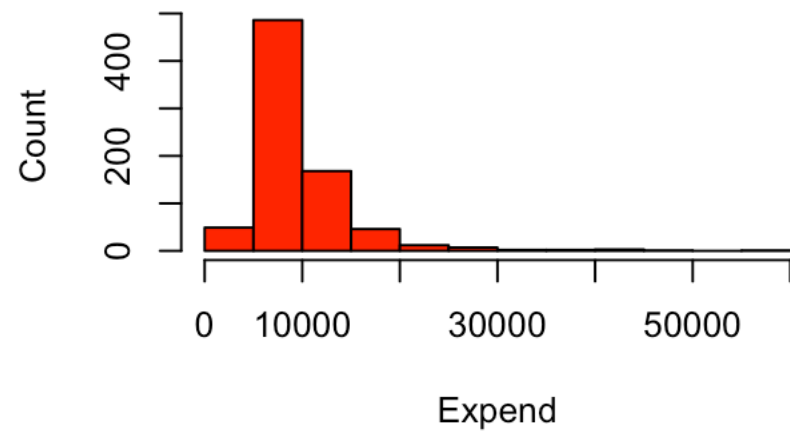
Histogram of college\$PhD



Histogram of college\$Grad.Rate



Histogram of college\$Expend



```
summary(college$Top25perc)
```

##	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
##	9.0	41.0	54.0	55.8	69.0	100.0

```
summary(college$Grad.Rate)
```

##	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
##	10.00	53.00	65.00	65.46	78.00	118.00

```
summary(college$PhD)
```

##	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
##	8.00	62.00	75.00	72.66	85.00	103.00

```
summary(college$Expend)
```

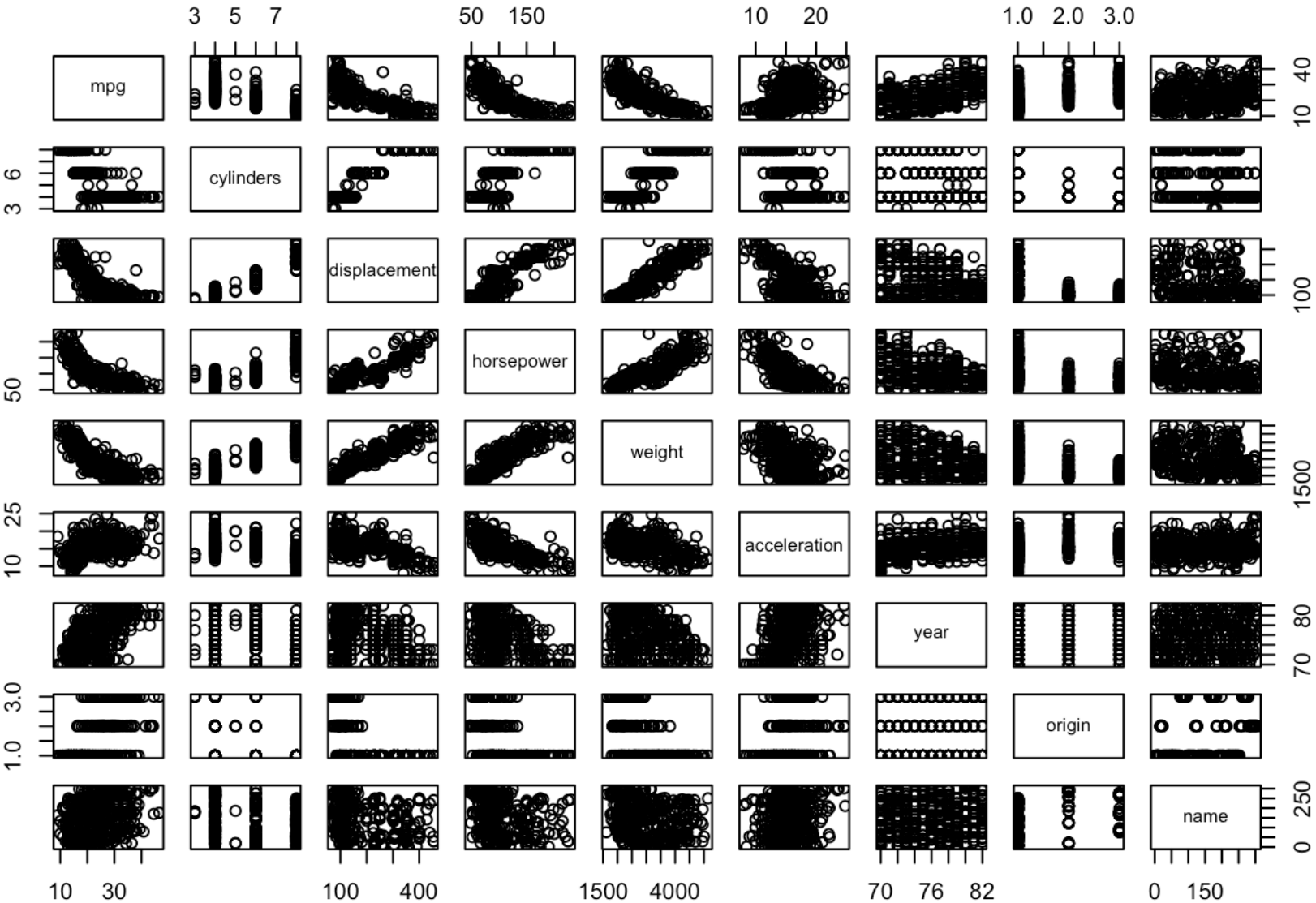
##	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
##	3186	6751	8377	9660	10830	56233

HW 2

Q1

a.

```
pairs(Auto)
```



b.

```
cor(Auto[1:8])
```

```
##          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.

```
fit <- lm(mpg ~ .-name, data = Auto)
summary(fit)
```

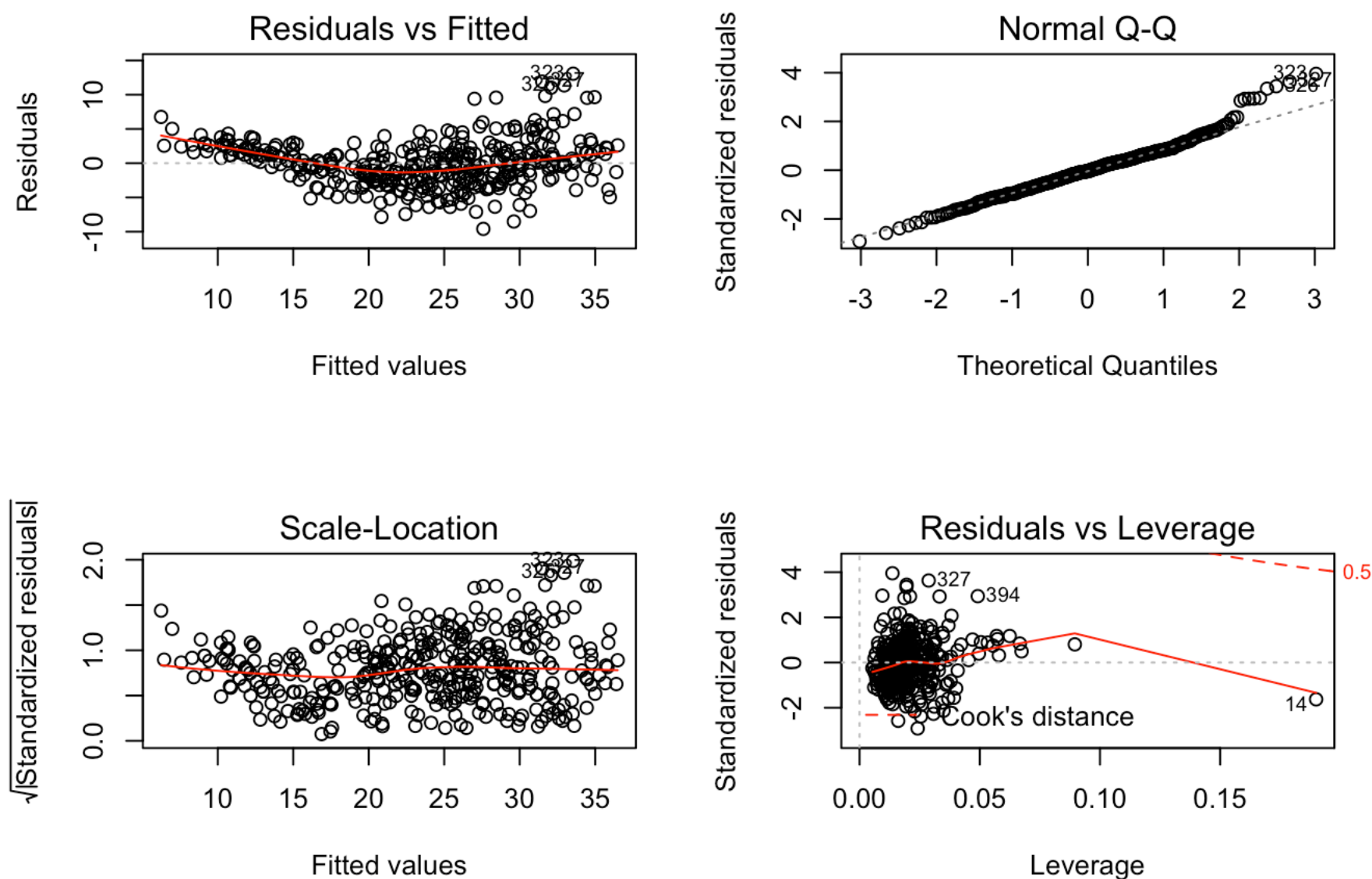
```
##
## Call:
## lm(formula = mpg ~ . - name, data = Auto)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -9.5903 -2.1565 -0.1169  1.8690 13.0604
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -17.218435    4.644294  -3.707  0.00024 ***
## 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
```


From the p-value obtained above, it can be said that all the variables, except only the variables `horsepower`, `cylinder` and `acceleration` are statistically significant.

The coefficient of `year` conveys that for unit change in the variable `year`, the variable under study changes by an amount of 0.7507, provided all the other values are kept constant.

d.

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



e.

```
fit1 <- lm(mpg ~ cylinders*displacement + displacement*weight, data = Auto)
summary(fit1)
```

```
##
## Call:
## lm(formula = mpg ~ cylinders * displacement + displacement *
##     weight, data = Auto)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.2934  -2.5184  -0.3476   1.8399  17.7723
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    5.262e+01  2.237e+00  23.519  < 2e-16 ***
## cylinders       7.606e-01  7.669e-01   0.992   0.322
## displacement  -7.351e-02  1.669e-02  -4.403 1.38e-05 ***
## weight        -9.888e-03  1.329e-03  -7.438 6.69e-13 ***
## cylinders:displacement -2.986e-03  3.426e-03  -0.872   0.384
## displacement:weight   2.128e-05  5.002e-06   4.254 2.64e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.103 on 386 degrees of freedom
## Multiple R-squared:  0.7272, Adjusted R-squared:  0.7237
## F-statistic: 205.8 on 5 and 386 DF,  p-value: < 2.2e-16
```

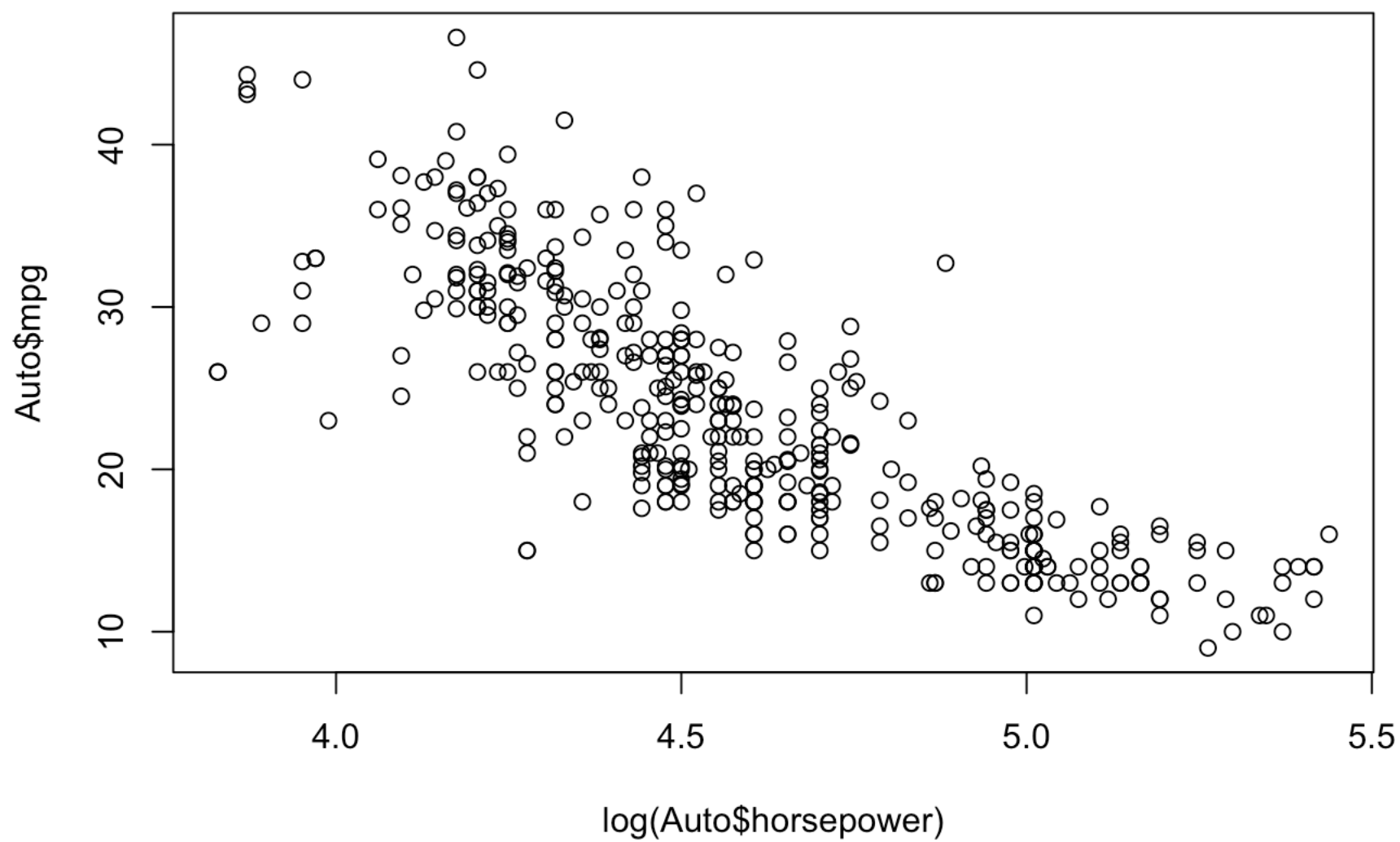
From the p-value, we could say that the relationship between `cylinder` and `displacement` is not statistically significant.

f.

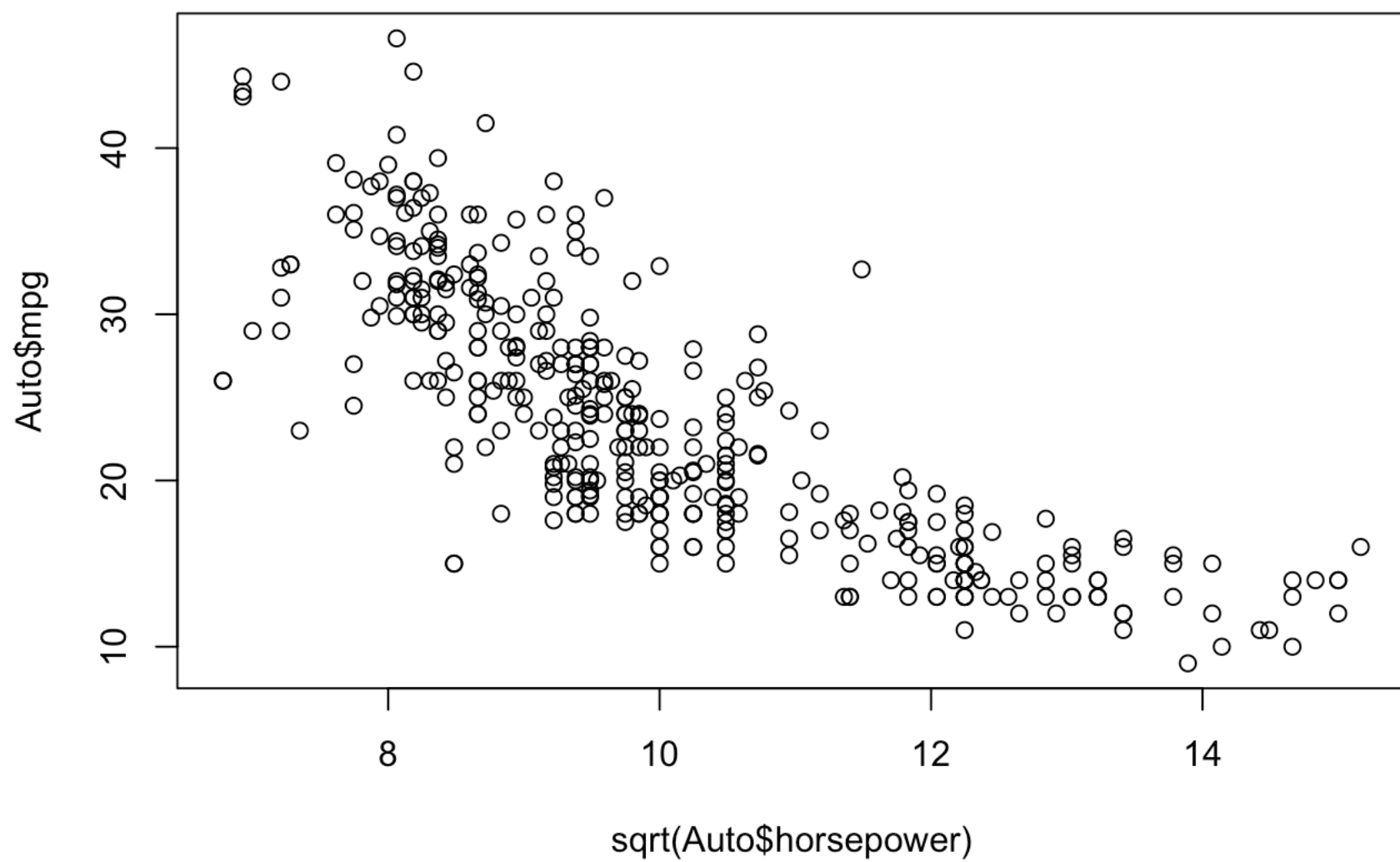
```
par(mfrow = c(2,2))
```

```
## Warning in par(mfrow = c(2, 2)): "mfrow" is not a graphical parameter
```

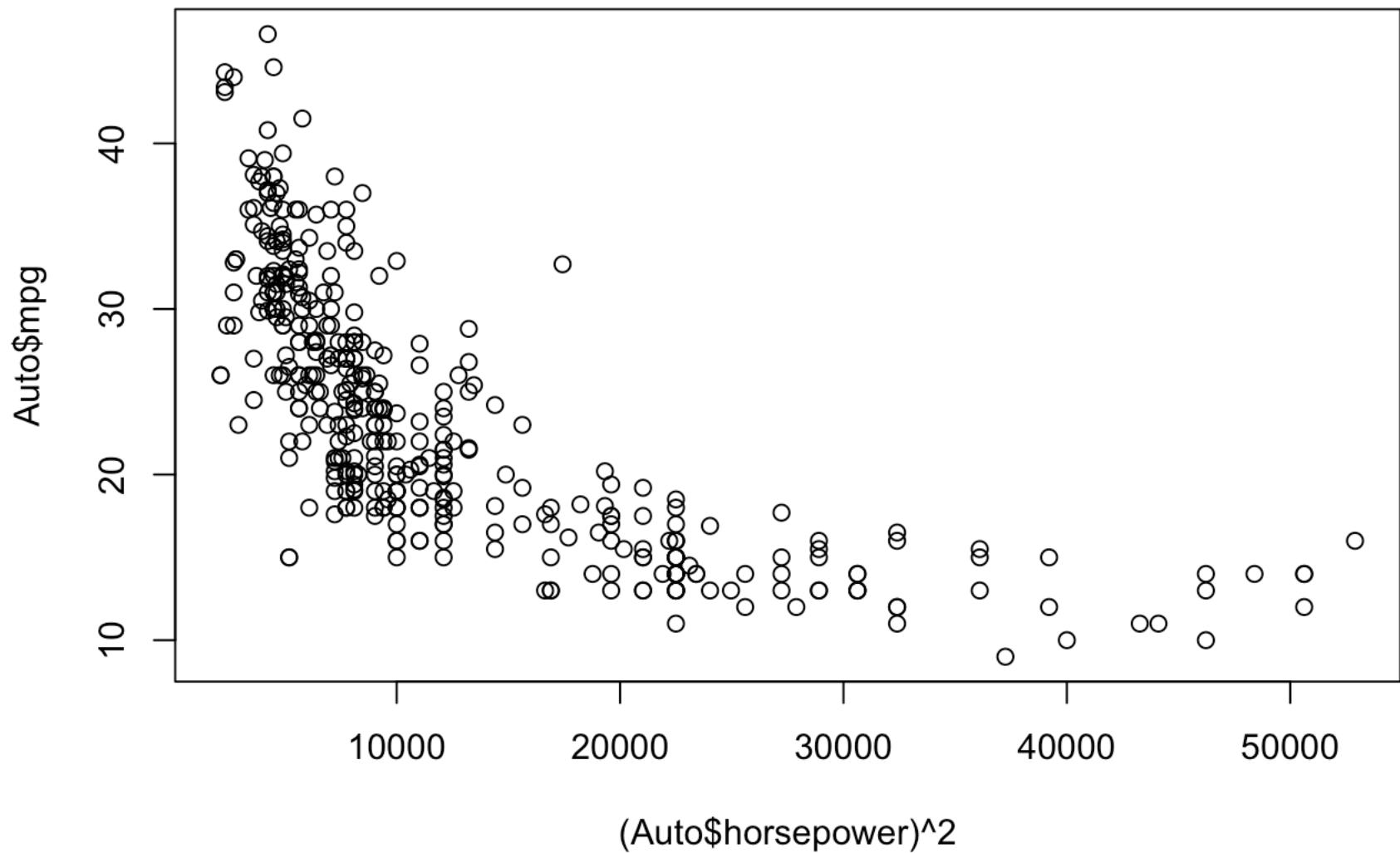
```
plot(log(Auto$horsepower), Auto$mpg)
```



```
plot(sqrt(Auto$horsepower), Auto$mpg)
```



```
plot((Auto$horsepower)^2, Auto$mpg)
```



Q15

a.

```
data(Boston)
coefs <- data.frame("predictor"=character(0), "Estimate"=numeric(0), "Std.Error"=numeric(0), "t.value"=numeric(0), "Pr.t"=numeric(0), "r.squared"=numeric(0), stringsAsFactors = FALSE)
j <- 1
for(i in names(Boston)){
  if(i != "crim"){
    summ.lm.fit <- summary(lm(crim ~ eval(parse(text=i)), data=Boston))
    coefs[j,] = c(i, summ.lm.fit$coefficients[2,], summ.lm.fit$r.squared)
    j <- j+1
  }
}

coefs[,-1] <- lapply(coefs[,-1], FUN=function(x) as.numeric(x))
coefs <- coefs[order(coefs$r.squared, decreasing = T),]
print(coefs)
```

##	predictor	Estimate	Std.Error	t.value	Pr.t	r.squared
## 8	rad	0.61791093	0.034331820	17.998199	2.693844e-56	0.391256687
## 9	tax	0.02974225	0.001847415	16.099388	2.357127e-47	0.339614243
## 12	lstat	0.54880478	0.047760971	11.490654	2.654277e-27	0.207590933
## 4	nox	31.24853120	2.999190381	10.418989	3.751739e-23	0.177217182
## 2	indus	0.50977633	0.051024332	9.990848	1.450349e-21	0.165310070
## 13	medv	-0.36315992	0.038390175	-9.459710	1.173987e-19	0.150780469
## 11	black	-0.03627964	0.003873154	-9.366951	2.487274e-19	0.148274239
## 7	dis	-1.55090168	0.168330031	-9.213458	8.519949e-19	0.144149375
## 6	age	0.10778623	0.012736436	8.462825	2.854869e-16	0.124421452
## 10	ptratio	1.15198279	0.169373609	6.801430	2.942922e-11	0.084068439
## 5	rm	-2.68405122	0.532041083	-5.044819	6.346703e-07	0.048069117
## 1	zn	-0.07393498	0.016094596	-4.593776	5.506472e-06	0.040187908
## 3	chas	-1.89277655	1.506115484	-1.256727	2.094345e-01	0.003123869

By p-value parameters, all predictors have a relevant association with response, rejecting the null hypothesis. By the R2 parameter, the response variance explained by the predictor, the most meaningful and also the best t-value is the rad variable. Either the tax variable is very well associated with the response, and it is the second of higher R2 value.

b.

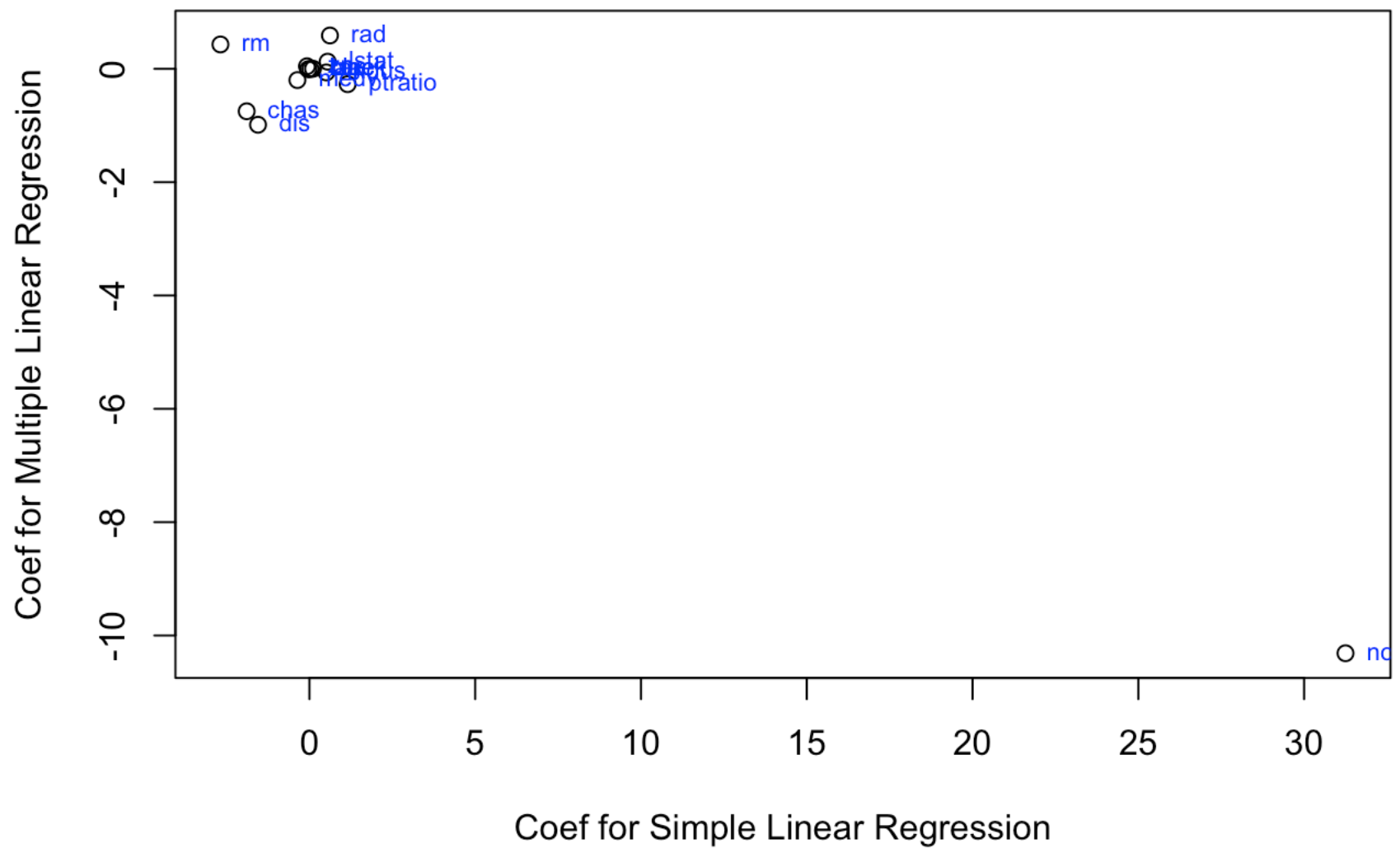
```
lm.fit.b <- lm(crim ~ ., data=Boston)
summary(lm.fit.b)
```

```
##
## 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 can reject the null hypothesis for: zn, nox, dis, rad, black, lstat and medv. They're 7 from 14 of the predictors.

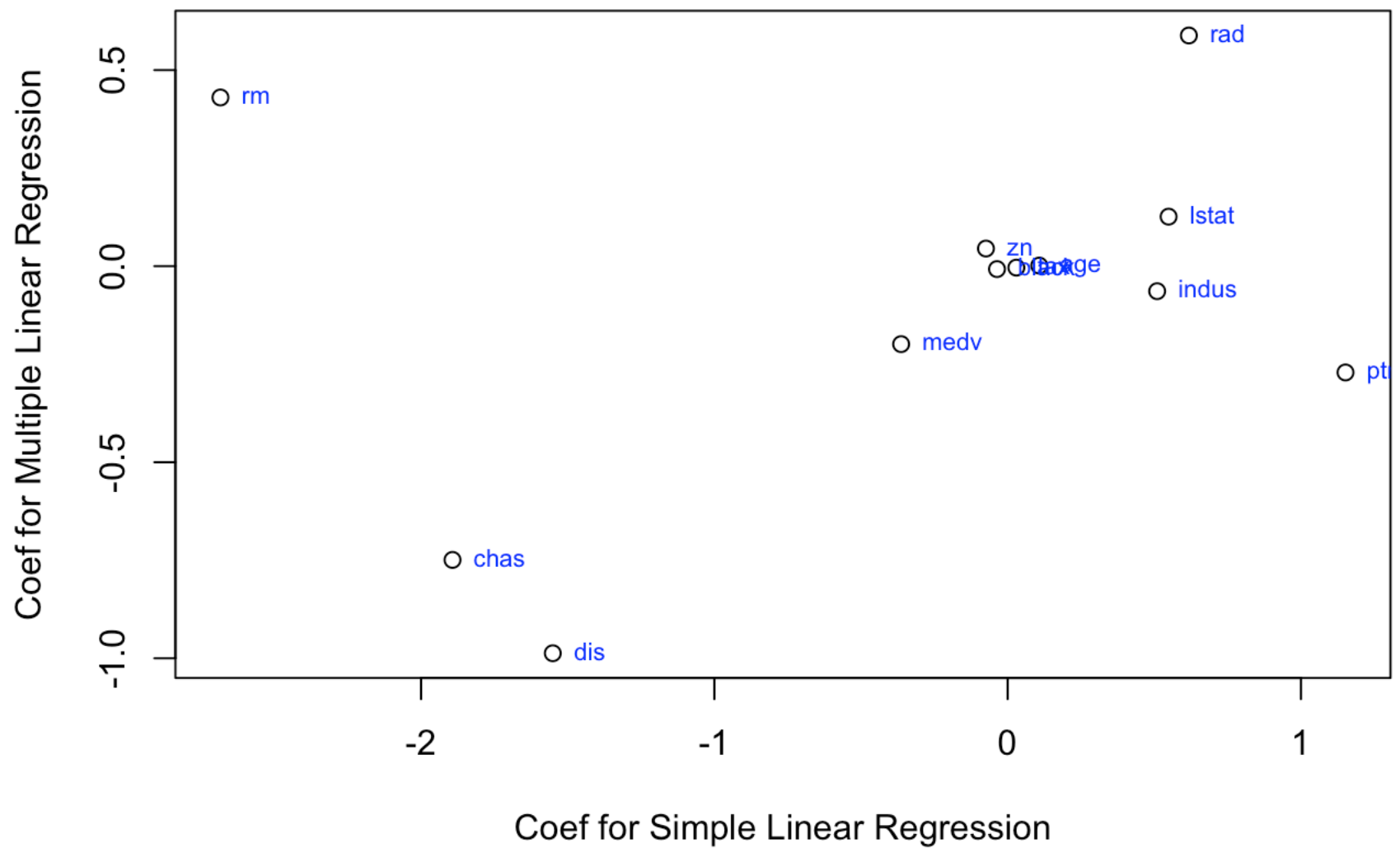
c.

```
df = data.frame("mult"=summary(lm.fit.b)$coefficients[-1,1])
df$simple <- NA
for(i in row.names(df)){
  df[row.names(df)==i, "simple"] = coefs[coefs[,1]==i, "Estimate"]
}
plot(df$simple, df$mult, xlab="Coef for Simple Linear Regression", ylab="Coef for Multiple Linear Regression")
text(x=df$simple, y=df$mult, labels=row.names(df), cex=.7, col="blue", pos=4)
```



The `nox` variable appears with a large displacement, messing the neatness of the graph, so i'll cut-off the `nox` to enhance the visualization.

```
df.clean = df[!(row.names(df)%in%"nox"),]
plot(df.clean$simple, df.clean$mult, xlab="Coef for Simple Linear Regression", ylab="
Coef for Multiple Linear Regression")
text(x=df.clean$simple, y=df.clean$mult, labels=row.names(df.clean), cex=.7, col="blue", pos=4)
```

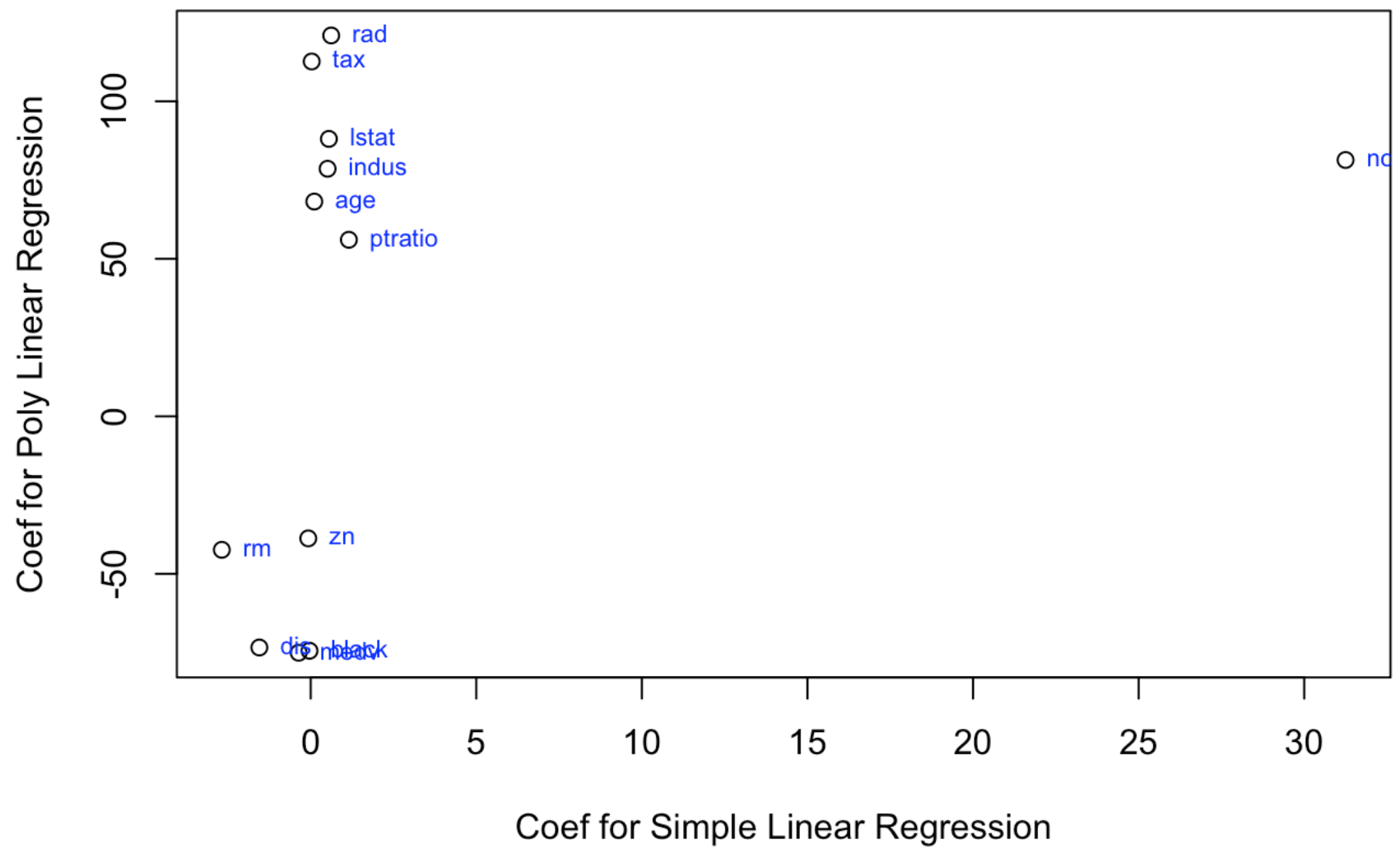
d.

```
coefs.poly <- data.frame("predictor"=character(0), "Estimate"=numeric(0), "Std.Error"
=numeric(0), "t.value"=numeric(0), "Pr.t"=numeric(0), "r.squared"=numeric(0), strings
AsFactors = FALSE)
j <- 1
for(i in names(Boston)){
  if(!(i %in% c("crim", "chas"))){
    summ.lm.fit <- summary(lm(crim ~ poly(eval(parse(text=i)),3), data=Boston))
    coefs.poly[j,] = c(i, summ.lm.fit$coefficients[2,], summ.lm.fit$r.squared)
    j <- j+1}}
coefs.poly[,-1] <- lapply(coefs.poly[,-1], FUN=function(x) as.numeric(x))
coefs.poly <- coefs.poly[order(coefs.poly$r.squared, decreasing = T),]
print(coefs.poly)
```

##	predictor	Estimate	Std.Error	t.value	Pr.t	r.squared
## 12	medv	-75.05761	6.569152	-11.425768	4.930818e-27	0.42020026
## 7	rad	120.90745	6.682402	18.093412	1.053211e-56	0.40003687
## 8	tax	112.64583	6.853707	16.435751	6.976314e-49	0.36888208
## 3	nox	81.37202	7.233605	11.249165	2.457491e-26	0.29697790
## 6	dis	-73.38859	7.331479	-10.010066	1.253249e-21	0.27782477
## 2	indus	78.59082	7.423121	10.587301	8.854243e-24	0.25965786
## 11	lstat	88.06967	7.629436	11.543404	1.678072e-27	0.21793243
## 5	age	68.18201	7.839703	8.697015	4.878803e-17	0.17423099
## 10	black	-74.43120	7.954643	-9.356951	2.730082e-19	0.14983983
## 9	ptratio	56.04523	8.121583	6.900777	1.565484e-11	0.11378158
## 4	rm	-42.37944	8.329676	-5.087766	5.128048e-07	0.06778606
## 1	zn	-38.74984	8.372207	-4.628389	4.697806e-06	0.05824197

For better analysis, i plot a graph between the coefficients in the simple linear graph and simple linear model with polynomial order.

```
df = data.frame("simple"=coefs[,2])
row.names(df) <- coefs[, 1]
df$poly <- NA
for(i in coefs.poly[,1]){
  df[row.names(df)==i, "poly"] <- coefs.poly[coefs.poly[,1]==i, "Estimate"]}
plot(df$simple, df$poly, xlab="Coef for Simple Linear Regression", ylab="Coef for Pol
y Linear Regression")
text(x=df$simple, y=df$poly, labels=row.names(df), cex=.7, col="blue", pos=4)
```



Titanic

```
library(titanic)
data("titanic_test")
testing <- titanic_test
summary(testing)
```

```
##      PassengerId      Pclass      Name      Sex
##  Min.      : 892.0    Min.      :1.000    Length:418    Length:418
##  1st Qu.: 996.2    1st Qu.:1.000    Class :character    Class :character
##  Median :1100.5    Median :3.000    Mode  :character    Mode  :character
##  Mean    :1100.5    Mean    :2.266
##  3rd Qu.:1204.8    3rd Qu.:3.000
##  Max.    :1309.0    Max.    :3.000
##
##      Age      SibSp      Parch      Ticket
##  Min.      : 0.17    Min.      :0.0000    Min.      :0.0000    Length:418
##  1st Qu.:21.00    1st Qu.:0.0000    1st Qu.:0.0000    Class :character
##  Median :27.00    Median :0.0000    Median :0.0000    Mode  :character
##  Mean    :30.27    Mean    :0.4474    Mean    :0.3923
##  3rd Qu.:39.00    3rd Qu.:1.0000    3rd Qu.:0.0000
##  Max.    :76.00    Max.    :8.0000    Max.    :9.0000
##  NA's     :86
##      Fare      Cabin      Embarked
##  Min.      : 0.000    Length:418    Length:418
##  1st Qu.: 7.896    Class :character    Class :character
##  Median : 14.454    Mode  :character    Mode  :character
##  Mean    : 35.627
##  3rd Qu.: 31.500
##  Max.    :512.329
##  NA's     :1
```

```
dim(testing)
```

```
## [1] 418 11
```

```
data("titanic_train")
training <- titanic_train
summary(training)
```

```
## PassengerId      Survived      Pclass      Name
## Min.      : 1.0    Min.      :0.0000    Min.      :1.000    Length:891
## 1st Qu.:223.5    1st Qu.:0.0000    1st Qu.:2.000    Class :character
## Median :446.0    Median :0.0000    Median :3.000    Mode  :character
## Mean    :446.0    Mean    :0.3838    Mean    :2.309
## 3rd Qu.:668.5    3rd Qu.:1.0000    3rd Qu.:3.000
## Max.    :891.0    Max.    :1.0000    Max.    :3.000
##
## Sex              Age              SibSp              Parch
## Length:891      Min.      : 0.42    Min.      :0.000    Min.      :0.0000
## Class :character 1st Qu.:20.12    1st Qu.:0.000    1st Qu.:0.0000
## Mode  :character Median :28.00    Median :0.000    Median :0.0000
##                      Mean    :29.70    Mean    :0.523    Mean    :0.3816
##                      3rd Qu.:38.00    3rd Qu.:1.000    3rd Qu.:0.0000
##                      Max.    :80.00    Max.    :8.000    Max.    :6.0000
##                      NA's    :177
## Ticket          Fare              Cabin              Embarked
## Length:891      Min.      : 0.00    Length:891      Length:891
## Class :character 1st Qu.: 7.91    Class :character Class :character
## Mode  :character Median : 14.45    Mode  :character Mode  :character
##                      Mean    : 32.20
##                      3rd Qu.: 31.00
##                      Max.    :512.33
##
```

```
dim(training)
```

```
## [1] 891 12
```

Cleaning the Missing Data

```
training1 <- training %>% na.omit()
```

```
testing1 <- testing %>% na.omit()
```

When we cleaning the Missing data, then we could not take the imputation in next steps.

Age Evaluation by Sex

```
training1 %>% group_by(Sex) %>%
  summarise(meanAge = mean(Age,na.rm = TRUE), medianAge = median(Age,na.rm = TRUE))
```

```
## # A tibble: 2 x 3
##   Sex      meanAge medianAge
##   <chr>    <dbl>    <dbl>
## 1 female    27.9        27
## 2 male     30.7        29
```

Age Evaluation by Pclass

```
training1 %>% group_by(Pclass) %>%
  summarise(meanAge = mean(Age,na.rm = TRUE), medianAge = median(Age,na.rm = TRUE))
```

```
## # A tibble: 3 x 3
##   Pclass meanAge medianAge
##   <int>    <dbl>    <dbl>
## 1      1    38.2        37
## 2      2    29.9        29
## 3      3    25.1        24
```

Data analysis

```
training1$`Survived` <- factor(training1$`Survived`)
training1$`Survived` <- fct_recode(training1$`Survived`,
                                   "Yes" = "1",
                                   "No" = "0")
Hmisc::describe(training1$`Survived`)
```

```
## training1$Survived
##           n missing distinct
##        714         0         2
##
## Value      No   Yes
## Frequency  424  290
## Proportion 0.594 0.406
```

```
t.test(training1$Age[training1$Survived == 'Yes'],
        training1$Age[training1$Survived == 'No'],
        var.equal = TRUE, paired = FALSE)
```

```
##
## Two Sample t-test
##
## data: training1$Age[training1$Survived == "Yes"] and training1$Age[training1$Survived == "No"]
## t = -2.0667, df = 712, p-value = 0.03912
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
## -4.450798 -0.114181
## sample estimates:
## mean of x mean of y
## 28.34369 30.62618
```

As we can see above, a 95% confidence interval for the mean standardized age of the surviving group is [-4.450798, -0.114181].

```
table(training1$Sex, training1$Pclass, training1$Survived)
```

```
## , , = No
##
##
##           1    2    3
## female    3    6   55
## male     61   84  215
##
## , , = Yes
##
##
##           1    2    3
## female   82   68   47
## male     40   15   38
```

```
library(caret)
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
## lift
```

```
## The following object is masked from 'package:mosaic':
##
## dotPlot
```

```
## The following object is masked from 'package:survival':  
##  
##      cluster
```

```
library(mlbench)  
logistTest <- train(Survived ~ Sex + Pclass, training1,  
                    method = 'glm', family = binomial())  
coef(summary(logistTest))
```

##	Estimate	Std. Error	z value	Pr(> z)
## (Intercept)	3.3467579	0.3231823	10.35563	3.945641e-25
## Sexmale	-2.5738858	0.2029872	-12.68004	7.629178e-37
## Pclass	-0.9910285	0.1182193	-8.38297	5.160873e-17

For the count statistics, we can see some interesting things, such as:

It is clear that, if you were a woman, your chances of survival were really good, despite if you were from 1st, 2nd or 3rd class. However, if you were a man, you would have to count in your economical status to have better chances of surviving the disaster. To have a quantitative evaluation over the importance of Sex and Pclass to predict survival, we made a mini logistic regression with just these two independent variables. As the P-values indicate, these two variables seem really good to explain the survival variability.

HW 3

Q13

```
summary(Boston)
```


##	crim	zn	indus	chas
##	Min. : 0.00632	Min. : 0.00	Min. : 0.46	Min. :0.00000
##	1st Qu.: 0.08204	1st Qu.: 0.00	1st Qu.: 5.19	1st Qu.:0.00000
##	Median : 0.25651	Median : 0.00	Median : 9.69	Median :0.00000
##	Mean : 3.61352	Mean : 11.36	Mean :11.14	Mean :0.06917
##	3rd Qu.: 3.67708	3rd Qu.: 12.50	3rd Qu.:18.10	3rd Qu.:0.00000
##	Max. :88.97620	Max. :100.00	Max. :27.74	Max. :1.00000
##	nox	rm	age	dis
##	Min. :0.3850	Min. :3.561	Min. : 2.90	Min. : 1.130
##	1st Qu.:0.4490	1st Qu.:5.886	1st Qu.: 45.02	1st Qu.: 2.100
##	Median :0.5380	Median :6.208	Median : 77.50	Median : 3.207
##	Mean :0.5547	Mean :6.285	Mean : 68.57	Mean : 3.795
##	3rd Qu.:0.6240	3rd Qu.:6.623	3rd Qu.: 94.08	3rd Qu.: 5.188
##	Max. :0.8710	Max. :8.780	Max. :100.00	Max. :12.127
##	rad	tax	ptratio	black
##	Min. : 1.000	Min. :187.0	Min. :12.60	Min. : 0.32
##	1st Qu.: 4.000	1st Qu.:279.0	1st Qu.:17.40	1st Qu.:375.38
##	Median : 5.000	Median :330.0	Median :19.05	Median :391.44
##	Mean : 9.549	Mean :408.2	Mean :18.46	Mean :356.67
##	3rd Qu.:24.000	3rd Qu.:666.0	3rd Qu.:20.20	3rd Qu.:396.23
##	Max. :24.000	Max. :711.0	Max. :22.00	Max. :396.90
##	lstat	medv		
##	Min. : 1.73	Min. : 5.00		
##	1st Qu.: 6.95	1st Qu.:17.02		
##	Median :11.36	Median :21.20		
##	Mean :12.65	Mean :22.53		
##	3rd Qu.:16.95	3rd Qu.:25.00		
##	Max. :37.97	Max. :50.00		

```
data("Boston")
crim01 <- rep(0, length(Boston$crim))
crim01[Boston$crim > median(Boston$crim)] <- 1
Boston <- data.frame(Boston, crim01)
summary(Boston)
```

##	crim	zn	indus	chas
##	Min. : 0.00632	Min. : 0.00	Min. : 0.46	Min. : 0.00000
##	1st Qu.: 0.08204	1st Qu.: 0.00	1st Qu.: 5.19	1st Qu.: 0.00000
##	Median : 0.25651	Median : 0.00	Median : 9.69	Median : 0.00000
##	Mean : 3.61352	Mean : 11.36	Mean : 11.14	Mean : 0.06917
##	3rd Qu.: 3.67708	3rd Qu.: 12.50	3rd Qu.: 18.10	3rd Qu.: 0.00000
##	Max. : 88.97620	Max. : 100.00	Max. : 27.74	Max. : 1.00000
##	nox	rm	age	dis
##	Min. : 0.3850	Min. : 3.561	Min. : 2.90	Min. : 1.130
##	1st Qu.: 0.4490	1st Qu.: 5.886	1st Qu.: 45.02	1st Qu.: 2.100
##	Median : 0.5380	Median : 6.208	Median : 77.50	Median : 3.207
##	Mean : 0.5547	Mean : 6.285	Mean : 68.57	Mean : 3.795
##	3rd Qu.: 0.6240	3rd Qu.: 6.623	3rd Qu.: 94.08	3rd Qu.: 5.188
##	Max. : 0.8710	Max. : 8.780	Max. : 100.00	Max. : 12.127
##	rad	tax	ptratio	black
##	Min. : 1.000	Min. : 187.0	Min. : 12.60	Min. : 0.32
##	1st Qu.: 4.000	1st Qu.: 279.0	1st Qu.: 17.40	1st Qu.: 375.38
##	Median : 5.000	Median : 330.0	Median : 19.05	Median : 391.44
##	Mean : 9.549	Mean : 408.2	Mean : 18.46	Mean : 356.67
##	3rd Qu.: 24.000	3rd Qu.: 666.0	3rd Qu.: 20.20	3rd Qu.: 396.23
##	Max. : 24.000	Max. : 711.0	Max. : 22.00	Max. : 396.90
##	lstat	medv	crim01	
##	Min. : 1.73	Min. : 5.00	Min. : 0.0	
##	1st Qu.: 6.95	1st Qu.: 17.02	1st Qu.: 0.0	
##	Median : 11.36	Median : 21.20	Median : 0.5	
##	Mean : 12.65	Mean : 22.53	Mean : 0.5	
##	3rd Qu.: 16.95	3rd Qu.: 25.00	3rd Qu.: 1.0	
##	Max. : 37.97	Max. : 50.00	Max. : 1.0	

```
set.seed(2019)
train <- sample(1:dim(Boston)[1], dim(Boston)[1]*.7, rep=FALSE)
test <- train
Boston.train <- Boston[train, ]
Boston.test <- Boston[test, ]
crim01.test <- crim01[test]
```

```
fit.glm13 <- glm(crim01 ~ . - crim01 - crim, data = Boston, family = binomial)
fit.glm13
```

```
##
## Call:  glm(formula = crim01 ~ . - crim01 - crim, family = binomial,
##       data = Boston)
##
## Coefficients:
## (Intercept)          zn          indus          chas          nox
## -34.103704    -0.079918   -0.059389    0.785327    48.523782
##          rm          age          dis          rad          tax
##  -0.425596    0.022172    0.691400    0.656465   -0.006412
##      ptratio        black        lstat          medv
##    0.368716   -0.013524    0.043862    0.167130
##
## Degrees of Freedom: 505 Total (i.e. Null);  492 Residual
## Null Deviance:          701.5
## Residual Deviance: 211.9      AIC: 239.9
```

```
fit.glm <- glm(crim01 ~ nox + indus + age + rad, data = Boston, family = binomial)
```

```
probs <- predict(fit.glm, Boston.test, type = "response")
pred.glm <- rep(0, length(probs))
pred.glm[probs > 0.5] <- 1
table(pred.glm, crim01.test)
```

```
##          crim01.test
## pred.glm    0     1
##           0 158   34
##           1  18  144
```

```
mean(pred.glm != crim01.test)
```

```
## [1] 0.1468927
```

For the logistic regression, we have a test error rate of 12.5%.

LDA

```
fit.lda <- lda(crim01 ~ nox + indus + age + rad , data = Boston)
pred.lda <- predict(fit.lda, Boston.test)
table(pred.lda$class, crim01.test)
```

```
##          crim01.test
##           0     1
##          0 166   46
##          1  10  132
```

```
mean(pred.lda$class != crim01.test)
```

```
## [1] 0.1581921
```

For the LDA regression model, we have a test error rate of 15.1%.

KNN

```
data = scale(Boston[, -c(1, 15)])
set.seed(2019)
train <- sample(1:dim(Boston)[1], dim(Boston)[1]*.7, rep=FALSE)
test <- -train
training_data = data[train, c("nox" , "indus" , "age" , "rad")]
testing_data = data[test, c("nox" , "indus" , "age" , "rad")]
train.crime01 = Boston$crim01[train]
test.crime01 = Boston$crim01[test]
```

```
set.seed(2019)
knn_pred_y = knn(training_data, testing_data, train.crime01, k = 1)
table(knn_pred_y, test.crime01)
```

```
##           test.crime01
## knn_pred_y  0   1
##           0 71   3
##           1   6 72
```

```
mean(knn_pred_y != test.crime01)
```

```
## [1] 0.05921053
```

For this KNN (k=1), we have a test error rate of 9.21%

```
knn_pred_y = NULL
error_rate = NULL
for(i in 1:dim(testing_data)[1]){
  set.seed(2019)
  knn_pred_y = knn(training_data, testing_data, train.crime01, k=i)
  error_rate[i] = mean(test.crime01 != knn_pred_y)
}
min_error_rate = min(error_rate)
print(min_error_rate)
```

```
## [1] 0.03947368
```

Minimum error rate is 6.57%.