STA380 Take Home Exam

```
library(MASS)
library(ggplot2)
library(corrplot)
```

```
## corrplot 0.84 loaded
```

Chapter 2, Number 10

a)

```
nrow(Boston)
```

```
## [1] 506
```

```
ncol(Boston)
```

```
## [1] 14
```

Each of the 506 rows represents one of the tracts in the 1970 census of Boston. Each column represents a measured variable.

By running the command "?Boston" the following column descriptions can be found:

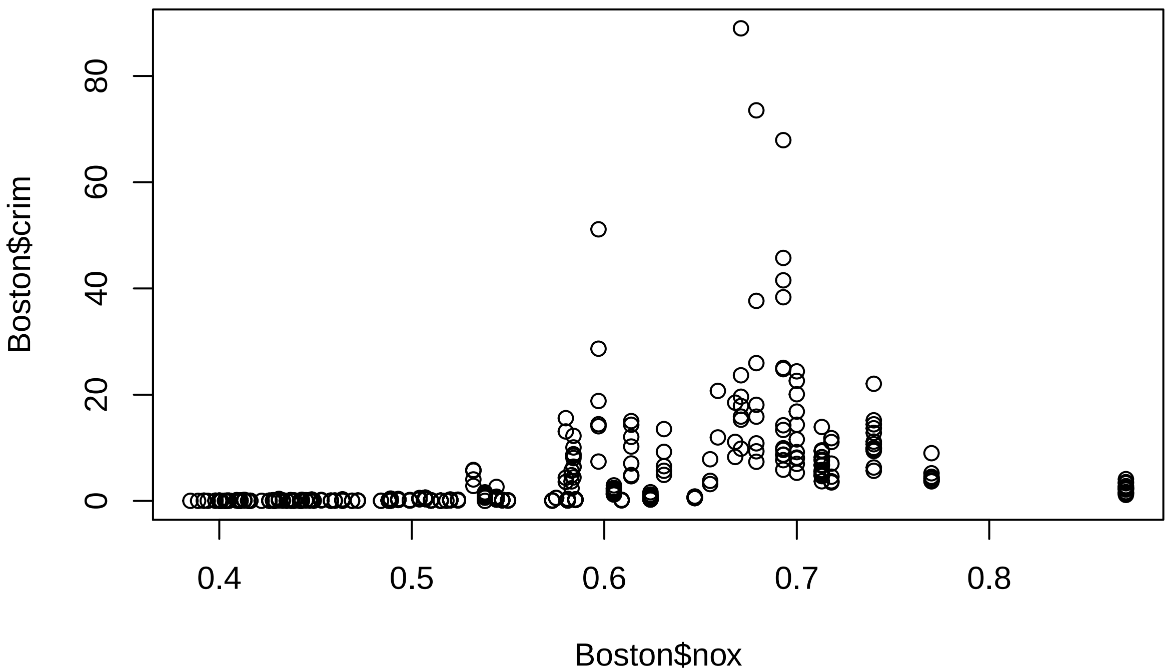
This data frame contains the following columns:

crim: per capita crime rate by town.
zn: proportion of residential land zoned for lots over 25,000 sq.ft.
indus: proportion of non-retail business acres per town.
chas:Charles River dummy variable (= 1 if tract bounds river; 0 otherwise).
nox: nitrogen oxides concentration (parts per 10 million).
rm: average number of rooms per dwelling.
age: proportion of owner-occupied units built prior to 1940.
dis: weighted mean of distances to five Boston employment centres.
rad: index of accessibility to radial highways.
tax: full-value property-tax rate per $10,000.
ptratio: pupil-teacher ratio by town.
black: 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town.
lstat: lower status of the population (percent).
medv: median value of owner-occupied homes in $1000s

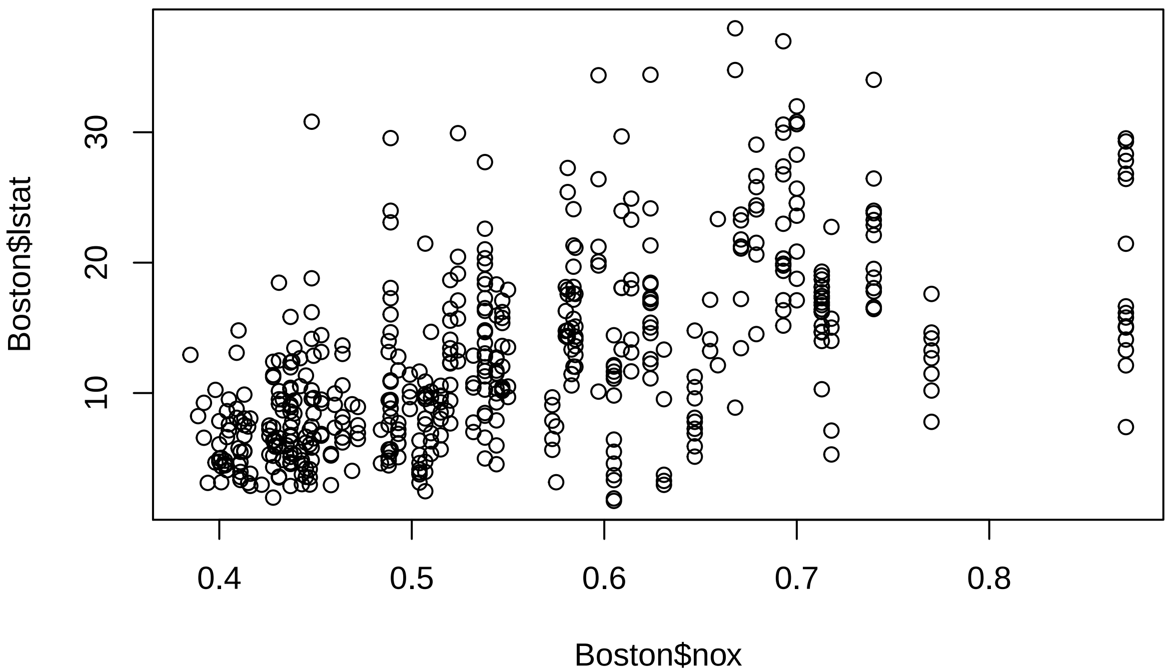
b)

I plotted several different predictors for crim and lstat that I expected to find trends within. As expected, comparing a predictor's effect on these had similiar shapes.

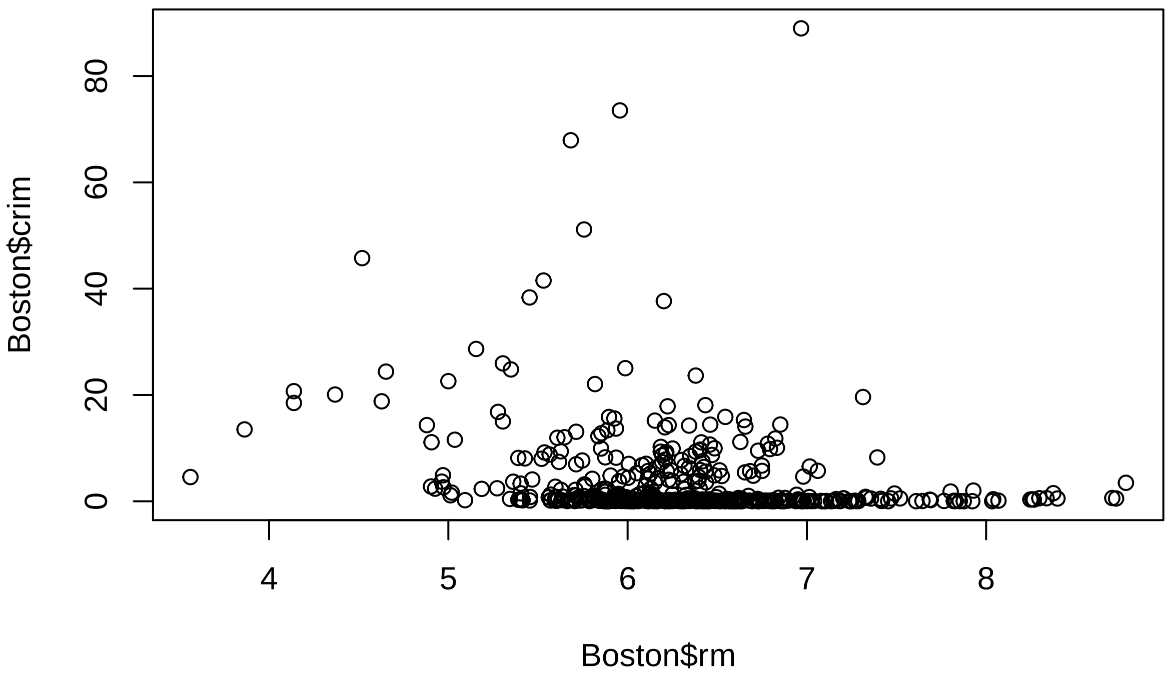
```
plot(Boston$nox, Boston$crim)
```



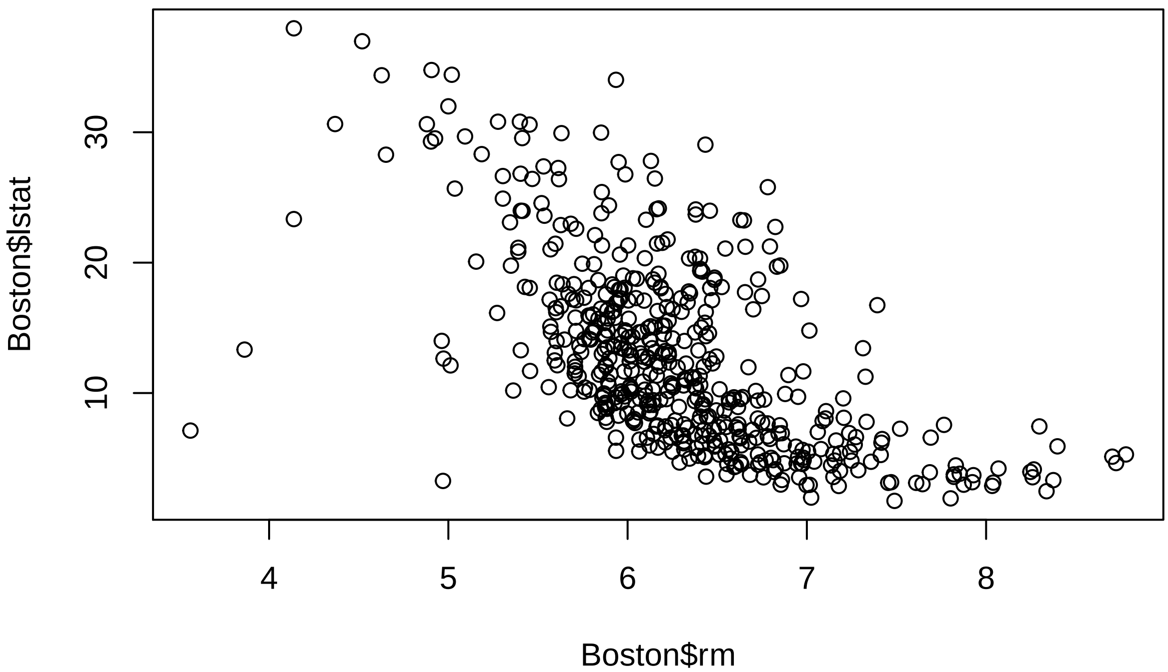

```r
plot(Boston$nox, Boston$lstat)
```



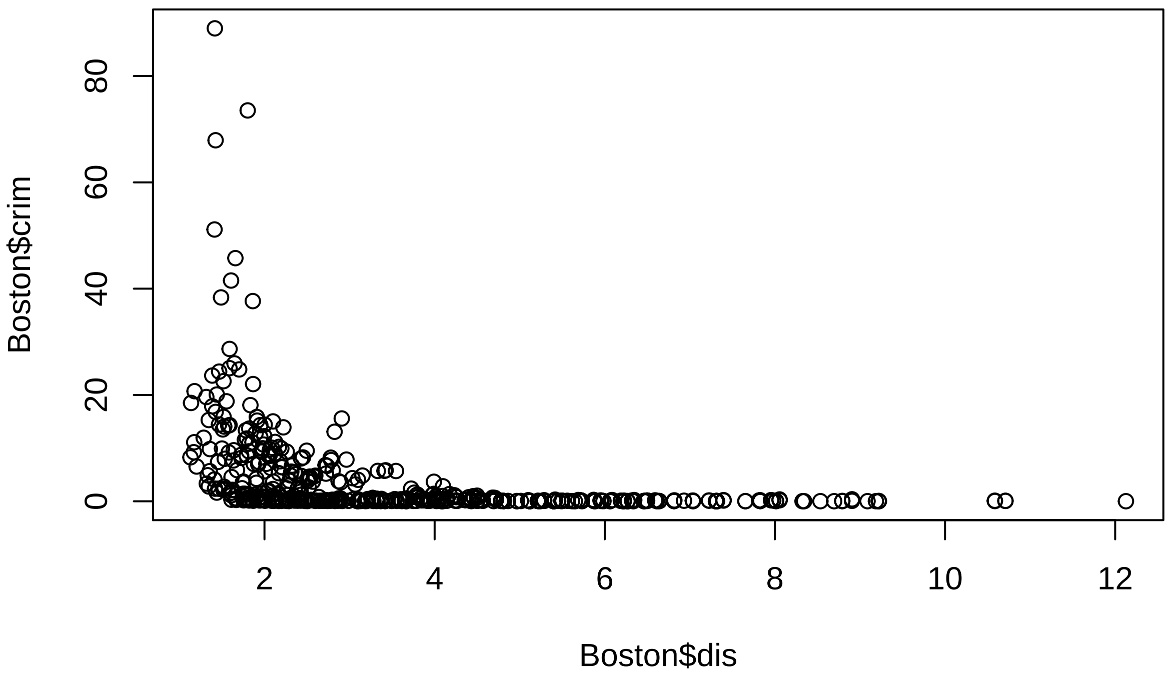

```r
plot(Boston$rm, Boston$crim)
```



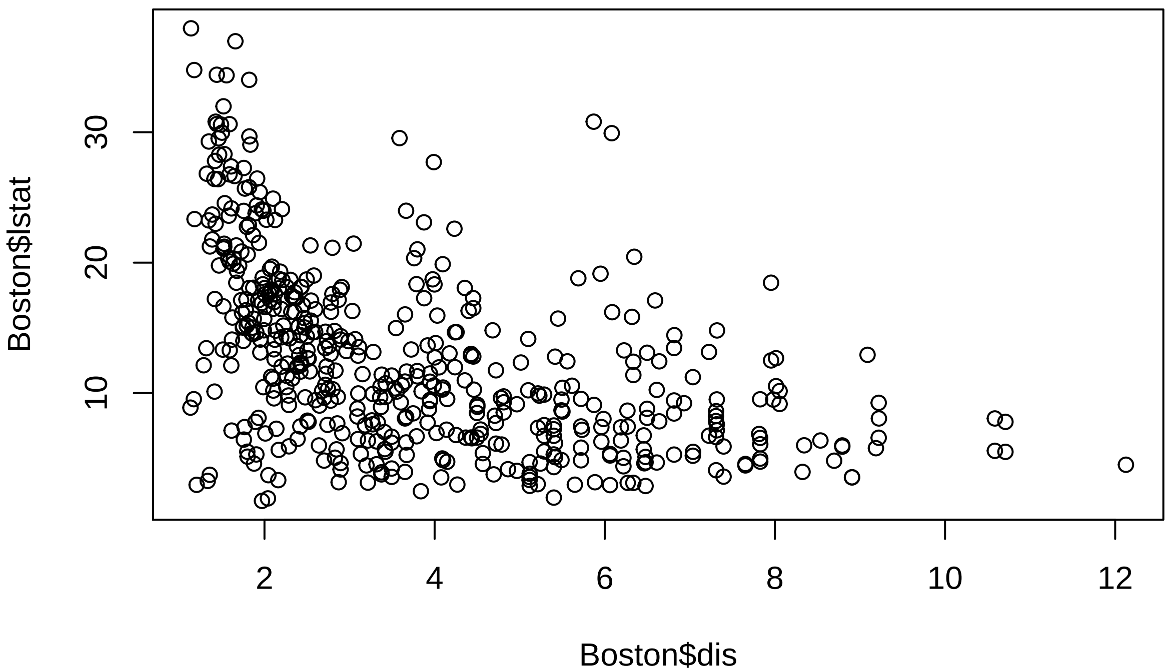

```r
plot(Boston$rm, Boston$lstat)
```



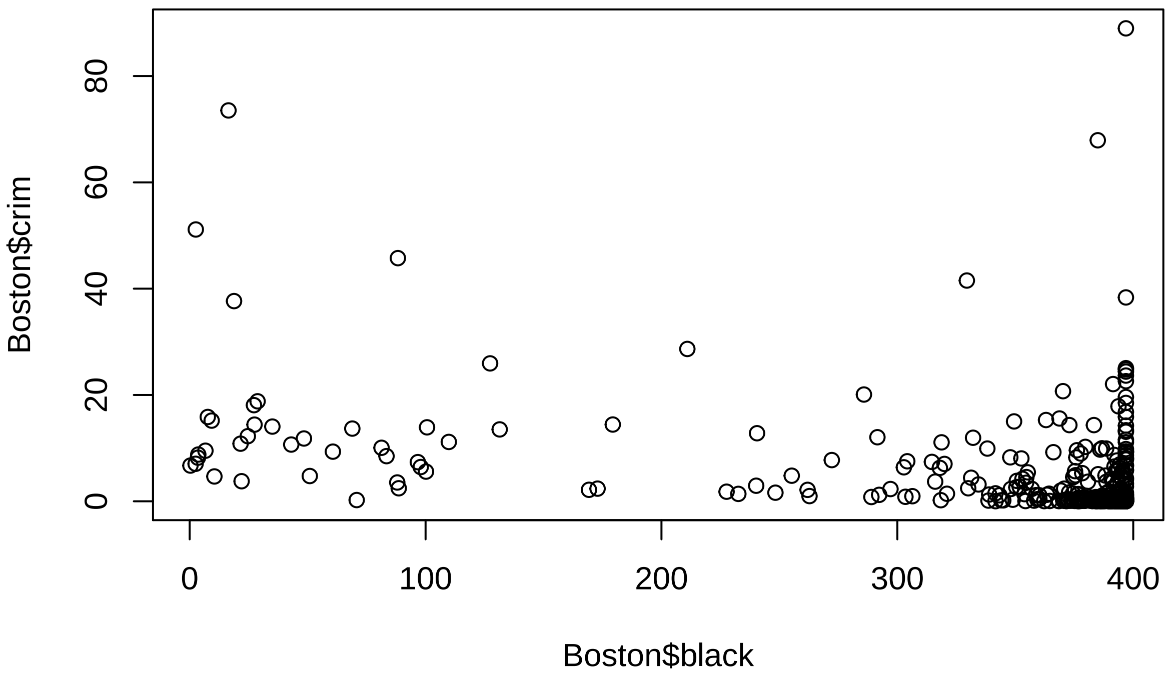

```
plot(Boston$dis, Boston$crim)
```



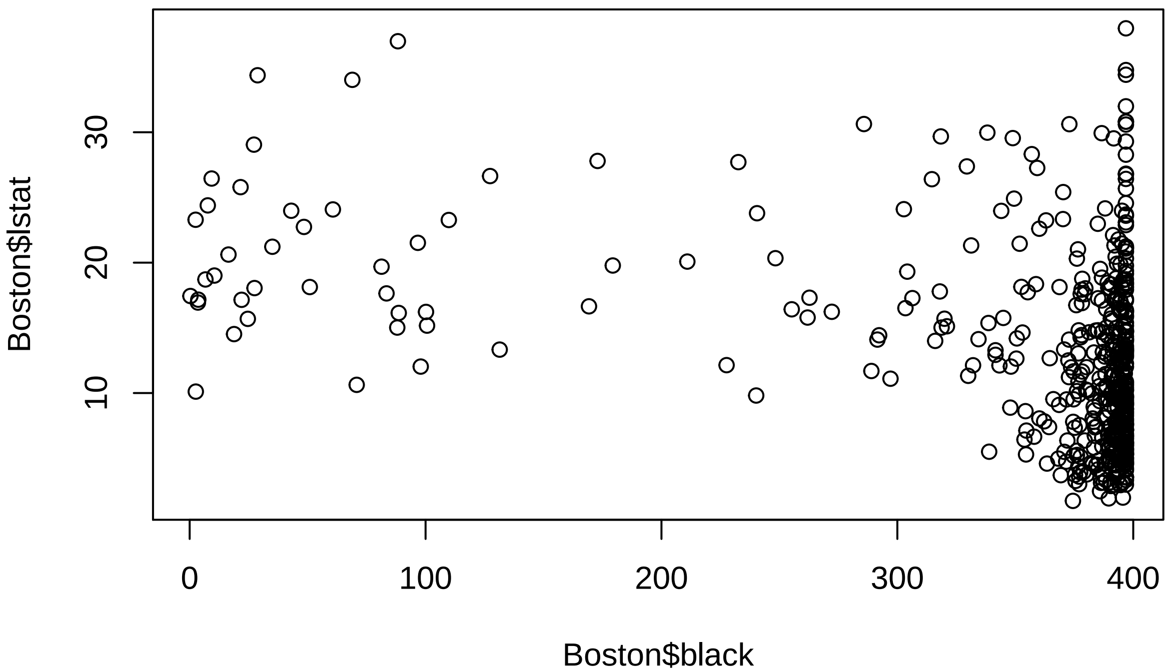

```r
plot(Boston$dis, Boston$lstat)
```



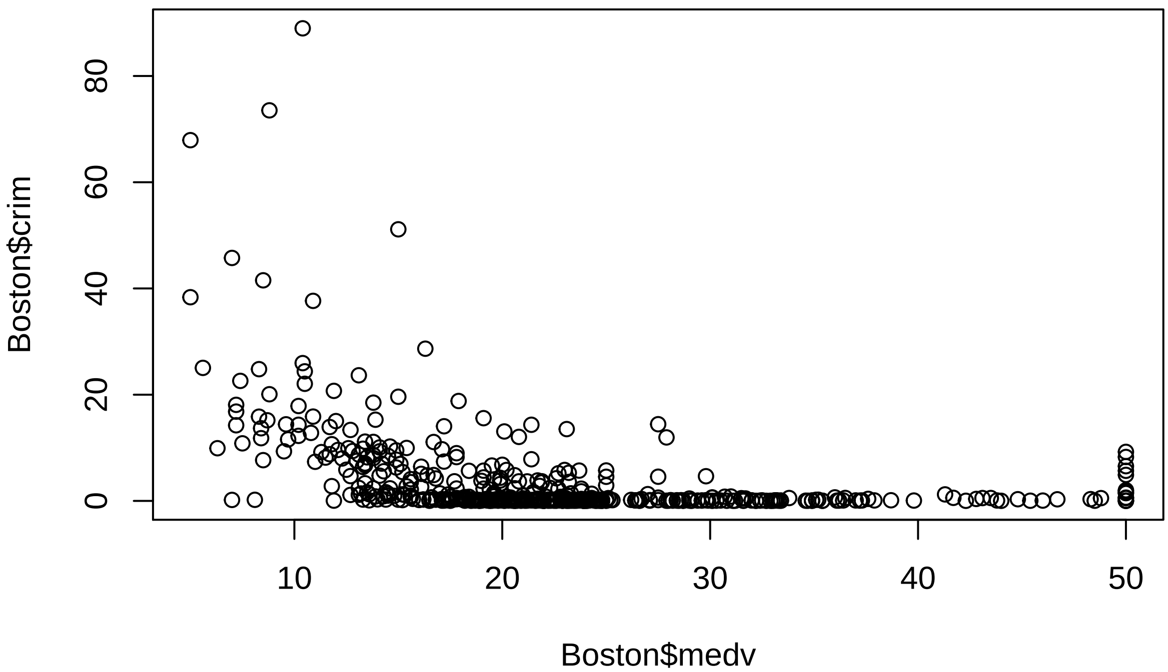

```r
plot(Boston$black, Boston$crim)
```



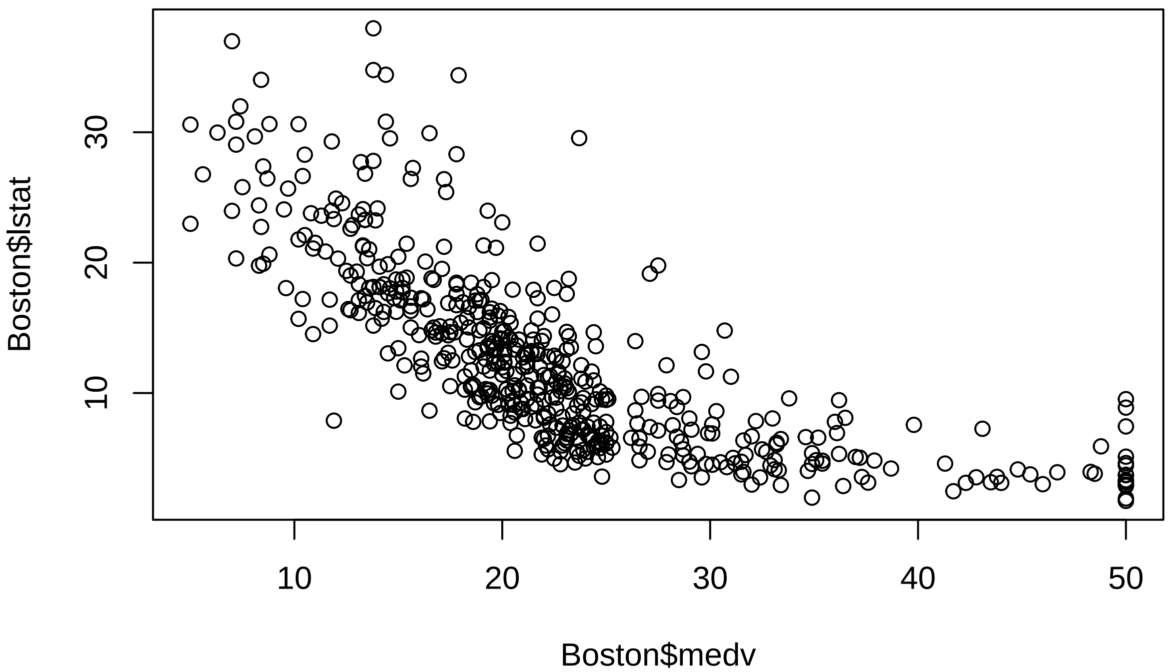

```r
plot(Boston$black, Boston$lstat)
```



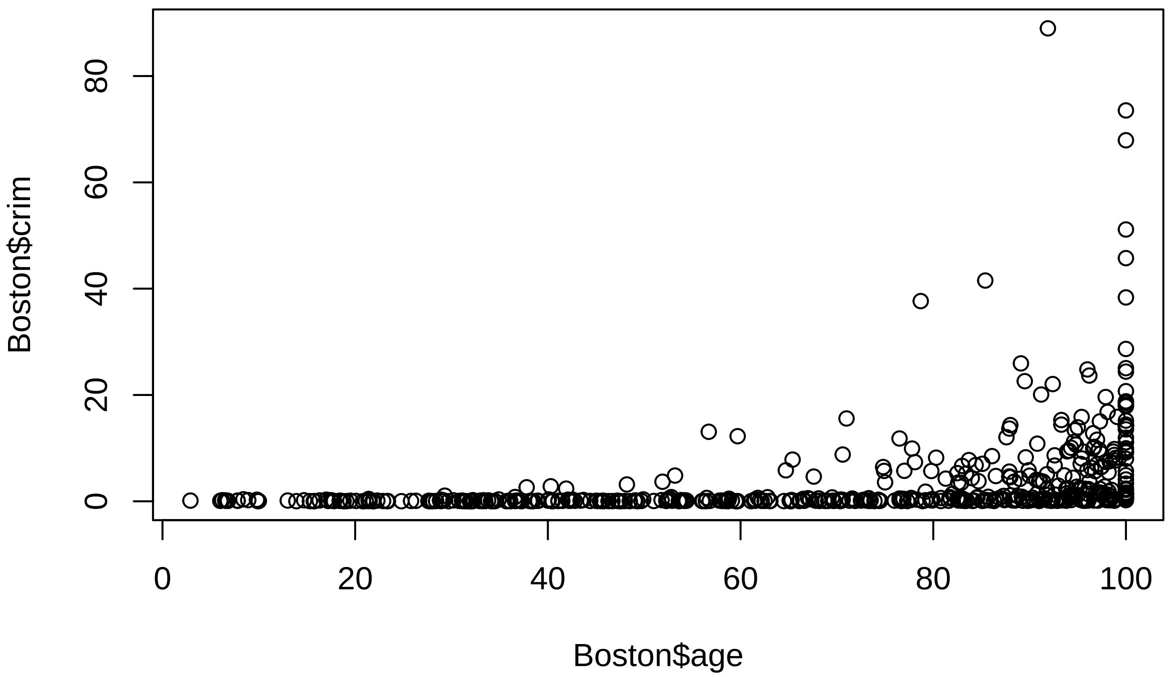

```
plot(Boston$medv, Boston$crim)
```



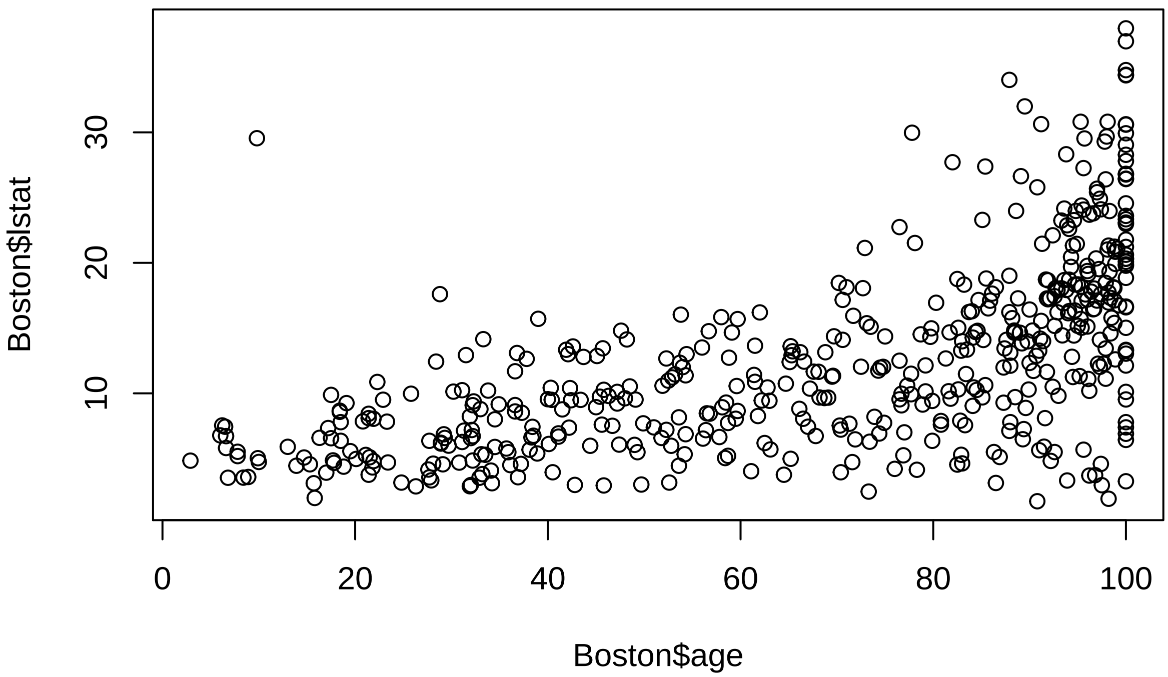

```r
plot(Boston$medv, Boston$lstat)
```



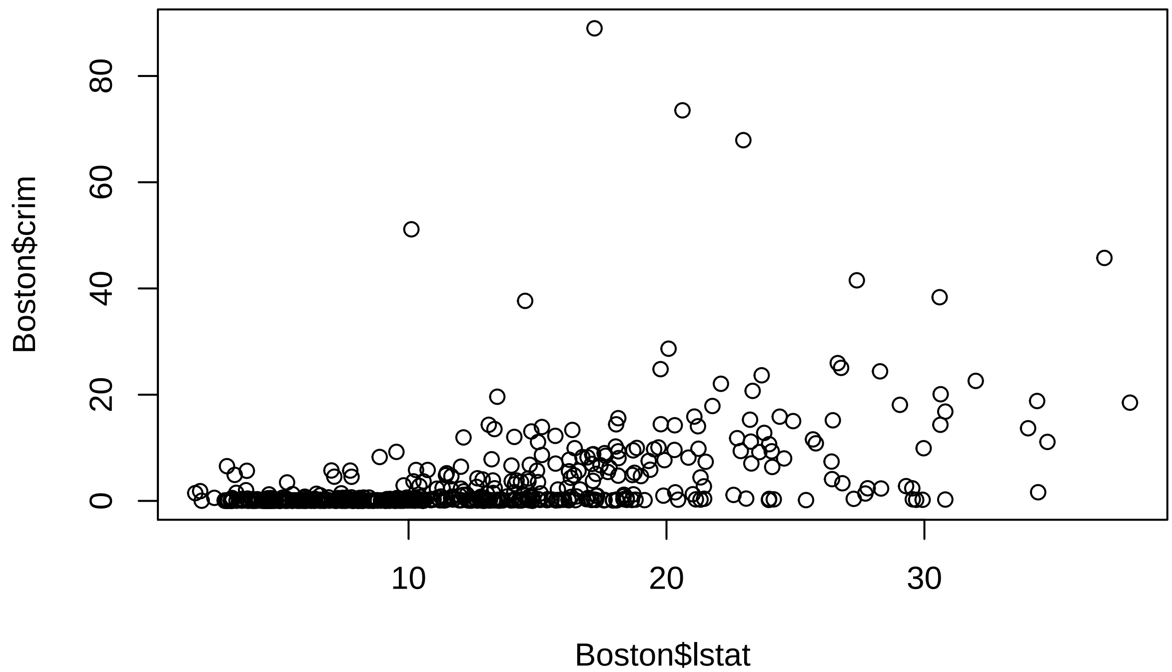

```
plot(Boston$age, Boston$crim)
```




```r
plot(Boston$age, Boston$lstat)
```

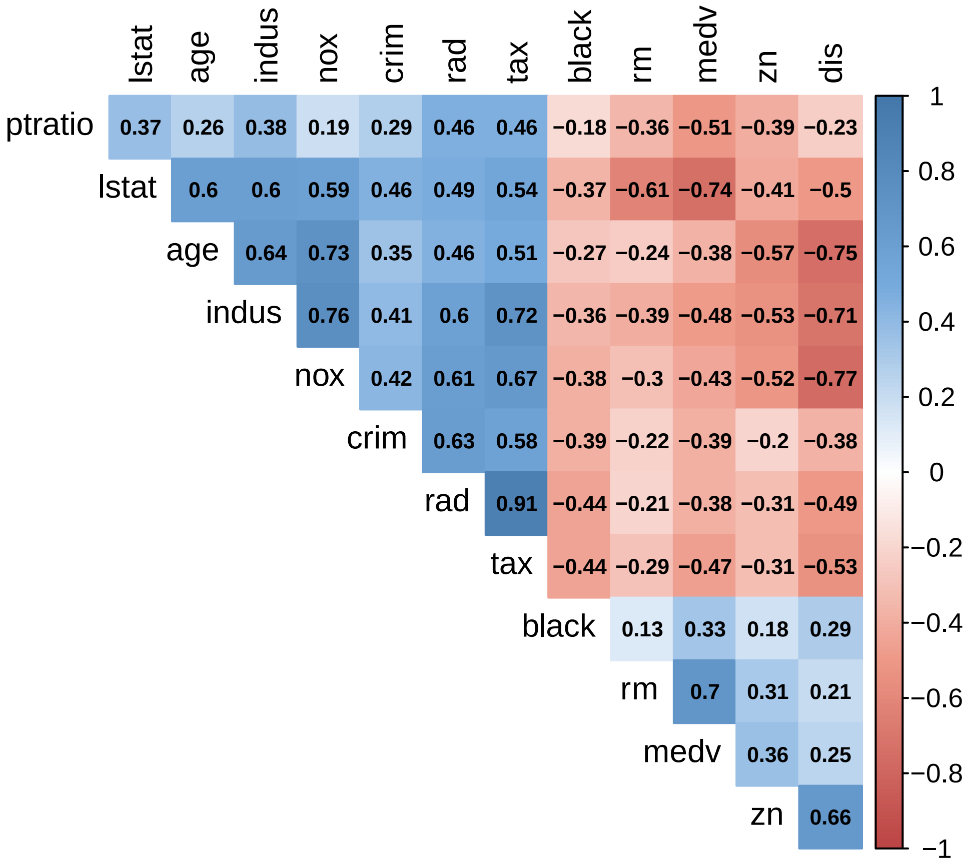



```
plot(Boston$lstat, Boston$crim)
```



c)

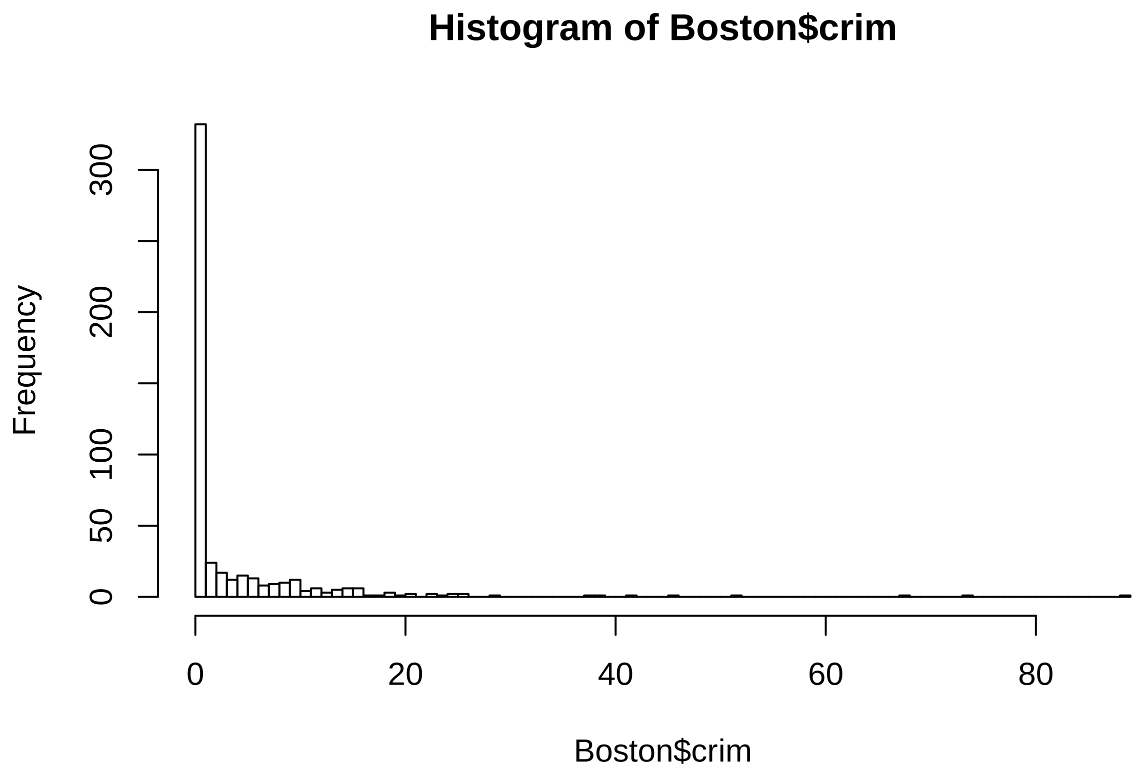
```r
Boston$chas <- as.factor(Boston$chas)

M <- cor(Boston[-4])

col <- colorRampPalette(c("#BB4444", "#EE9988", "#FFFFFF", "#77AADD", "#4477AA"))
corrplot(M, method = "color", col = col(200),
         type = "upper", order = "hclust", number.cex = .7,
         addCoef.col = "black", # Add coefficient of correlation
         tl.col = "black", tl.srt = 90, # Text label color and rotation
         # Combine with significance
         sig.level = 0.01, insig = "blank",
         # hide correlation coefficient on the principal diagonal
         diag = FALSE)
```



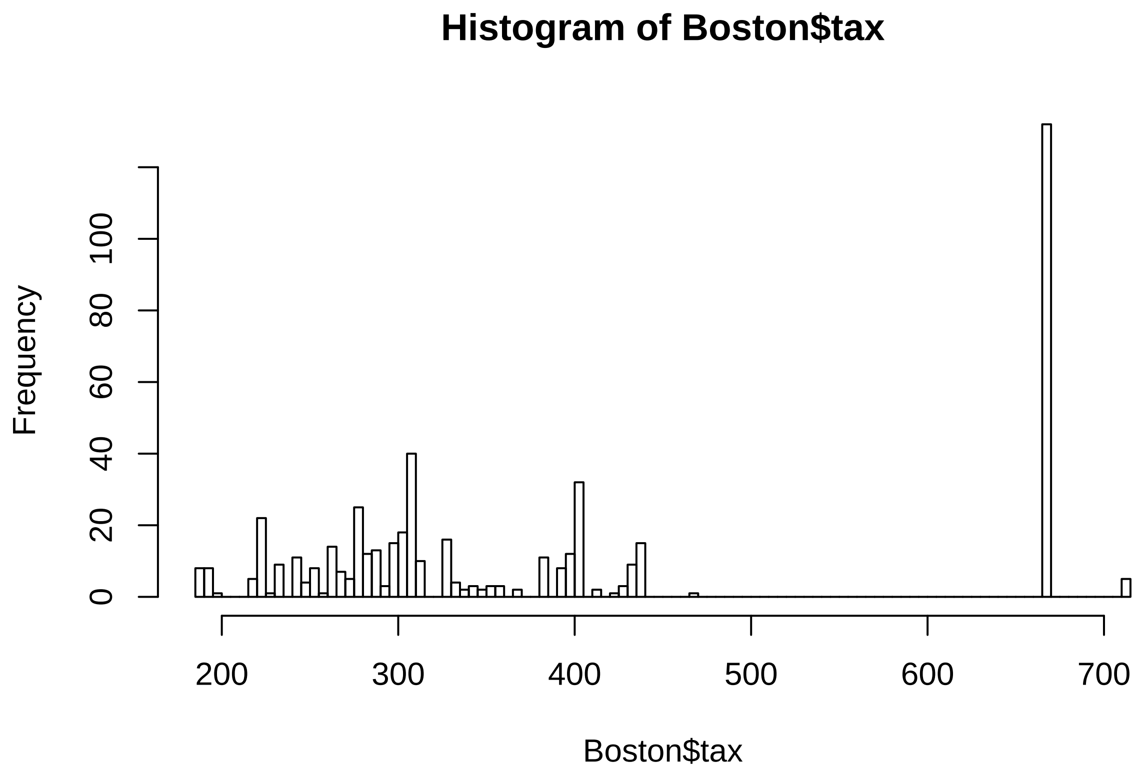
It appears that several variables included have a (relatively) high correlation with the per capita crime rate.

d)

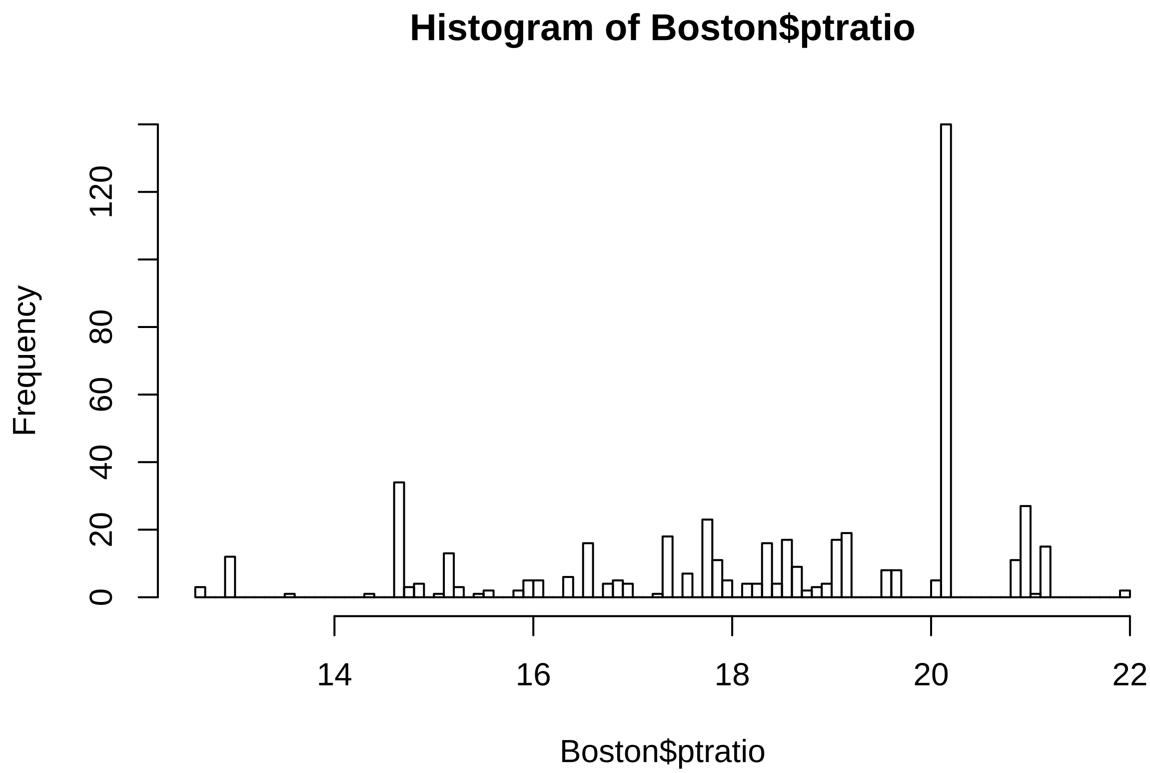
```r
hist(Boston$crim, breaks = 100)
```




```
hist(Boston$tax, breaks = 100)
```




```r
hist(Boston$ptratio, breaks = 100)
```




```r
nrow(Boston[Boston$crim < 10 , ])
```

```
## [1] 452
```

It appears that the majority of the census tracts have a low crime per capita.

```r
nrow(Boston[Boston$tax > 700,])
```

```
## [1] 5
```

```r
nrow(Boston[Boston$tax > 650,])
```

```
## [1] 137
```

While most of the variable tax appears aomewhat normally distributed for lower values, there are 137 tracts at a significantly higher level.A similiar spike of ptratio happens around 20.

e)

```r
nrow(Boston[Boston$chas == 1, ])
```

```
## [1] 35
```

35 suburbs in the data set cound the Charles River

f)

```r
median(Boston$ptratio)
```

```
## [1] 19.05
```

The median n pupil-teacher ratio is 19.05

g)

```r
Boston[min(Boston$medv), ]
```

```
##      crim zn indus chas   nox    rm  age    dis rad tax ptratio black
## 5 0.06905  0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7 396.9
##   lstat medv
## 5  5.33 36.2
```

It appears that the most notable feature of this particuliar suburb is that it is near the bottom range of the tax variable (understanble considering that these are both metrics for home value)

h)

```r
head(Boston[Boston$rm > 7, ])
```

```
##       crim   zn indus chas    nox    rm  age    dis rad tax ptratio  black
## 3  0.02729  0.0  7.07    0 0.4690 7.185 61.1 4.9671   2 242    17.8 392.83
## 5  0.06905  0.0  2.18    0 0.4580 7.147 54.2 6.0622   3 222    18.7 396.90
## 41 0.03359 75.0  2.95    0 0.4280 7.024 15.8 5.4011   3 252    18.3 395.62
## 56 0.01311 90.0  1.22    0 0.4030 7.249 21.9 8.6966   5 226    17.9 395.93
## 65 0.01951 17.5  1.38    0 0.4161 7.104 59.5 9.2229   3 216    18.6 393.24
## 89 0.05660  0.0  3.41    0 0.4890 7.007 86.3 3.4217   2 270    17.8 396.90
##    lstat medv
## 3   4.03 34.7
## 5   5.33 36.2
## 41  1.98 34.9
## 56  4.81 35.4
## 65  8.05 33.0
## 89  5.50 23.6
```

```r
head(Boston[Boston$rm > 8, ])
```

```
##        crim zn indus chas    nox    rm  age    dis rad tax ptratio  black
## 98  0.12083  0  2.89    0 0.4450 8.069 76.0 3.4952   2 276    18.0 396.90
## 164 1.51902  0 19.58    1 0.6050 8.375 93.9 2.1620   5 403    14.7 388.45
## 205 0.02009 95  2.68    0 0.4161 8.034 31.9 5.1180   4 224    14.7 390.55
## 225 0.31533  0  6.20    0 0.5040 8.266 78.3 2.8944   8 307    17.4 385.05
## 226 0.52693  0  6.20    0 0.5040 8.725 83.0 2.8944   8 307    17.4 382.00
## 227 0.38214  0  6.20    0 0.5040 8.040 86.5 3.2157   8 307    17.4 387.38
##     lstat medv
## 98   4.21 38.7
## 164  3.32 50.0
## 205  2.88 50.0
## 225  4.14 44.8
## 226  4.63 50.0
## 227  3.13 37.6
```

64 suburbs average more than seven rooms per dwelling and 13 suburbs average more than eight rooms per dwelling. For suburbs averagign more than eight rooms per dwelling, it appears that age is significantly higher than average

Chapter 3, Question 15

a)

```
removeCrim = Boston[, names(Boston) != "crim"]

for (i in seq_along(removeCrim)){
  assign(paste(names(removeCrim)[i], "Regression", sep=""), lm(crim ~ as.matrix(removeCrim[i]), data = B
  print(names(removeCrim)[i])
  print(summary(get(paste(names(removeCrim)[i], "Regression", sep=""))))
}
```

```
## [1] "zn"
##
## Call:
## lm(formula = crim ~ as.matrix(removeCrim[i]), 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 ***
## as.matrix(removeCrim[i]) -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
##
## [1] "indus"
##
## Call:
## lm(formula = crim ~ as.matrix(removeCrim[i]), 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 **
## as.matrix(removeCrim[i])  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
##
## [1] "chas"
##
```

```
## Call:
## lm(formula = crim ~ as.matrix(removeCrim[i]), 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 ***
## as.matrix(removeCrim[i])1  -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
##
## [1] "nox"
##
## Call:
## lm(formula = crim ~ as.matrix(removeCrim[i]), 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 ***
## as.matrix(removeCrim[i])   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
##
## [1] "rm"
##
## Call:
## lm(formula = crim ~ as.matrix(removeCrim[i]), 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 ***
## as.matrix(removeCrim[i])   -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
##
## [1] "age"
##
## Call:
## lm(formula = crim ~ as.matrix(removeCrim[i]), 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 ***
## as.matrix(removeCrim[i])  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
##
## [1] "dis"
##
## Call:
## lm(formula = crim ~ as.matrix(removeCrim[i]), 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 ***
## as.matrix(removeCrim[i])  -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
##
## [1] "rad"
##
## Call:
## lm(formula = crim ~ as.matrix(removeCrim[i]), data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -10.164  -1.381  -0.141   0.660  76.433
##
## Coefficients:
```

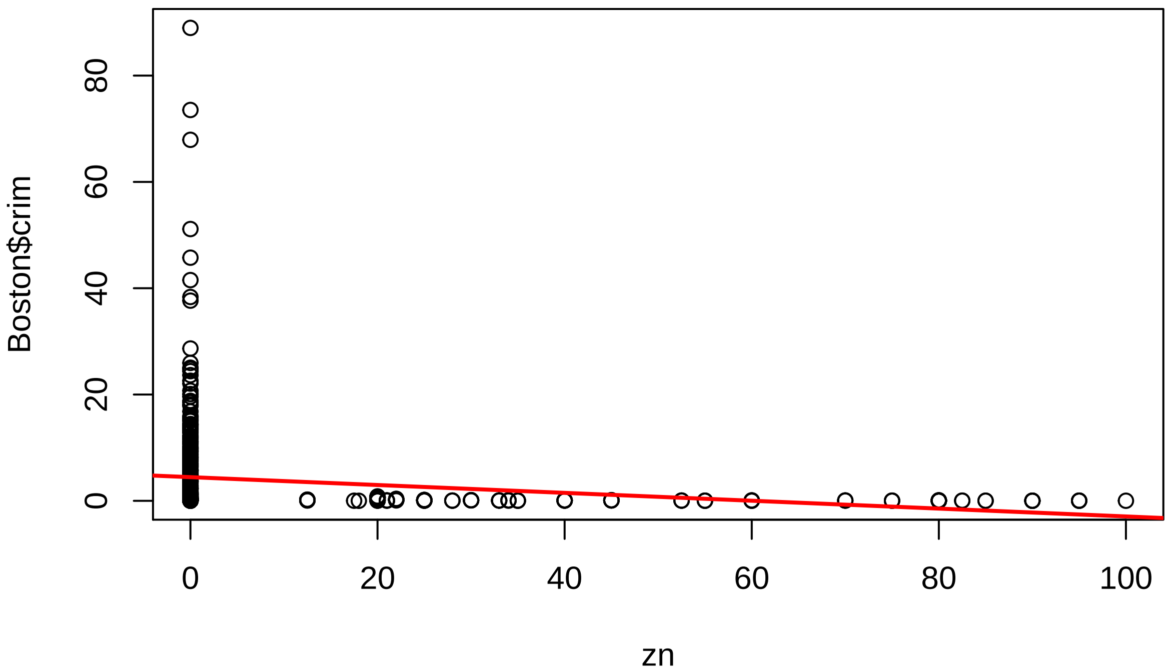
```
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)              -2.28716    0.44348  -5.157 3.61e-07 ***
## as.matrix(removeCrim[i])  0.61791    0.03433  17.998  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard 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
##
## [1] "tax"
##
## Call:
## lm(formula = crim ~ as.matrix(removeCrim[i]), 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 ***
## as.matrix(removeCrim[i])  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
##
## [1] "ptratio"
##
## Call:
## lm(formula = crim ~ as.matrix(removeCrim[i]), 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 ***
## as.matrix(removeCrim[i])   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
##
## [1] "black"
##
## Call:
## lm(formula = crim ~ as.matrix(removeCrim[i]), data = Boston)
```

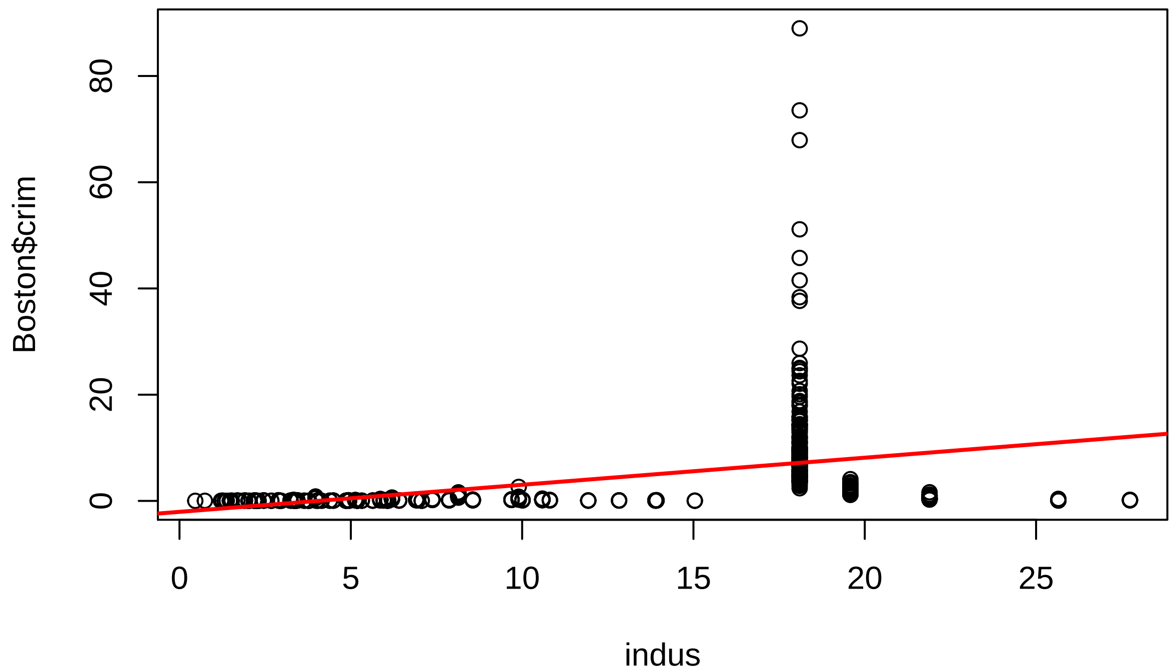
```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -13.756  -2.299  -2.095  -1.296  86.822
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)              16.553529   1.425903  11.609   <2e-16 ***
## as.matrix(removeCrim[i]) -0.036280   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
##
## [1] "lstat"
##
## Call:
## lm(formula = crim ~ as.matrix(removeCrim[i]), 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 ***
## as.matrix(removeCrim[i])  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
##
## [1] "medv"
##
## Call:
## lm(formula = crim ~ as.matrix(removeCrim[i]), 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 ***
## as.matrix(removeCrim[i]) -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
```

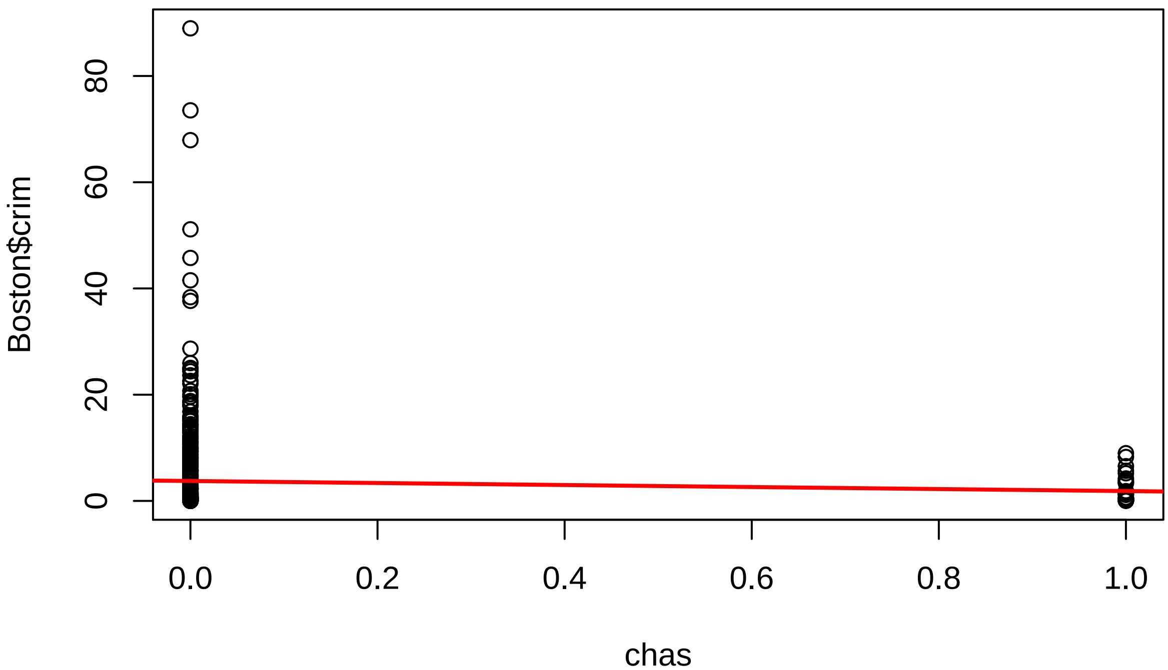
Each predictor except chas is significant at the alpha = .05 significance level. Below are plots for each regression:

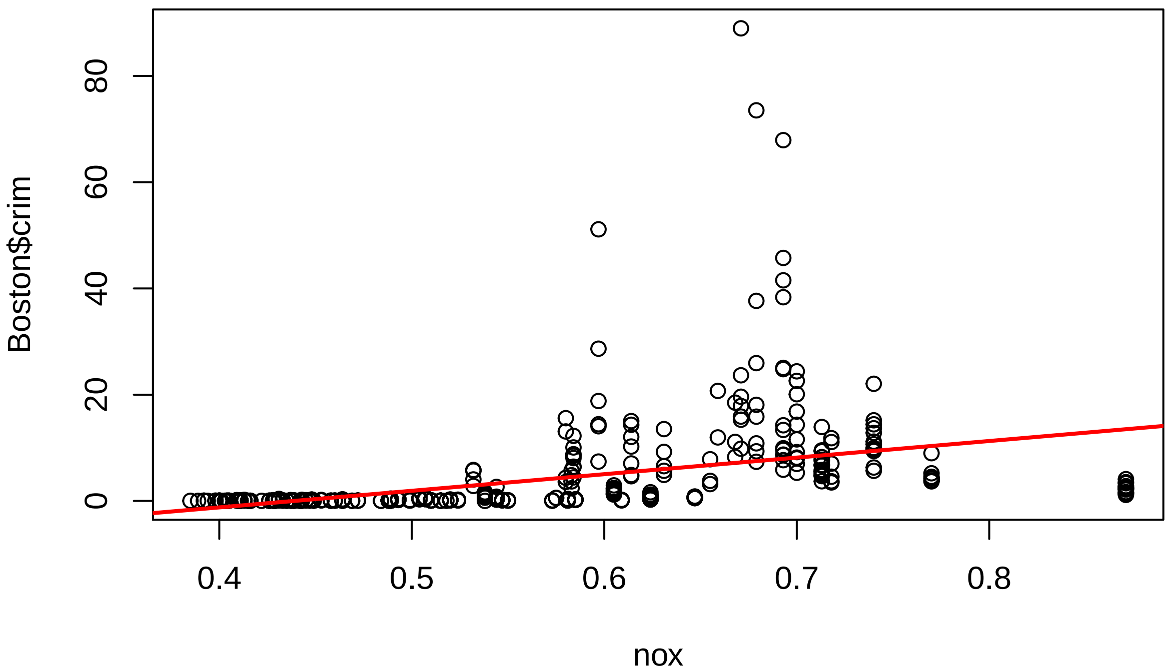
```r
for (i in seq_along(Boston[-ncol(Boston)])){
  plot(Boston$crim ~ as.matrix(removeCrim[i]), xlab=names(removeCrim)[i])
  abline(lm(crim ~ as.matrix(removeCrim[i]), data = Boston), col="red", lwd = 2)
}
```

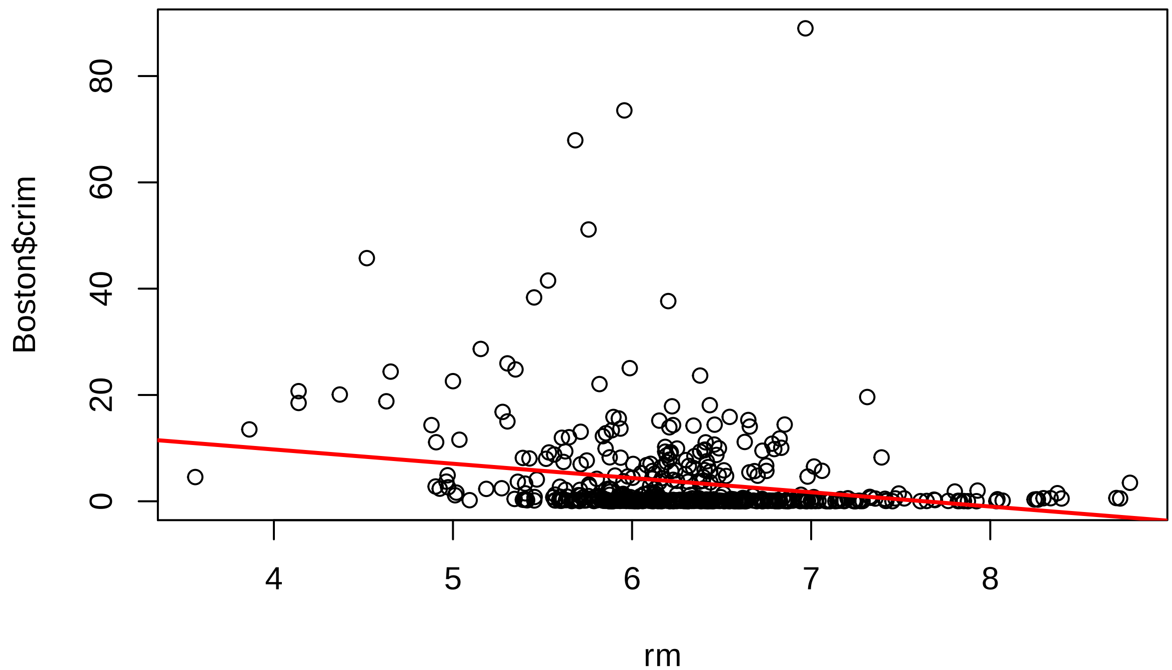


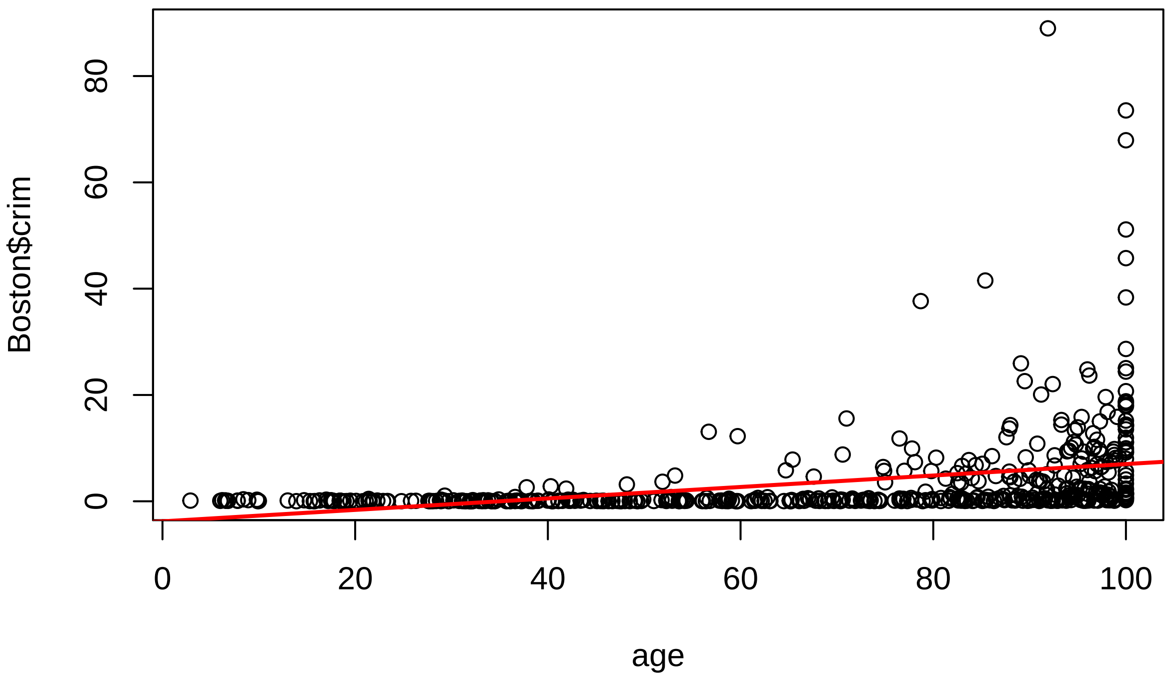


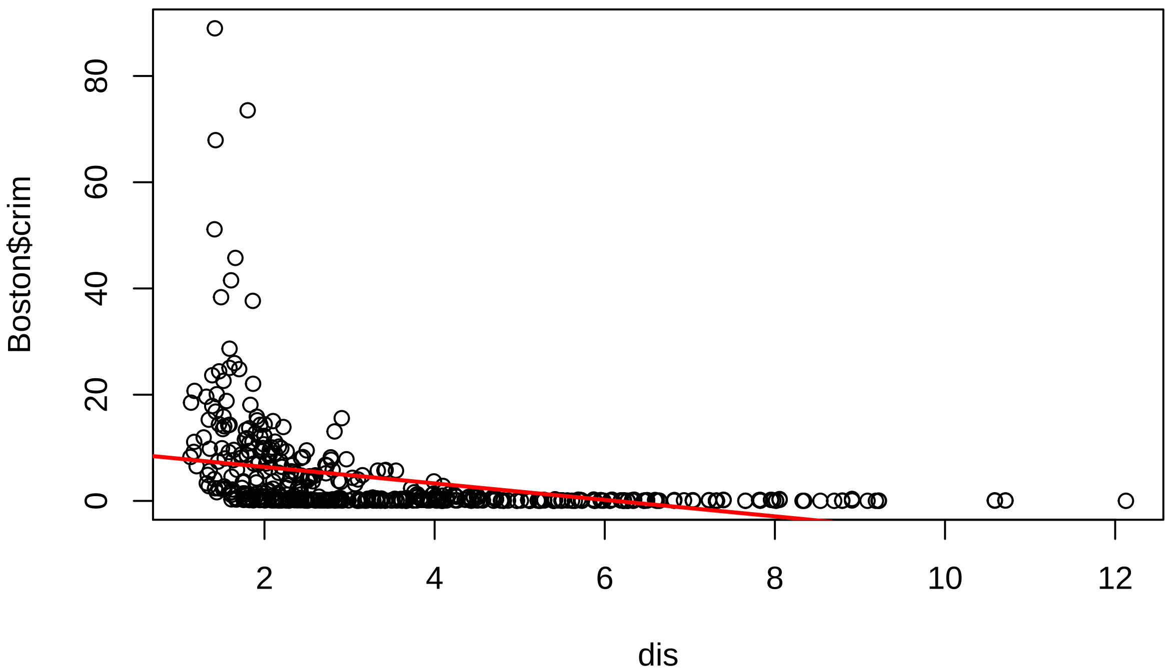
Boston$crim
0
20
40
60
80
0
5
10
15
20
25
indus


Boston$crim
0
20
40
60
80
0.0
0.2
0.4
0.6
0.8
1.0
chas

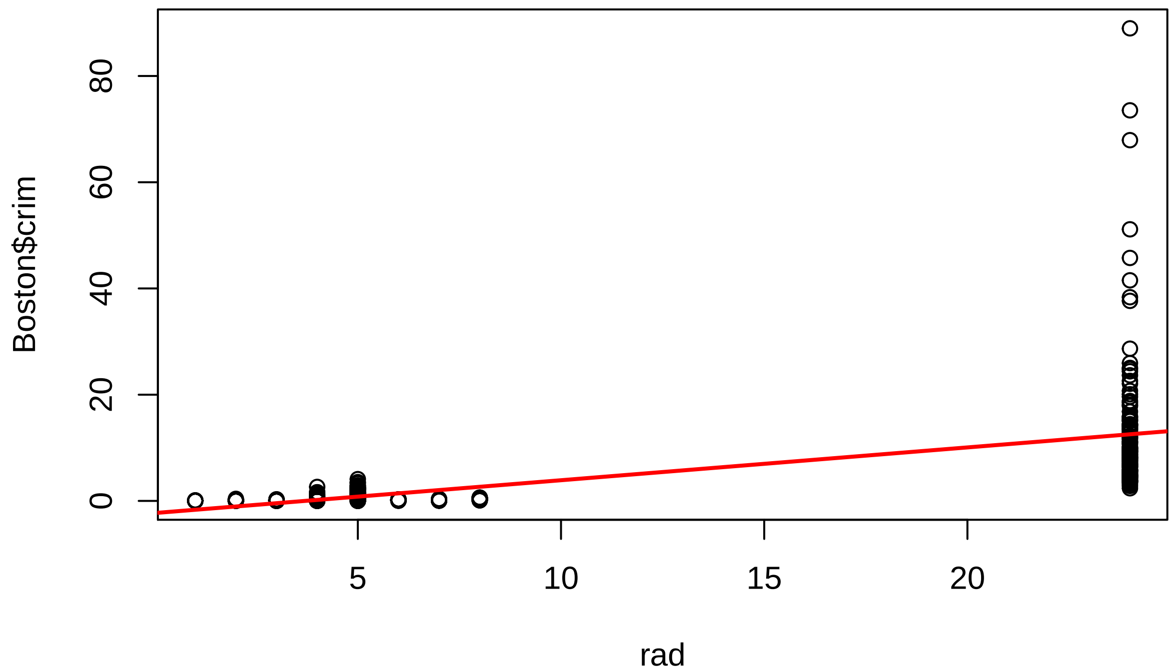

Boston$crim
0
20
40
60
80
0.4
0.5
0.6
0.7
0.8
nox

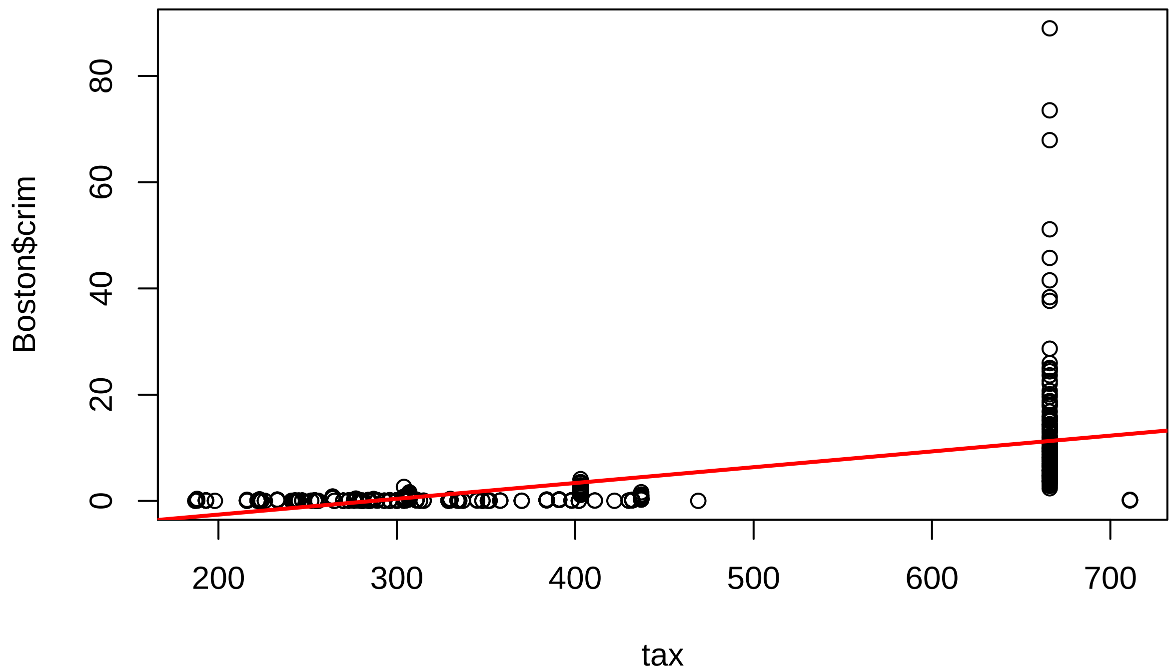

Boston$crim
0
20
40
60
80
4
5
6
7
8
rm


Boston$crim
0
20
40
60
80
0
20
40
60
80
100
age

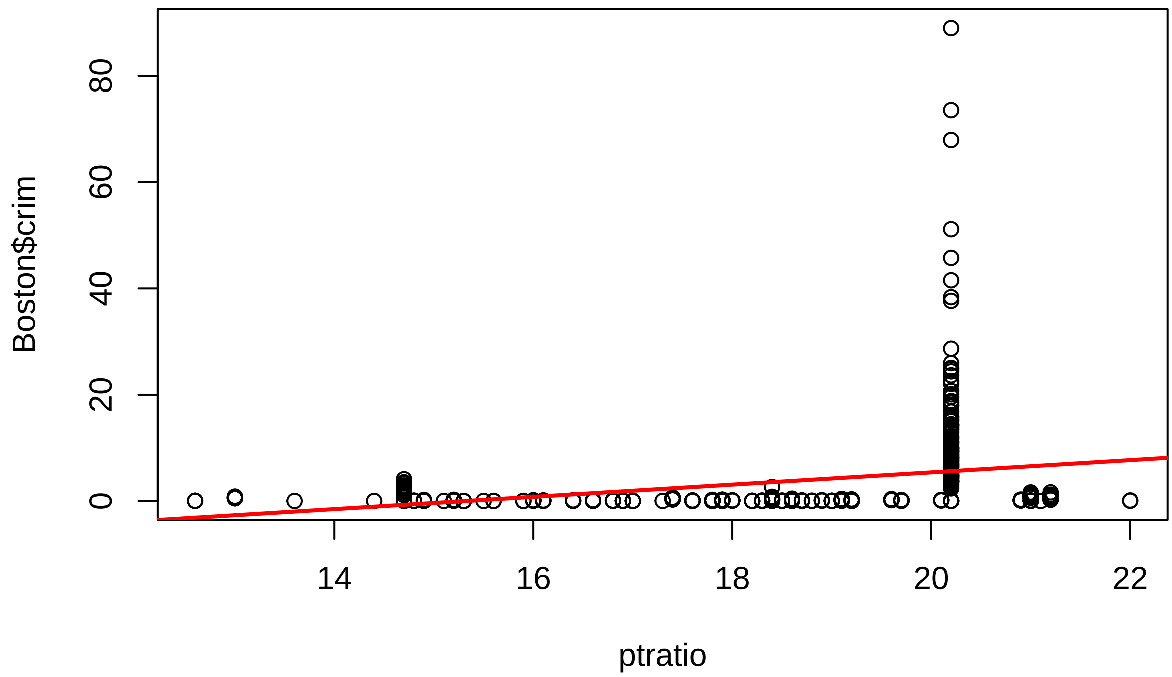


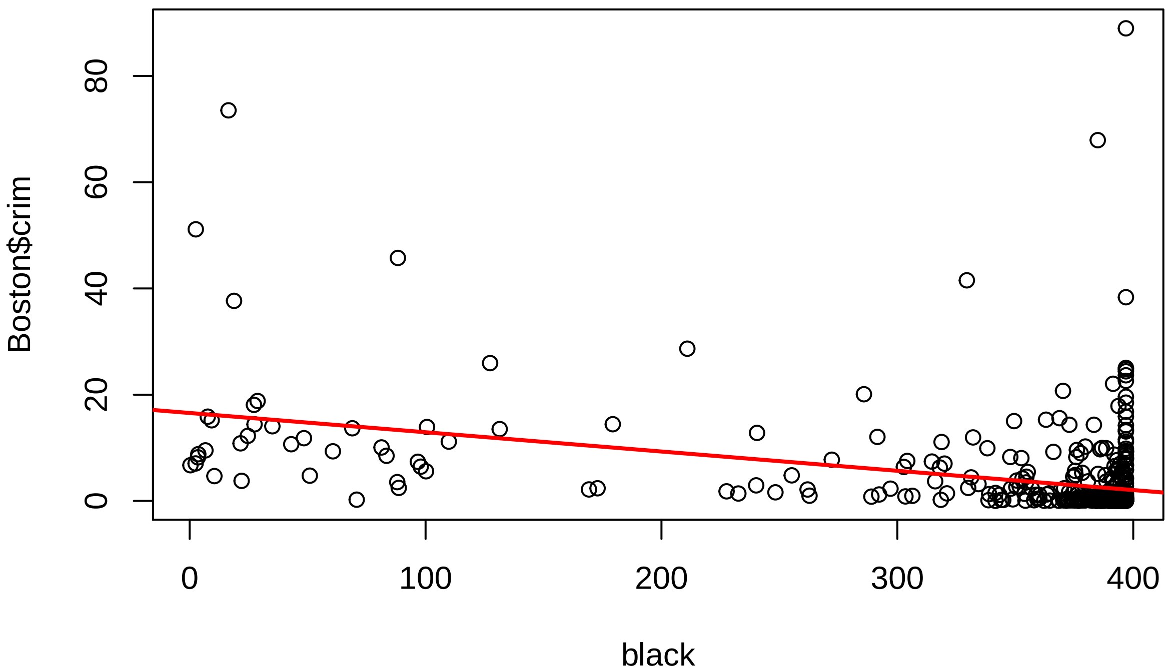
Boston$crim
0
20
40
60
80
2
4
6
8
10
12
dis


Boston$crim
rad
0
20
40
60
80
5
10
15
20

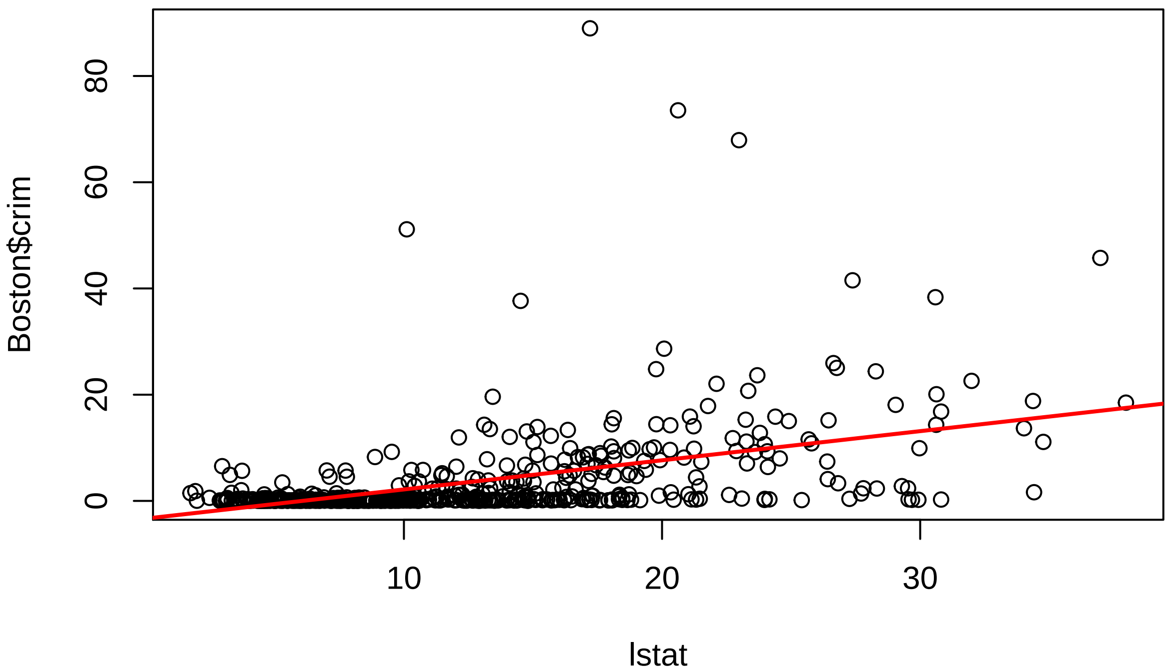


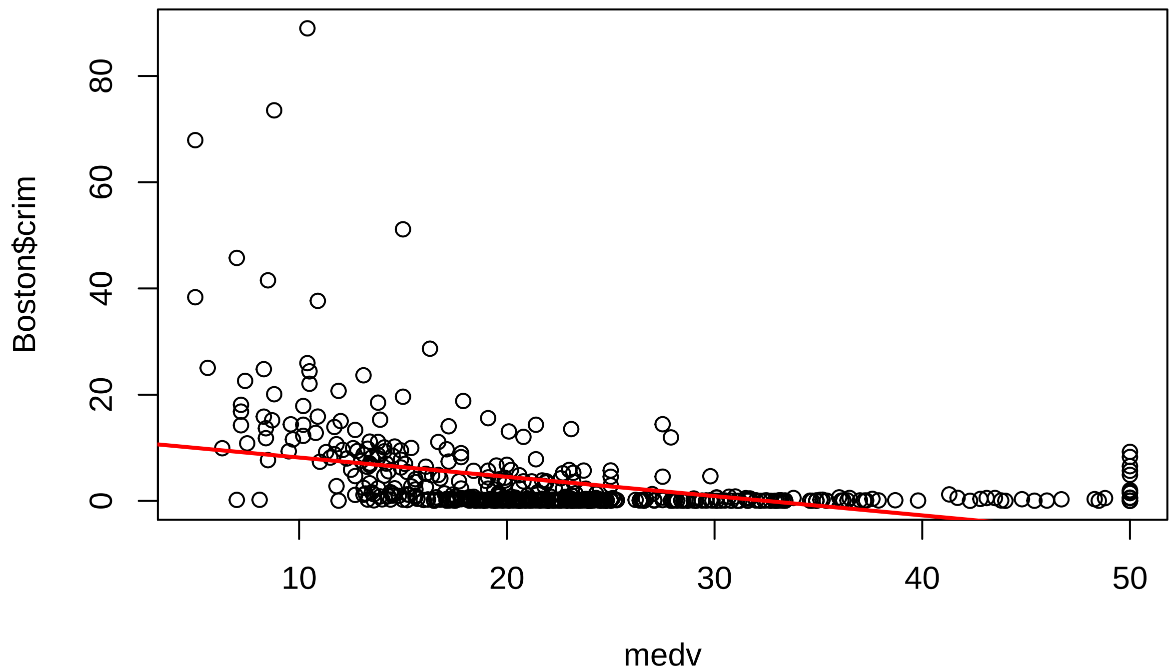
Boston$crim
0
20
40
60
80
200
300
400
500
600
700
tax


Boston$crim
0
20
40
60
80
14
16
18
20
22
ptratio



Boston$crim
0
20
40
60
80
0
100
200
300
400
black


Boston$crim
0
20
40
60
80
10
20
30
lstat



b)

At the alapha = .05 level, We reject the null hypothesis for "zn", "dis", "rad", "black" and "medv".

```
allVar = lm(crim ~ ., data = Boston)
summary(allVar)
```

```
##
## 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
## chas1        -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
```

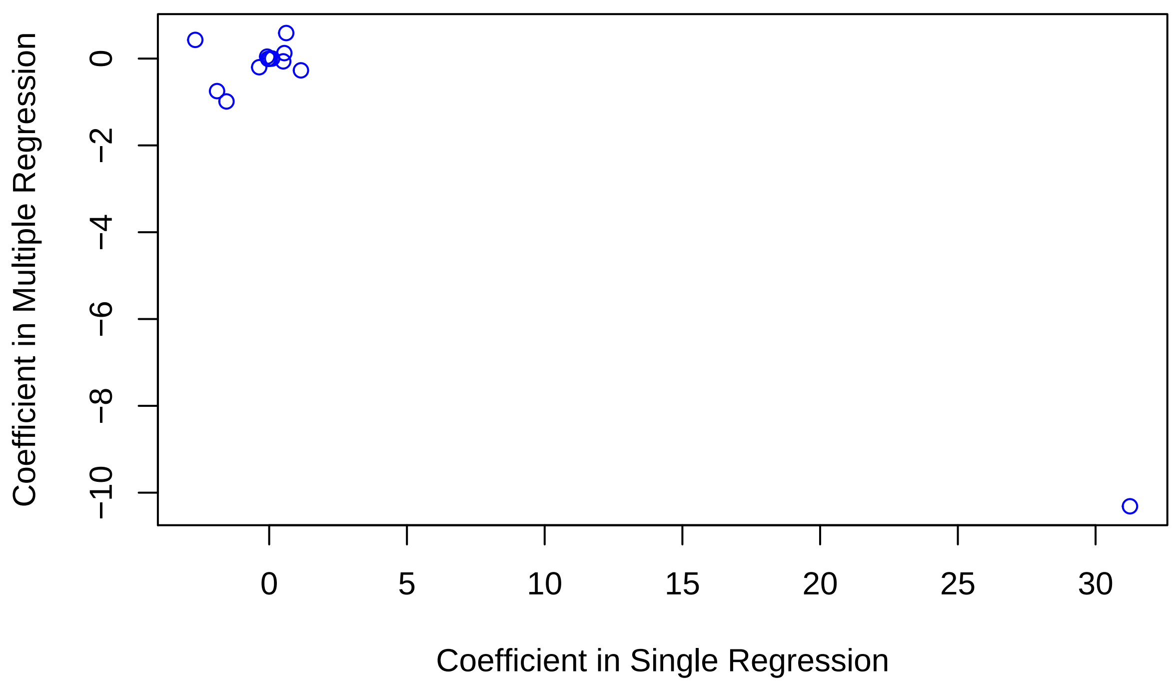
```
## ptratio      -0.271081   0.186450  -1.454 0.146611
## black        -0.007538   0.003673  -2.052 0.040702 *
## lstat         0.126211   0.075725   1.667 0.096208 .
## medv         -0.198887   0.060516  -3.287 0.001087 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## 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
```

c)

```r
linReg <- vector("numeric", 0)

for (i in seq_along(Boston[-ncol(Boston)])){
  linReg <- c(linReg, get(paste(names(removeCrim)[i], "Regression", sep=""))$coefficient[2])

}
```

```r
multReg <- c(allVar$coefficients[-1])
```

```r
combined = cbind(linReg, multReg)


plot(linReg, multReg, xlab = "Coefficient in Single Regression", ylab = "Coefficient in Multiple Regres
```



While there is a expected small variation amonst most of the coefficients between each single regression and the corresponding coefficient in the multiple regression, the variable "nox" has a much more significant difference, suggesting that its significance as a singular predictor is likely due to correlation with another relevant predictor that is mitigated in the case of multiple regression.

d)

It appears that cubic models fit "indus", "nox", "age", "dis", "ptratio" and "medv" as a predictor

```
removeChas = removeCrim[, names(removeCrim) != "chas"]

for (i in seq_along(removeChas)){
  print(names(removeChas)[i])
  print(summary(lm(crim ~ poly(as.matrix(removeChas[i]), 3), data = Boston)))
}
```

```
## [1] "zn"
##
## Call:
## lm(formula = crim ~ poly(as.matrix(removeChas[i]), 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(as.matrix(removeChas[i]), 3)1 -38.7498     8.3722  -4.628  4.7e-06
## poly(as.matrix(removeChas[i]), 3)2  23.9398     8.3722   2.859  0.00442
## poly(as.matrix(removeChas[i]), 3)3 -10.0719     8.3722  -1.203  0.22954
##
## (Intercept)                        ***
## poly(as.matrix(removeChas[i]), 3)1 ***
## poly(as.matrix(removeChas[i]), 3)2 **
## poly(as.matrix(removeChas[i]), 3)3
## ---
## 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
##
## [1] "indus"
##
## Call:
## lm(formula = crim ~ poly(as.matrix(removeChas[i]), 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(as.matrix(removeChas[i]), 3)1   78.591      7.423  10.587  < 2e-16
## poly(as.matrix(removeChas[i]), 3)2  -24.395      7.423  -3.286  0.00109
## poly(as.matrix(removeChas[i]), 3)3  -54.130      7.423  -7.292  1.2e-12
##
## (Intercept)                        ***
## poly(as.matrix(removeChas[i]), 3)1 ***
## poly(as.matrix(removeChas[i]), 3)2 **
## poly(as.matrix(removeChas[i]), 3)3 ***
## ---
## 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
##
## [1] "nox"
##
## Call:
## lm(formula = crim ~ poly(as.matrix(removeChas[i]), 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(as.matrix(removeChas[i]), 3)1  81.3720     7.2336  11.249  < 2e-16
## poly(as.matrix(removeChas[i]), 3)2 -28.8286     7.2336  -3.985 7.74e-05
## poly(as.matrix(removeChas[i]), 3)3 -60.3619     7.2336  -8.345 6.96e-16
##
## (Intercept)                        ***
## poly(as.matrix(removeChas[i]), 3)1 ***
## poly(as.matrix(removeChas[i]), 3)2 ***
## poly(as.matrix(removeChas[i]), 3)3 ***
## ---
## 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
##
## [1] "rm"
##
## Call:
## lm(formula = crim ~ poly(as.matrix(removeChas[i]), 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(as.matrix(removeChas[i]), 3)1 -42.3794     8.3297  -5.088 5.13e-07
## poly(as.matrix(removeChas[i]), 3)2  26.5768     8.3297   3.191  0.00151
## poly(as.matrix(removeChas[i]), 3)3  -5.5103     8.3297  -0.662  0.50858
##
## (Intercept)                        ***
## poly(as.matrix(removeChas[i]), 3)1 ***
## poly(as.matrix(removeChas[i]), 3)2 **
## poly(as.matrix(removeChas[i]), 3)3
## ---
## 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
##
## [1] "age"
##
## Call:
## lm(formula = crim ~ poly(as.matrix(removeChas[i]), 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(as.matrix(removeChas[i]), 3)1  68.1820     7.8397   8.697  < 2e-16
## poly(as.matrix(removeChas[i]), 3)2  37.4845     7.8397   4.781 2.29e-06
## poly(as.matrix(removeChas[i]), 3)3  21.3532     7.8397   2.724  0.00668
##
## (Intercept)                        ***
## poly(as.matrix(removeChas[i]), 3)1 ***
## poly(as.matrix(removeChas[i]), 3)2 ***
## poly(as.matrix(removeChas[i]), 3)3 **
## ---
## 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
##
## [1] "dis"
##
## Call:
## lm(formula = crim ~ poly(as.matrix(removeChas[i]), 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(as.matrix(removeChas[i]), 3)1 -73.3886     7.3315 -10.010  < 2e-16
## poly(as.matrix(removeChas[i]), 3)2  56.3730     7.3315   7.689 7.87e-14
## poly(as.matrix(removeChas[i]), 3)3 -42.6219     7.3315  -5.814 1.09e-08
##
## (Intercept)                        ***
## poly(as.matrix(removeChas[i]), 3)1 ***
## poly(as.matrix(removeChas[i]), 3)2 ***
## poly(as.matrix(removeChas[i]), 3)3 ***
## ---
## 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
##
## [1] "rad"
##
## Call:
## lm(formula = crim ~ poly(as.matrix(removeChas[i]), 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(as.matrix(removeChas[i]), 3)1 120.9074     6.6824  18.093  < 2e-16
## poly(as.matrix(removeChas[i]), 3)2  17.4923     6.6824   2.618  0.00912
## poly(as.matrix(removeChas[i]), 3)3   4.6985     6.6824   0.703  0.48231
##
## (Intercept)                        ***
## poly(as.matrix(removeChas[i]), 3)1 ***
## poly(as.matrix(removeChas[i]), 3)2 **
## poly(as.matrix(removeChas[i]), 3)3
## ---
## 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
##
## [1] "tax"
##
## Call:
## lm(formula = crim ~ poly(as.matrix(removeChas[i]), 3), data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -13.273  -1.389   0.046   0.536  76.950
##
## Coefficients:
##                                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)                          3.6135     0.3047  11.860  < 2e-16
```

```
## poly(as.matrix(removeChas[i]), 3)1 112.6458     6.8537  16.436  < 2e-16
## poly(as.matrix(removeChas[i]), 3)2  32.0873     6.8537   4.682 3.67e-06
## poly(as.matrix(removeChas[i]), 3)3  -7.9968     6.8537  -1.167    0.244
##
## (Intercept)                        ***
## poly(as.matrix(removeChas[i]), 3)1 ***
## poly(as.matrix(removeChas[i]), 3)2 ***
## poly(as.matrix(removeChas[i]), 3)3
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.854 on 502 degrees of freedom
## Multiple R-squared:  0.3689, Adjusted R-squared:  0.3651
## F-statistic:  97.8 on 3 and 502 DF,  p-value: < 2.2e-16
##
## [1] "ptratio"
##
## Call:
## lm(formula = crim ~ poly(as.matrix(removeChas[i]), 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(as.matrix(removeChas[i]), 3)1   56.045      8.122   6.901 1.57e-11
## poly(as.matrix(removeChas[i]), 3)2   24.775      8.122   3.050  0.00241
## poly(as.matrix(removeChas[i]), 3)3  -22.280      8.122  -2.743  0.00630
##
## (Intercept)                        ***
## poly(as.matrix(removeChas[i]), 3)1 ***
## poly(as.matrix(removeChas[i]), 3)2 **
## poly(as.matrix(removeChas[i]), 3)3 **
## ---
## 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
##
## [1] "black"
##
## Call:
## lm(formula = crim ~ poly(as.matrix(removeChas[i]), 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(as.matrix(removeChas[i]), 3)1 -74.4312     7.9546  -9.357   <2e-16
## poly(as.matrix(removeChas[i]), 3)2   5.9264     7.9546   0.745    0.457
## poly(as.matrix(removeChas[i]), 3)3  -4.8346     7.9546  -0.608    0.544
##
## (Intercept)                        ***
## poly(as.matrix(removeChas[i]), 3)1 ***
## poly(as.matrix(removeChas[i]), 3)2
## poly(as.matrix(removeChas[i]), 3)3
## ---
## 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
##
## [1] "lstat"
##
## Call:
## lm(formula = crim ~ poly(as.matrix(removeChas[i]), 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(as.matrix(removeChas[i]), 3)1  88.0697     7.6294  11.543   <2e-16
## poly(as.matrix(removeChas[i]), 3)2  15.8882     7.6294   2.082   0.0378
## poly(as.matrix(removeChas[i]), 3)3 -11.5740     7.6294  -1.517   0.1299
##
## (Intercept)                        ***
## poly(as.matrix(removeChas[i]), 3)1 ***
## poly(as.matrix(removeChas[i]), 3)2 *
## poly(as.matrix(removeChas[i]), 3)3
## ---
## 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
##
## [1] "medv"
##
## Call:
## lm(formula = crim ~ poly(as.matrix(removeChas[i]), 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(as.matrix(removeChas[i]), 3)1  -75.058      6.569 -11.426  < 2e-16
## poly(as.matrix(removeChas[i]), 3)2   88.086      6.569  13.409  < 2e-16
## poly(as.matrix(removeChas[i]), 3)3  -48.033      6.569  -7.312 1.05e-12
##
## (Intercept)                        ***
## poly(as.matrix(removeChas[i]), 3)1 ***
## poly(as.matrix(removeChas[i]), 3)2 ***
## poly(as.matrix(removeChas[i]), 3)3 ***
## ---
## 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
```

Chapter 6, Number 9

a)

```r
library(ISLR)
train = sample(1:dim(College)[1], dim(College)[1]/2)
test <- -train
trainData <- College[train, ]
testData <- College[test, ]
```

b)

```r
trainRegression = lm(Apps ~ ., data = trainData)
testPredictions = predict(trainRegression, testData)
cat("Least Squares Test MSE =", mean((testPredictions - testData$Apps)^2))
```

```
## Least Squares Test MSE = 1504984
```

c)

```r
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-18
```

```r
trainMatrix <- model.matrix(Apps ~ ., data = trainData)
testMatrix <- model.matrix(Apps ~ ., data = testData)
grid <- 10 ^ seq(4, -2, length = 100)
ridgeRegression <- glmnet(trainMatrix, trainData$Apps, alpha = 0, lambda = grid, thresh = 1e-12)
ridgeCV <- cv.glmnet(trainMatrix, trainData$Apps, alpha = 0, lambda = grid, thresh = 1e-12)
ridgeLambdaMin = ridgeCV$lambda.min
ridgeTestPredictions = predict(ridgeRegression, s = ridgeLambdaMin, newx = testMatrix)
cat("Ridge Test MSE = ", mean((ridgeTestPredictions - testData$Apps)^2))
```

```
## Ridge Test MSE =  1582955
```

d)

```r
lasso <- glmnet(trainMatrix, trainData$Apps, alpha = 1, lambda = grid, thresh = 1e-12)
lassoCV <- cv.glmnet(trainMatrix, trainData$Apps, alpha = 1, lambda = grid, thresh = 1e-12)
lassoLambdaMin <- lassoCV$lambda.min
lassoTestPredictions <- predict(lasso, s = lassoLambdaMin, newx = testMatrix)
cat("Lasso Test MSE = ", mean((lassoTestPredictions - testData$Apps)^2), "\n\n")
```

```
## Lasso Test MSE =  1551055
```

```r
predict(lasso, s = lassoLambdaMin, type = "coefficients")
```

```
## 19 x 1 sparse Matrix of class "dgCMatrix"
##                         1
## (Intercept) -887.11438450
## (Intercept)    .
## PrivateYes  -593.95291558
## Accept         1.28617024
## Enroll        -0.22971045
## Top10perc     32.51418490
## Top25perc     -2.82376948
## F.Undergrad    0.05426733
## P.Undergrad    .
## Outstate      -0.03126398
## Room.Board     0.20435982
## Books          0.14995119
## Personal       0.08010923
## PhD           -9.26101719
## Terminal      -4.86458586
## S.F.Ratio     25.10466029
## perc.alumni   -4.80865810
## Expend         0.07504836
## Grad.Rate      3.81148101
```

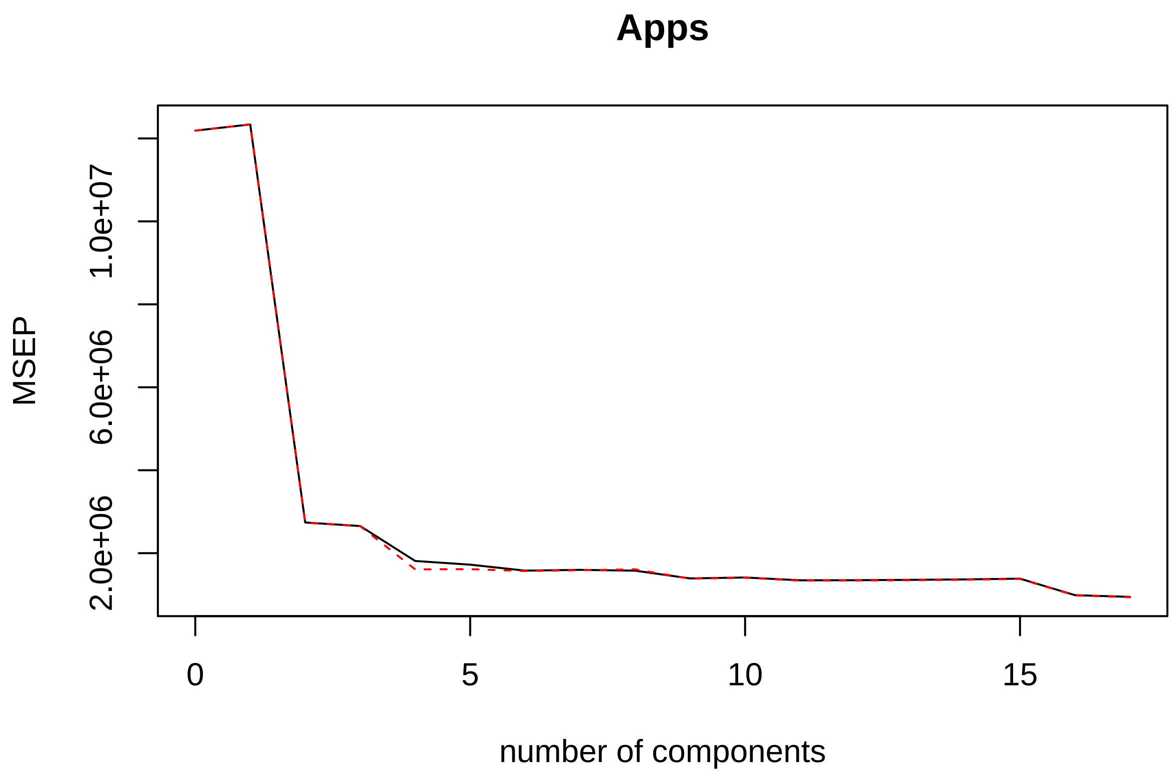
Note that the above listing of coefficients appears to be very dependant upon the training and test set selection, indicating that these may not be extremely reliable results.

e)

```r
library(pls)
```

```
##
## Attaching package: 'pls'

## The following object is masked from 'package:corrplot':
##
##     corrplot

## The following object is masked from 'package:stats':
##
##     loadings
```

```r
PCR <- pcr(Apps ~ ., data = trainData, scale = TRUE, validation = "CV")
validationplot(PCR, val.type = "MSEP")
```

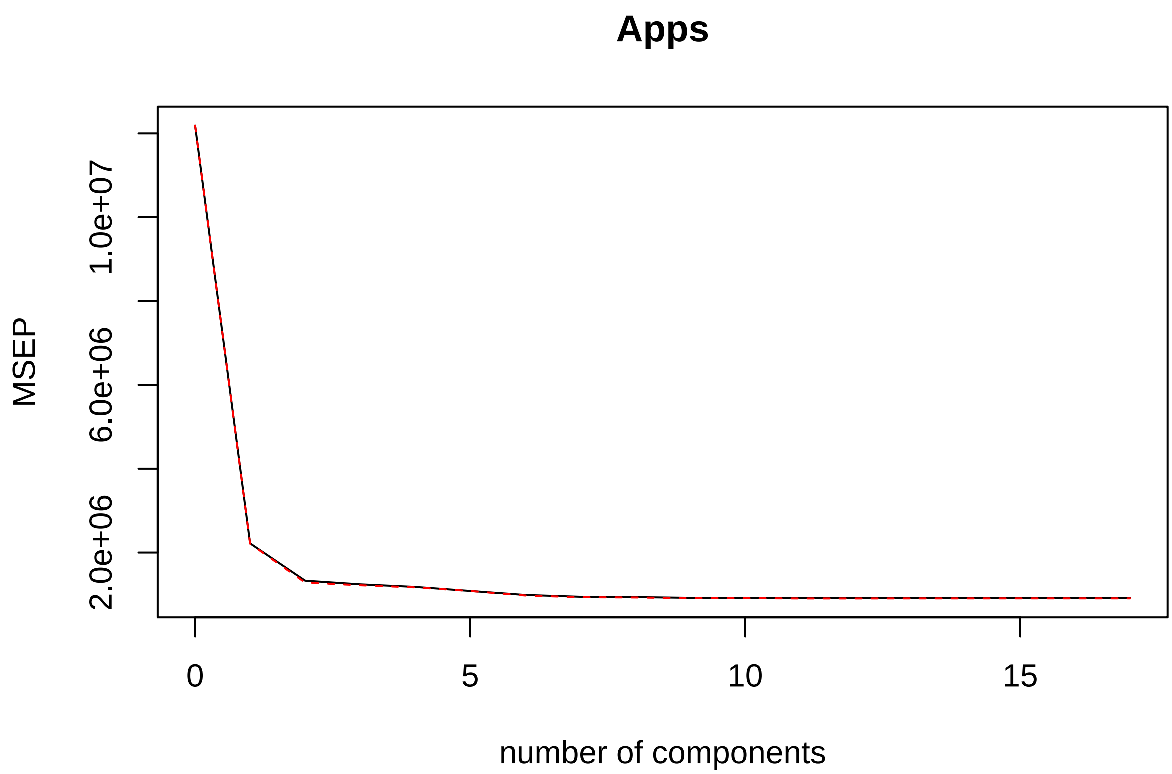



```
for (i in 1:17){
  assign(paste("pcrPrediction", i, sep = ""), predict(PCR, testData, ncomp = i))
  cat("PCR Test MSE for", i, "predictors =", mean((get(paste("pcrPrediction", i, sep = "")) - testData$
}
```

```
## PCR Test MSE for 1 predictors = 17794875
## PCR Test MSE for 2 predictors = 5704108
## PCR Test MSE for 3 predictors = 5717555
## PCR Test MSE for 4 predictors = 3368130
## PCR Test MSE for 5 predictors = 3350844
## PCR Test MSE for 6 predictors = 3338283
## PCR Test MSE for 7 predictors = 3336918
## PCR Test MSE for 8 predictors = 3273326
## PCR Test MSE for 9 predictors = 3146857
## PCR Test MSE for 10 predictors = 3136997
## PCR Test MSE for 11 predictors = 3169565
## PCR Test MSE for 12 predictors = 3155873
## PCR Test MSE for 13 predictors = 3160750
## PCR Test MSE for 14 predictors = 3165793
## PCR Test MSE for 15 predictors = 3149565
## PCR Test MSE for 16 predictors = 1621920
## PCR Test MSE for 17 predictors = 1504984
```

f)

```r
PLS <- plsr(Apps ~ ., data = trainData, scale = TRUE, validation = "CV")
validationplot(PLS, val.type = "MSEP")
```




```r
for (i in 1:17){
  assign(paste("plsPrediction", i, sep = ""), predict(PLS, testData, ncomp = i))
  cat("PLS Test MSE for", i, "predictors =", mean((get(paste("plsPrediction", i, sep = "")) - testData$A
}
```

```
## PLS Test MSE for 1 predictors = 4813844
## PLS Test MSE for 2 predictors = 2860877
## PLS Test MSE for 3 predictors = 2759902
## PLS Test MSE for 4 predictors = 2546673
## PLS Test MSE for 5 predictors = 2210918
## PLS Test MSE for 6 predictors = 1625748
## PLS Test MSE for 7 predictors = 1501600
## PLS Test MSE for 8 predictors = 1486517
## PLS Test MSE for 9 predictors = 1491456
## PLS Test MSE for 10 predictors = 1497336
## PLS Test MSE for 11 predictors = 1504354
## PLS Test MSE for 12 predictors = 1504373
## PLS Test MSE for 13 predictors = 1505194
## PLS Test MSE for 14 predictors = 1505035
## PLS Test MSE for 15 predictors = 1504989
## PLS Test MSE for 16 predictors = 1505049
## PLS Test MSE for 17 predictors = 1504984
```

g)

```
testMean = mean(testData$Apps)
testSStot = mean((testMean - testData$Apps)^2)

linearR2 <- 1 - mean((testPredictions - testData$Apps)^2) / testSStot
ridgeR2 <- 1 - mean((ridgeTestPredictions - testData$Apps)^2) / testSStot
lassoR2 <- 1 - mean((lassoTestPredictions - testData$Apps)^2) / testSStot

cat("Linear R^2 =", linearR2, "\n")
```

```
## Linear R^2 = 0.9153726
```

```
cat("Ridge R^2 =", ridgeR2, "\n")
```

```
## Ridge R^2 = 0.9109882
```

```
cat("Lasso R^2 =", lassoR2, "\n\n")
```

```
## Lasso R^2 = 0.912782
```

```
for (i in 1:17){
  cat("PCR R^2 for", i, "predictors =", 1 - mean((get(paste("pcrPrediction", i, sep = "")) - testData$Ap
}
```

```
## PCR R^2 for 1 predictors = -0.0006308662
## PCR R^2 for 2 predictors = 0.67925
## PCR R^2 for 3 predictors = 0.6784938
## PCR R^2 for 4 predictors = 0.8106053
## PCR R^2 for 5 predictors = 0.8115773
## PCR R^2 for 6 predictors = 0.8122836
## PCR R^2 for 7 predictors = 0.8123604
## PCR R^2 for 8 predictors = 0.8159363
## PCR R^2 for 9 predictors = 0.8230478
## PCR R^2 for 10 predictors = 0.8236023
## PCR R^2 for 11 predictors = 0.8217709
## PCR R^2 for 12 predictors = 0.8225408
## PCR R^2 for 13 predictors = 0.8222666
## PCR R^2 for 14 predictors = 0.821983
## PCR R^2 for 15 predictors = 0.8228955
## PCR R^2 for 16 predictors = 0.9087971
## PCR R^2 for 17 predictors = 0.9153726
```

```
cat("\n")
```

```
for (i in 1:17){
  cat("PLS R^2 for", i, "predictors =", 1 - mean((get(paste("plsPrediction", i, sep = "")) - testData$Ap
}
```

```
## PLS R^2 for 1 predictors = 0.7293108
## PLS R^2 for 2 predictors = 0.8391289
## PLS R^2 for 3 predictors = 0.8448068
## PLS R^2 for 4 predictors = 0.856797
## PLS R^2 for 5 predictors = 0.875677
## PLS R^2 for 6 predictors = 0.9085819
## PLS R^2 for 7 predictors = 0.9155629
## PLS R^2 for 8 predictors = 0.9164111
## PLS R^2 for 9 predictors = 0.9161333
```

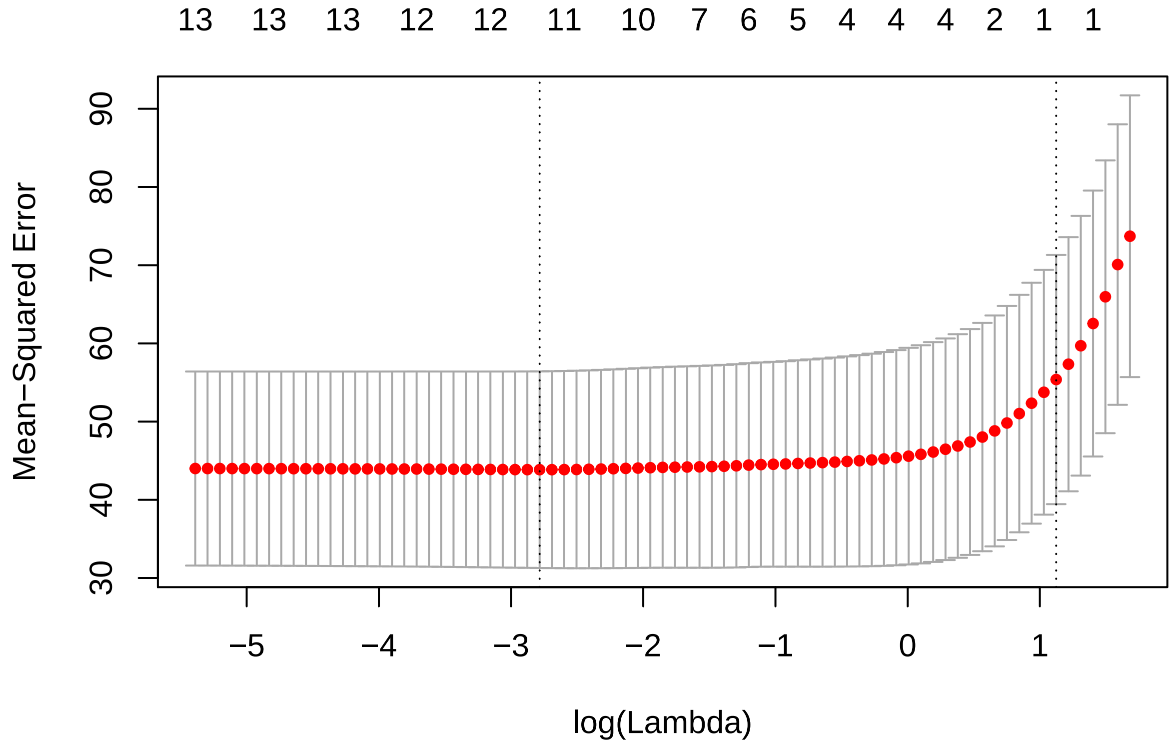
```
## PLS R^2 for 10 predictors = 0.9158027
## PLS R^2 for 11 predictors = 0.9154081
## PLS R^2 for 12 predictors = 0.915407
## PLS R^2 for 13 predictors = 0.9153608
## PLS R^2 for 14 predictors = 0.9153698
## PLS R^2 for 15 predictors = 0.9153723
## PLS R^2 for 16 predictors = 0.915369
## PLS R^2 for 17 predictors = 0.9153726
```

Each of the models seems to predict college applications with a reasonably high R^2 value, with the exception of some PCR models with a low number of predictors.

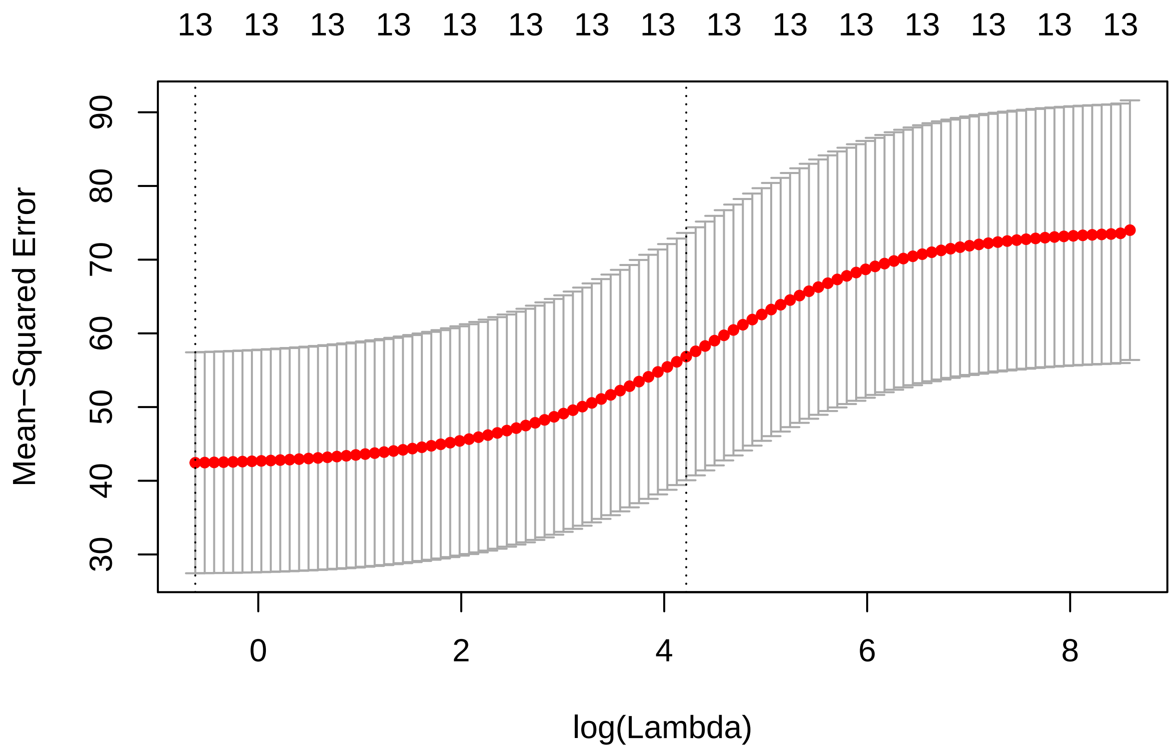
Chapter 6, Number 11

a)

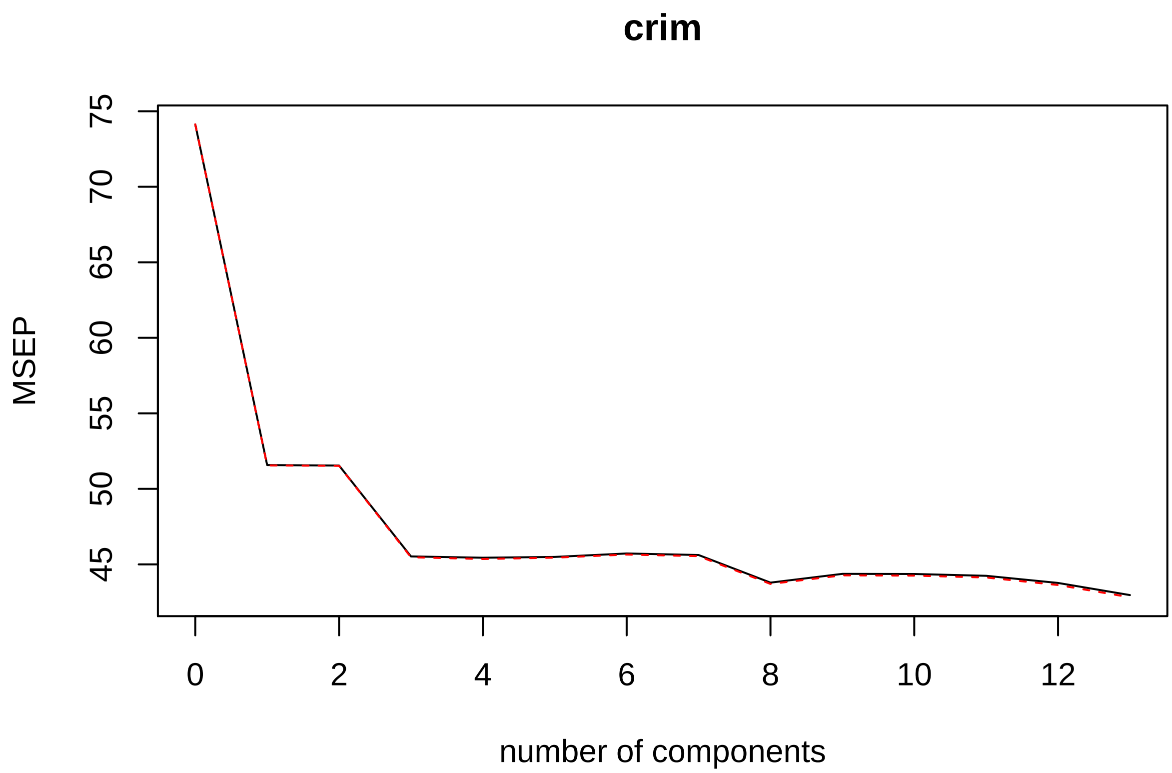
```r
library(MASS)
library(glmnet)
library(pls)

predictors <- model.matrix(crim ~., Boston)[,-1]
response <- Boston$crim

cvLASSO <- cv.glmnet(predictors, response, alpha = 1, type.measure = 'mse')
cvRidge <- cv.glmnet(predictors, response, alpha = 0, type.measure = 'mse')
pcrBoston <- pcr(crim ~ ., data = Boston, scale = TRUE, validation = "CV")

plot(cvLASSO)
```




```
plot(cvRidge)
```




```r
validationplot(pcrBoston, val.type = 'MSEP')
```




```
LASSOmse <- cvLASSO$cvm[cvLASSO$lambda == cvLASSO$lambda.min]
RidgeMse <- cvRidge$cvm[cvRidge$lambda == cvRidge$lambda.min]

cat('LASSO MSE =', LASSOmse, '\n')
```

```
## LASSO MSE = 43.85391
```

```
cat('Ridge MSE =', RidgeMse, '\n\n')
```

```
## Ridge MSE = 42.43959
```

```
for (i in seq_along(pcrBoston$validation$adj)){
  cat('Test MSE of', pcrBoston$validation$adj[i], 'for', i, 'predictors\n')
}
```

```
## Test MSE of 51.20156 for 1 predictors
## Test MSE of 51.06825 for 2 predictors
## Test MSE of 44.8968 for 3 predictors
## Test MSE of 44.67979 for 4 predictors
## Test MSE of 44.64128 for 5 predictors
## Test MSE of 44.47565 for 6 predictors
## Test MSE of 44.27302 for 7 predictors
## Test MSE of 42.56511 for 8 predictors
## Test MSE of 42.52089 for 9 predictors
## Test MSE of 42.36248 for 10 predictors
## Test MSE of 42.16973 for 11 predictors
## Test MSE of 41.38944 for 12 predictors
## Test MSE of 40.45835 for 13 predictors
```

Each of the LASSO, Ridge, and PCR provide models with similiar MSE. The PCR model with all predictors appears to have the lowest MSE, though there is certainly a trade off of potential bias and lack of ability to interpret coefficients.

Chapter 4, Number 10

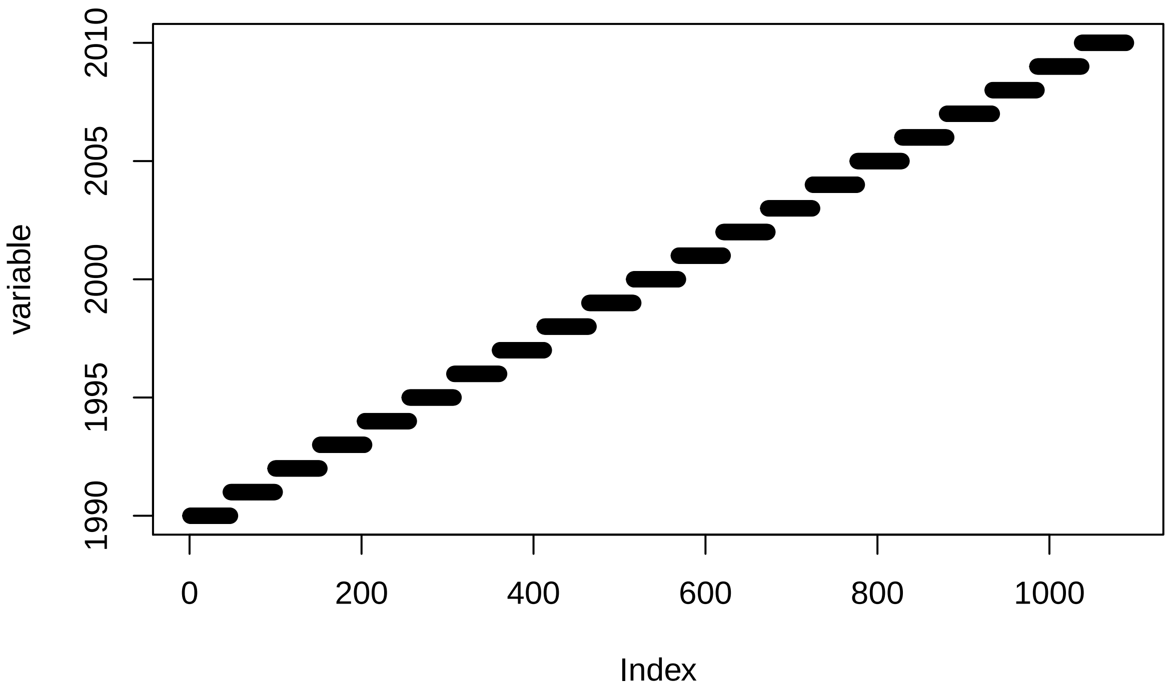
a)

```
cor(Weekly[, -9])
```

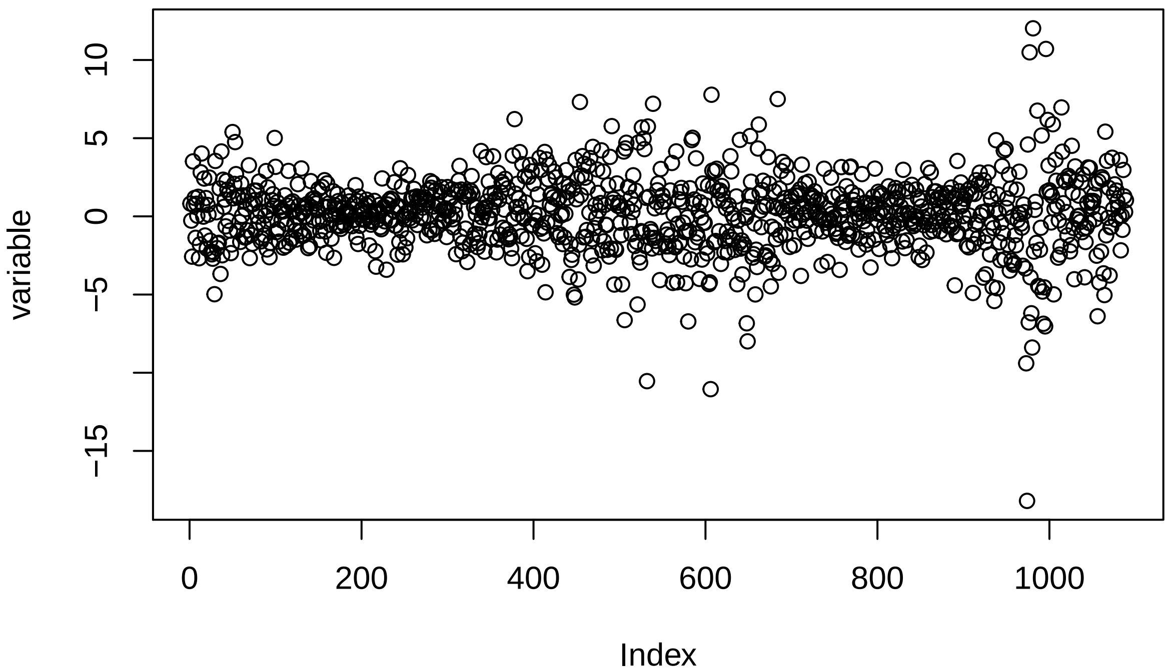
```
##               Year         Lag1        Lag2        Lag3         Lag4
## Year    1.00000000 -0.032289274 -0.03339001 -0.03000649 -0.031127923
## Lag1   -0.03228927  1.000000000 -0.07485305  0.05863568 -0.071273876
## Lag2   -0.03339001 -0.074853051  1.00000000 -0.07572091  0.058381535
## Lag3   -0.03000649  0.058635682 -0.07572091  1.00000000 -0.075395865
## Lag4   -0.03112792 -0.071273876  0.05838153 -0.07539587  1.000000000
## Lag5   -0.03051910 -0.008183096 -0.07249948  0.06065717 -0.075675027
## Volume  0.84194162 -0.064951313 -0.08551314 -0.06928771 -0.061074617
## Today  -0.03245989 -0.075031842  0.05916672 -0.07124364 -0.007825873
##                Lag5      Volume        Today
## Year   -0.030519101  0.84194162 -0.032459894
## Lag1   -0.008183096 -0.06495131 -0.075031842
## Lag2   -0.072499482 -0.08551314  0.059166717
## Lag3    0.060657175 -0.06928771 -0.071243639
## Lag4   -0.075675027 -0.06107462 -0.007825873
## Lag5    1.000000000 -0.05851741  0.011012698
## Volume -0.058517414  1.00000000 -0.033077783
## Today   0.011012698 -0.03307778  1.000000000
```

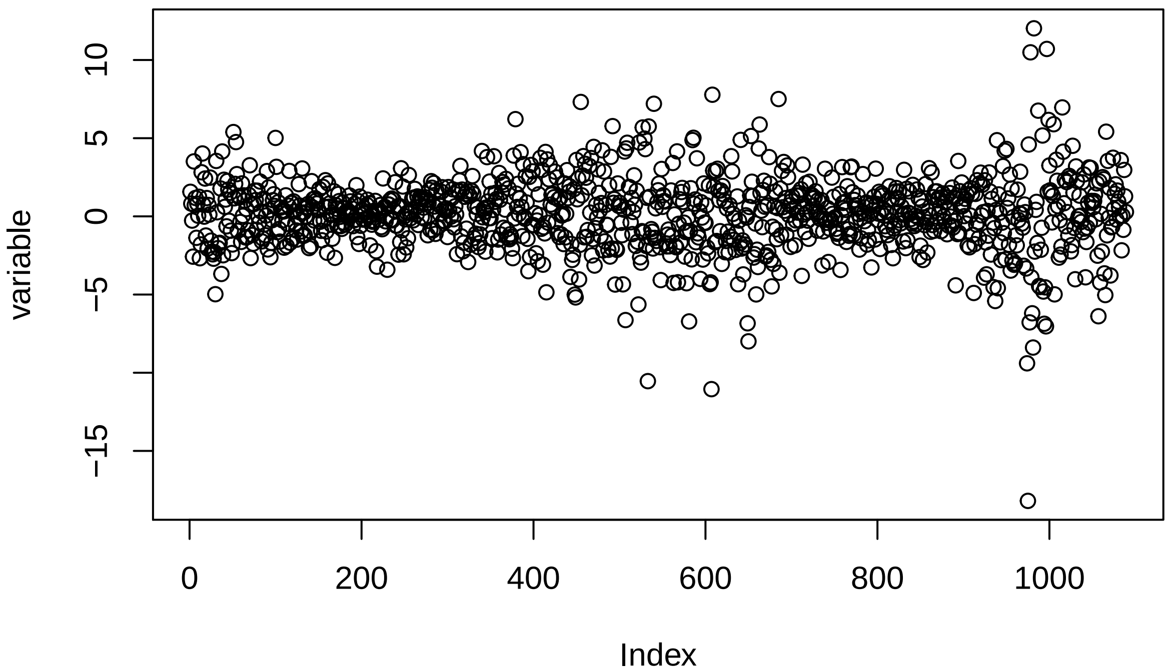
The only readily apparent trends are between the “year” and volume variables.

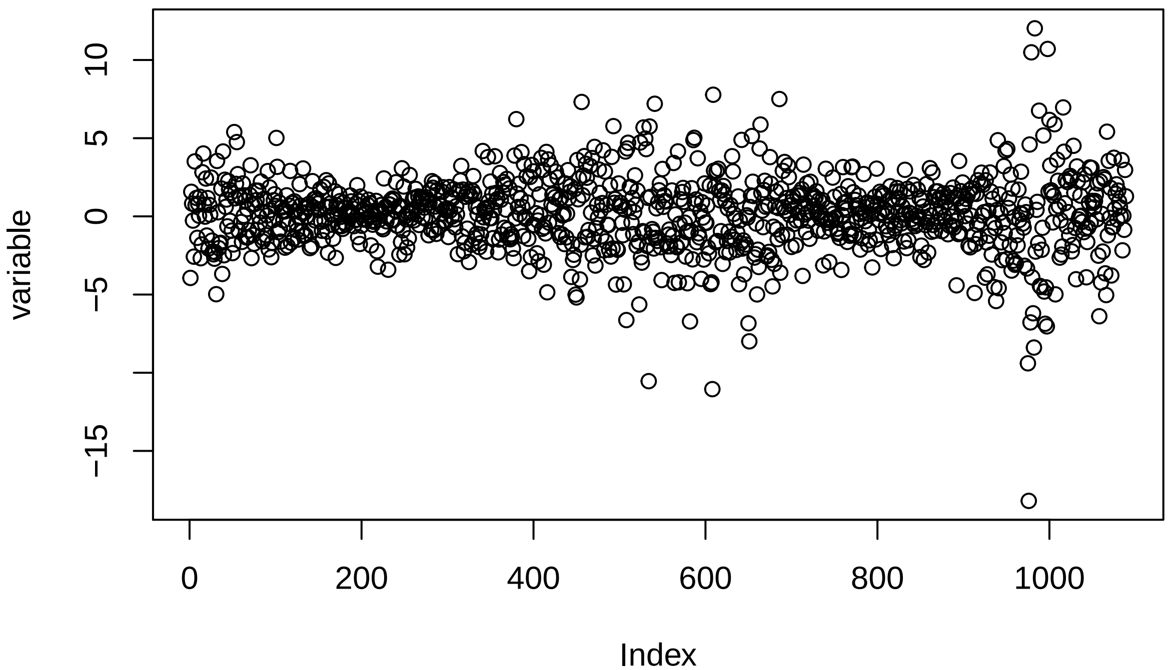
```
attach(Weekly)
for (variable in Weekly){
  plot(variable)
}
```

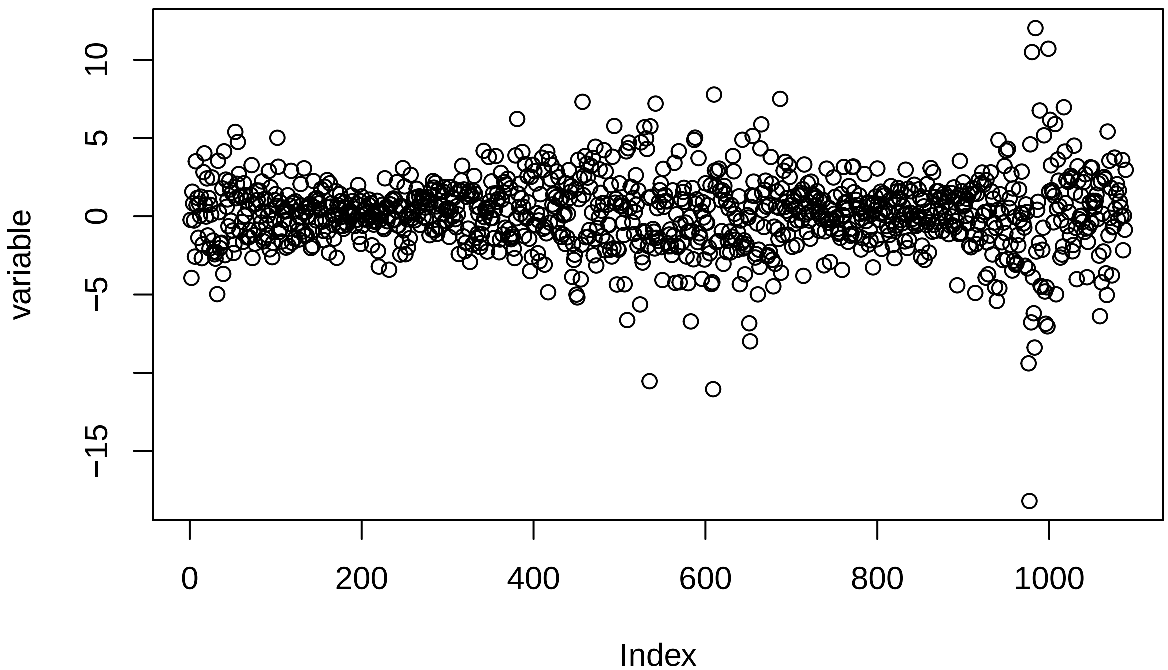


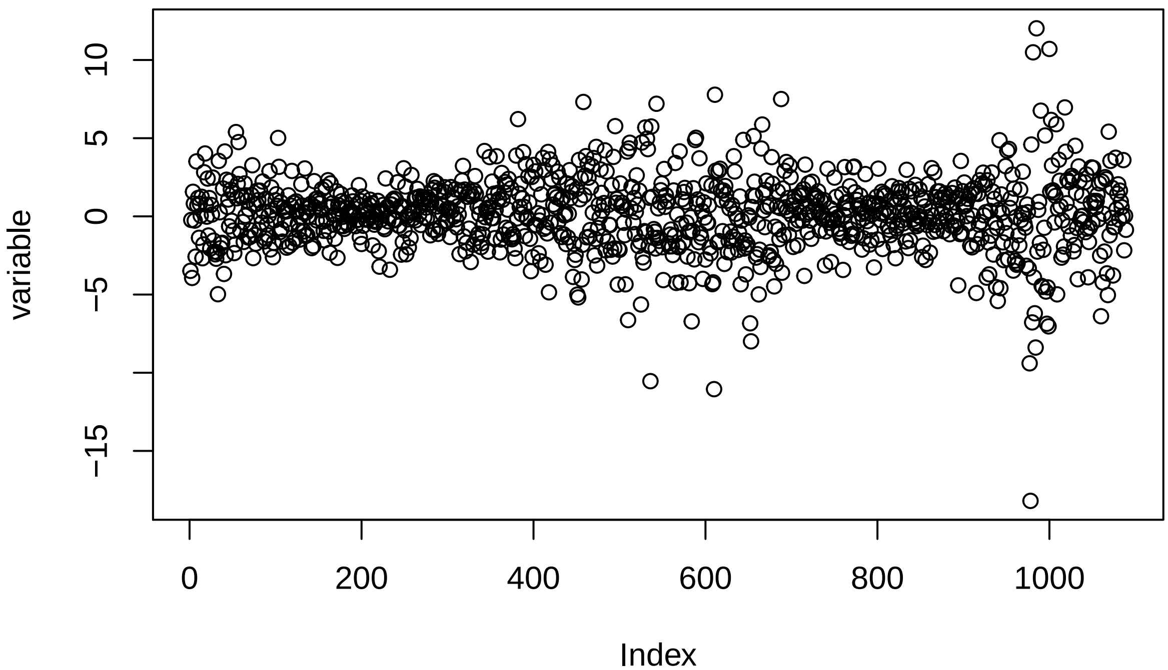
2010
2005
2000
1995
1990
variable
0
200
400
600
800
1000
Index

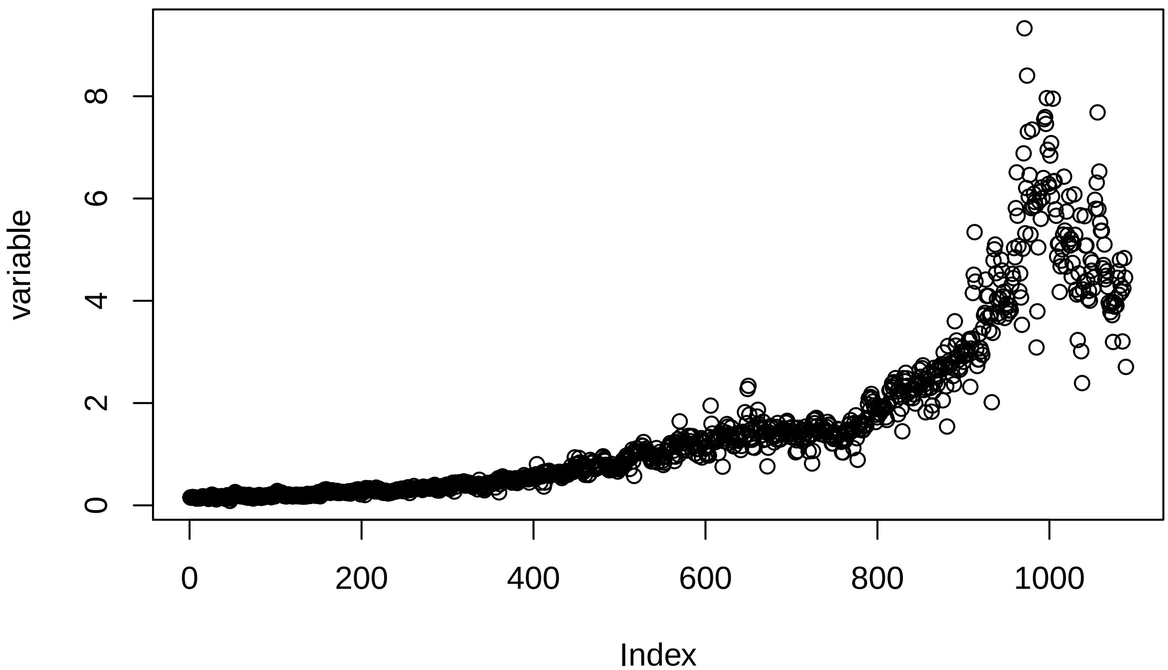

variable
10
5
0
−5
−15
0
200
400
600
800
1000
Index

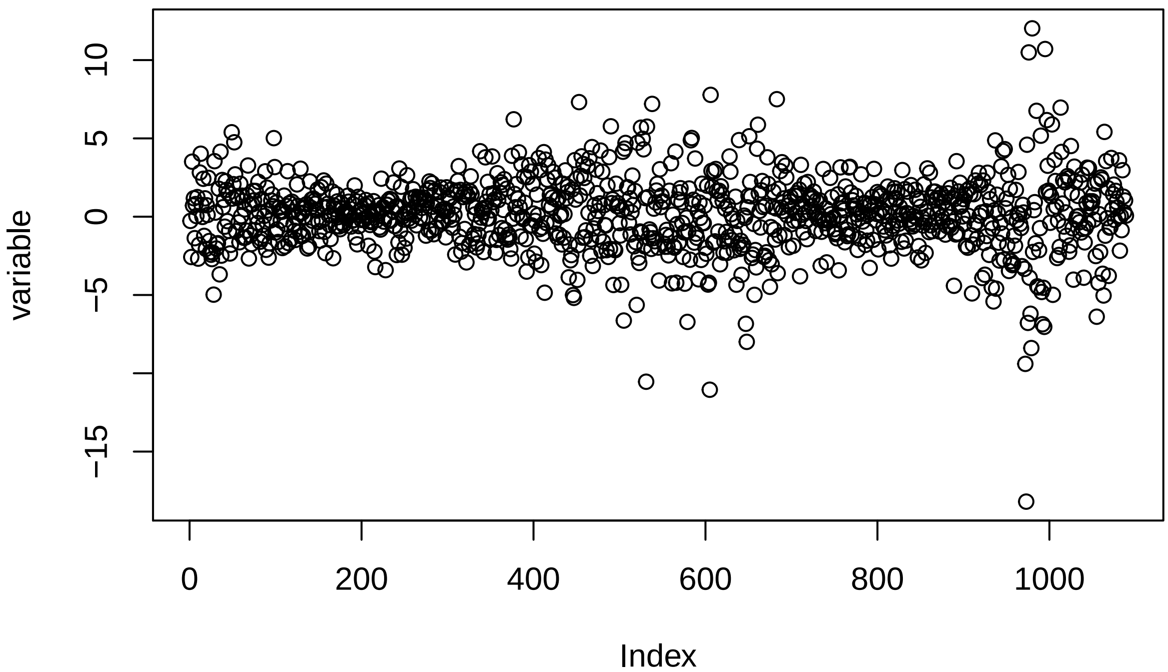

variable
10
5
0
−5
−15
0
200
400
600
800
1000
Index

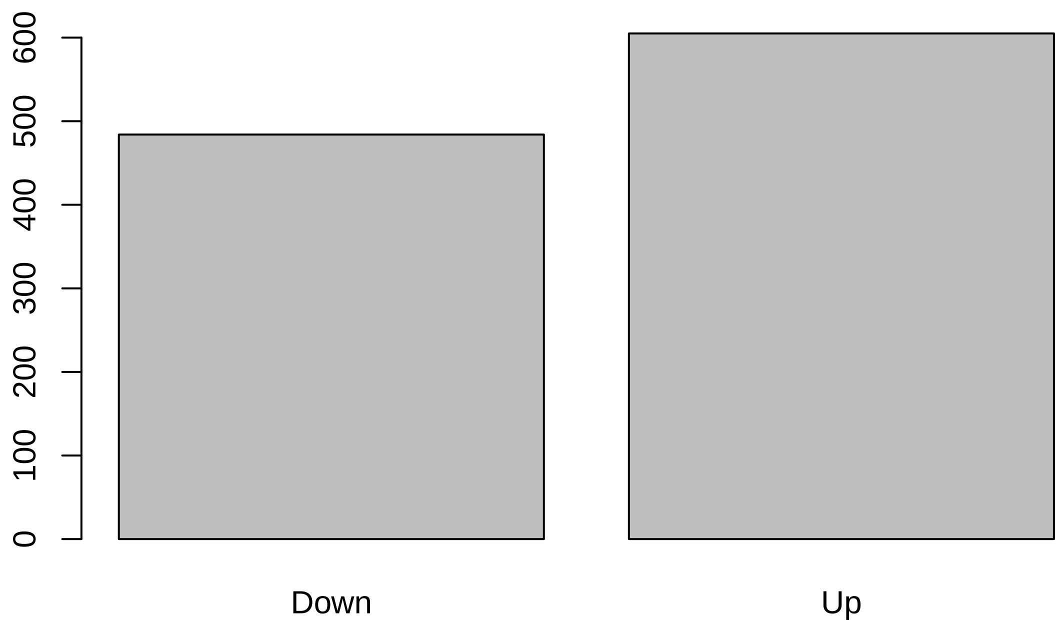

variable
10
5
0
−5
−15
0
200
400
600
800
1000
Index


variable
10
5
0
−5
−15
0
200
400
600
800
1000
Index


variable
10
5
0
−5
−15
Index
0
200
400
600
800
1000


variable
0
2
4
6
8
0
200
400
600
800
1000
Index


variable
10
5
0
−5
−15
0
200
400
600
800
1000
Index



b)

Lag2 is statistically significant at alpha = .05

```
logisticReg <- glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume, data = Weekly, family = binomi
summary(logisticReg)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
##     Volume, family = binomial, data = Weekly)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6949  -1.2565   0.9913   1.0849   1.4579
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.26686    0.08593   3.106   0.0019 **
## Lag1        -0.04127    0.02641  -1.563   0.1181
## Lag2         0.05844    0.02686   2.175   0.0296 *
## Lag3        -0.01606    0.02666  -0.602   0.5469
## Lag4        -0.02779    0.02646  -1.050   0.2937
## Lag5        -0.01447    0.02638  -0.549   0.5833
## Volume      -0.02274    0.03690  -0.616   0.5377
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1496.2  on 1088  degrees of freedom
## Residual deviance: 1486.4  on 1082  degrees of freedom
## AIC: 1500.4
##
## Number of Fisher Scoring iterations: 4
```

c)

```
logisticPrediction = predict(logisticReg, type = "response")

catPrediction <- rep("", length(logisticPrediction))
catPrediction[logisticPrediction > 0.5] <- "Up"
catPrediction[logisticPrediction < 0.5] <- "Down"

cmTable <- table(catPrediction, Direction)
cmTable
```

```
##               Direction
## catPrediction Down  Up
##          Down   54  48
##          Up    430 557
```

Interpretation:

```
cat("Proportion of correct predictions:", (cmTable[,'Down']['Down']+cmTable[,'Up']['Up'])/sum(cmTable))
```

```
## Proportion of correct predictions: 0.5610652
```

```
cat("\nProportion of correct predictions while market is increasing:", cmTable[,'Up']['Up']/(sum(cmTabl
```

```
##
## Proportion of correct predictions while market is increasing: 0.9206612
```

```
cat("\nProportion of correct predictions while market is decreasing:", cmTable[,'Down']['Down']/(sum(cm
```

```
##
## Proportion of correct predictions while market is decreasing: 0.1115702
```

d)

```
train <- (Year <= 2008)
Data0910 <- Weekly[!train,]
Direction0910 <- Direction[!train]
logisticRegTrain <- glm(Direction ~ Lag2, data = Weekly, family = binomial, subset = train)
summary(logisticRegTrain)
```

```
##
## Call:
## glm(formula = Direction ~ Lag2, family = binomial, data = Weekly,
##     subset = train)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.536  -1.264   1.021   1.091   1.368
```

```
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.20326    0.06428   3.162  0.00157 **
## Lag2         0.05810    0.02870   2.024  0.04298 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1354.7  on 984  degrees of freedom
## Residual deviance: 1350.5  on 983  degrees of freedom
## AIC: 1354.5
##
## Number of Fisher Scoring iterations: 4
```

```
logisticPrediction2 <- predict(logisticRegTrain, Data0910, type = "response")

catPrediction2 <- rep("", length(logisticPrediction2))
catPrediction2[logisticPrediction2 > 0.5] <- "Up"
catPrediction2[logisticPrediction2 < 0.5] <- "Down"

cmTable2 <- table(catPrediction2, Direction0910)
cmTable2
```

```
##               Direction0910
## catPrediction2 Down Up
##           Down    9  5
##           Up     34 56
```

Interpretation:

```
cat("Proportion of correct predictions:", (cmTable2[,'Down']['Down']+cmTable2[,'Up']['Up'])/sum(cmTable2
```

```
## Proportion of correct predictions: 0.625
```

```
cat("\nProportion of correct predictions while market is increasing:", cmTable2[,'Up']['Up']/(sum(cmTab
```

```
##
## Proportion of correct predictions while market is increasing: 0.9180328
```

```
cat("\nProportion of correct predictions while market is decreasing:", cmTable2[,'Down']['Down']/(sum(cm
```

```
##
## Proportion of correct predictions while market is decreasing: 0.2093023
```

g)

```
library(class)
trainKNN <- as.matrix(Lag2[train])
testKNN <- as.matrix(Lag2[!train])
trainKKNDirection <- Direction[train]
set.seed(1)
knnPrediction <- knn(trainKNN, testKNN, trainKKNDirection, k = 1)
knnTable <- table(knnPrediction, Direction0910)
knnTable
```

```
##              Direction0910
```

```
## knnPrediction Down Up
##          Down   21 30
##          Up     22 31
```

Interpretation:

```r
cat("Proportion of correct predictions:", (knnTable[,'Down']['Down']+knnTable[,'Up']['Up'])/sum(knnTabl
```

```
## Proportion of correct predictions: 0.5
```

```r
cat("\nProportion of correct predictions while market is increasing:", knnTable[,'Up']['Up']/(sum(knnTa
```

```
##
## Proportion of correct predictions while market is increasing: 0.5081967
```

```r
cat("\nProportion of correct predictions while market is decreasing:", knnTable[,'Down']['Down']/(sum(kn
```

```
##
## Proportion of correct predictions while market is decreasing: 0.4883721
```

h)

In just comparing the logisitc and K-nn regression (k=1), it appears that the logisitc regression has a lower error rate for the given training (1990-2008) and test (2009-2010) that are given.

i)

```r
for (i in c(1, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100)){
  set.seed(1)
  knnLoop <- knn(trainKNN, testKNN, trainKKNDirection, k = i)
  confusionTable <- table(knnLoop, Direction0910)
  cat("k =", i, "\n")
  print(confusionTable)

  total = confusionTable[1,1] + confusionTable[1,2] + confusionTable[2,1] + confusionTable[2,2]
  correct = (confusionTable[1,1] + confusionTable[2,2]) / total
  increaseCorrect = confusionTable[2,2] / (confusionTable[1,2] + confusionTable[2,2])
  decreaseCorrect = confusionTable[1,1] / (confusionTable[1,1] + confusionTable[2,1])

  cat("\nProportion of correct predictions:", correct)
  cat("\nProportion of correct predictions while market is increasing:", increaseCorrect)
  cat("\nProportion of correct predictions while market is decreasing:", decreaseCorrect, "\n\n\n")
}
```

```
## k = 1
##        Direction0910
## knnLoop Down Up
##    Down   21 30
##    Up     22 31
##
## Proportion of correct predictions: 0.5
## Proportion of correct predictions while market is increasing: 0.5081967
## Proportion of correct predictions while market is decreasing: 0.4883721
##
##
## k = 10
##        Direction0910
```

```
## knnLoop Down Up
##    Down   17 21
##    Up     26 40
##
## Proportion of correct predictions: 0.5480769
## Proportion of correct predictions while market is increasing: 0.6557377
## Proportion of correct predictions while market is decreasing: 0.3953488
##
##
## k = 20
##        Direction0910
## knnLoop Down Up
##    Down   21 21
##    Up     22 40
##
## Proportion of correct predictions: 0.5865385
## Proportion of correct predictions while market is increasing: 0.6557377
## Proportion of correct predictions while market is decreasing: 0.4883721
##
##
## k = 30
##        Direction0910
## knnLoop Down Up
##    Down   20 24
##    Up     23 37
##
## Proportion of correct predictions: 0.5480769
## Proportion of correct predictions while market is increasing: 0.6065574
## Proportion of correct predictions while market is decreasing: 0.4651163
##
##
## k = 40
##        Direction0910
## knnLoop Down Up
##    Down   21 24
##    Up     22 37
##
## Proportion of correct predictions: 0.5576923
## Proportion of correct predictions while market is increasing: 0.6065574
## Proportion of correct predictions while market is decreasing: 0.4883721
##
##
## k = 50
##        Direction0910
## knnLoop Down Up
##    Down   20 23
##    Up     23 38
##
## Proportion of correct predictions: 0.5576923
## Proportion of correct predictions while market is increasing: 0.6229508
## Proportion of correct predictions while market is decreasing: 0.4651163
##
##
## k = 60
```

```
##        Direction0910
## knnLoop Down Up
##    Down   18 20
##    Up     25 41
##
## Proportion of correct predictions: 0.5673077
## Proportion of correct predictions while market is increasing: 0.6721311
## Proportion of correct predictions while market is decreasing: 0.4186047
##
##
## k = 70
##        Direction0910
## knnLoop Down Up
##    Down   13 15
##    Up     30 46
##
## Proportion of correct predictions: 0.5673077
## Proportion of correct predictions while market is increasing: 0.7540984
## Proportion of correct predictions while market is decreasing: 0.3023256
##
##
## k = 80
##        Direction0910
## knnLoop Down Up
##    Down   10 14
##    Up     33 47
##
## Proportion of correct predictions: 0.5480769
## Proportion of correct predictions while market is increasing: 0.7704918
## Proportion of correct predictions while market is decreasing: 0.2325581
##
##
## k = 90
##        Direction0910
## knnLoop Down Up
##    Down   10 10
##    Up     33 51
##
## Proportion of correct predictions: 0.5865385
## Proportion of correct predictions while market is increasing: 0.8360656
## Proportion of correct predictions while market is decreasing: 0.2325581
##
##
## k = 100
##        Direction0910
## knnLoop Down Up
##    Down   10 11
##    Up     33 50
##
## Proportion of correct predictions: 0.5769231
## Proportion of correct predictions while market is increasing: 0.8196721
## Proportion of correct predictions while market is decreasing: 0.2325581
```

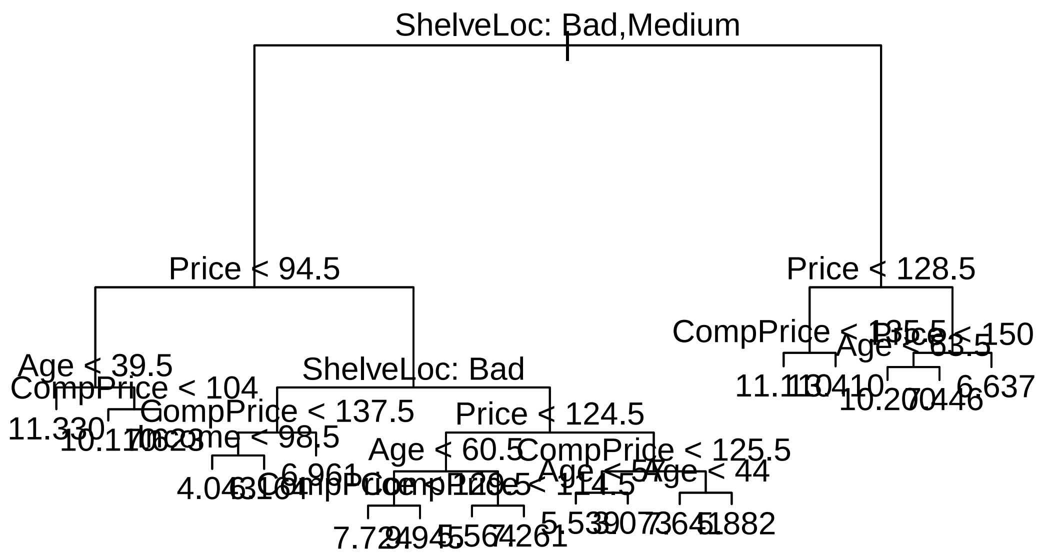
Chapter 8, Number 8

a)

```r
library(tree)
train <- sample(1:nrow(Carseats), nrow(Carseats) / 2)
trainCar <- Carseats[train, ]
testCar <- Carseats[-train, ]
```

b)

```r
carTree <- tree(Sales ~ ., data = trainCar)
plot(carTree)
text(carTree, pretty = 0)
```




```r
yhat <- predict(carTree, newdata = testCar)
cat('Tree Test MSE =', mean((yhat - testCar$Sales)^2))
```

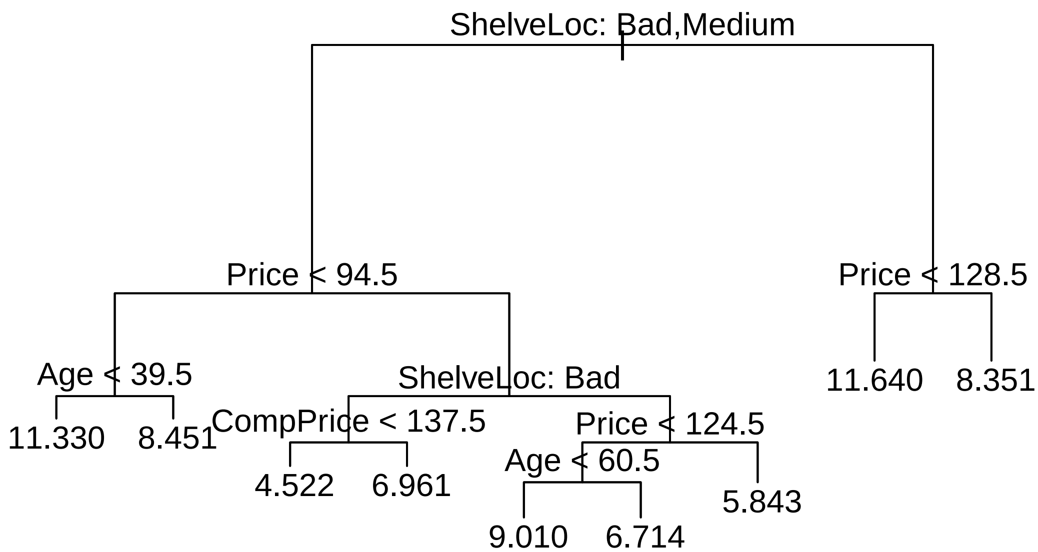
```
## Tree Test MSE = 5.179885
```

c)

```r
carCV <- cv.tree(carTree)
minTree <- which.min(carCV$dev)
cat("Cross validation suggests a tree of size", minTree)
```

```
## Cross validation suggests a tree of size 9
```

```r
carPrune <- prune.tree(carTree, best = minTree)
plot(carPrune)
text(carPrune, pretty = 0)
```




```r
yhat <- predict(carPrune, newdata = testCar)
cat('Pruned Tree Test MSE = ', mean((yhat - testCar$Sales)^2))
```

```
## Pruned Tree Test MSE =  5.123651
```

Note again that the above pruning is highly susceptible to variance in our random selection of the training/test split, but does in general raise MSE.

d)

```r
library(randomForest)
```

```
## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
carBag <- randomForest(Sales ~ ., data = trainCar, mtry = 10, ntree = 500, importance = TRUE)
yhatBag <- predict(carBag, newdata = testCar)
cat('Bagging Test MSE = ', mean((yhatBag - testCar$Sales)^2))
```

```
## Bagging Test MSE =  2.632852
```

```r
importance(carBag)
```

```
##                %IncMSE IncNodePurity
## CompPrice   27.3610852     188.89095
## Income       6.7676873     103.26122
## Advertising  9.4594556      71.70372
## Population  -4.0169669      55.17620
## Price       55.6874147     533.19036
## ShelveLoc   56.1958781     480.23664
## Age         15.8632898     141.27742
## Education    0.1875584      38.47232
## Urban        2.7546215      13.95130
## US           2.8419682      10.30215
```

e)

```r
for (i in 1:10){
  carForest <- randomForest(Sales ~ ., data = trainCar, mtry = i, ntree = 500, importance = TRUE)
  yhatForest <- predict(carForest, newdata = testCar)
  MSE <- mean((yhatForest - testCar$Sales)^2)

  cat('Random Forest Test MSE for ', i, ' predictors =', MSE, '\n\n')
  cat('Variable significance:\n\n')
  print(importance(carForest))
  cat('\n\n\n')
}
```

```
## Random Forest Test MSE for  1  predictors = 4.420172
##
## Variable significance:
##
##               %IncMSE IncNodePurity
## CompPrice    8.667729     129.08694
## Income       1.135553      99.91755
## Advertising  4.869872      96.26276
## Population  -1.224545      99.60549
## Price       21.076001     223.33380
## ShelveLoc   22.014370     195.63630
## Age         10.377518     137.69425
## Education    0.656436      74.53713
## Urban        2.789531      23.91611
## US           4.866622      32.90717
##
##
##
## Random Forest Test MSE for  2  predictors = 3.200449
##
## Variable significance:
```

```
##
##                %IncMSE IncNodePurity
## CompPrice   13.6154811     174.68334
## Income       2.5850525     133.82580
## Advertising  5.7402840     115.11911
## Population  -1.8133431     111.44957
## Price       30.2296356     346.23648
## ShelveLoc   32.9031504     308.34139
## Age         12.1838934     196.18286
## Education    0.7112872      85.88711
## Urban        1.1488451      27.43914
## US           3.6768052      30.24995
##
##
##
## Random Forest Test MSE for  3  predictors = 2.811538
##
## Variable significance:
##
##                %IncMSE IncNodePurity
## CompPrice   13.4637497     171.98697
## Income       3.2928392     125.78188
## Advertising  6.9026231     106.15211
## Population  -4.7218122      99.31323
## Price       36.4760695     396.80301
## ShelveLoc   40.1127509     357.49987
## Age         14.9889328     196.55210
## Education    0.7976227      71.06736
## Urban        0.4028820      22.70786
## US           3.8242542      24.97396
##
##
##
## Random Forest Test MSE for  4  predictors = 2.653847
##
## Variable significance:
##
##                %IncMSE IncNodePurity
## CompPrice   16.6209467     180.15816
## Income       3.8627611     119.64400
## Advertising  7.4755866      94.09206
## Population  -0.8854176      86.64159
## Price       40.6167409     427.53218
## ShelveLoc   41.0611068     406.98926
## Age         13.7839205     189.45967
## Education    1.5611106      63.32915
## Urban        1.9913057      18.25757
## US           3.5328825      22.24011
##
##
##
## Random Forest Test MSE for  5  predictors = 2.533278
##
## Variable significance:
```

```
##
##               %IncMSE IncNodePurity
## CompPrice   16.890784     181.82955
## Income       5.627055     109.77118
## Advertising  6.808810      92.58206
## Population  -3.510390      72.02929
## Price       43.272190     459.46880
## ShelveLoc   48.115279     438.22624
## Age         15.201112     176.41294
## Education   -1.415935      53.09625
## Urban        2.789161      17.08755
## US           4.324524      18.13996
##
##
##
## Random Forest Test MSE for  6  predictors = 2.49997
##
## Variable significance:
##
##                %IncMSE IncNodePurity
## CompPrice   23.5313349     174.54243
## Income       5.8156673     106.70718
## Advertising  7.0452769      87.38187
## Population  -1.7903977      69.03262
## Price       47.5164058     494.14911
## ShelveLoc   49.9557547     445.74211
## Age         17.3933134     172.19561
## Education    0.1685286      49.41324
## Urban        1.9672722      15.84302
## US           3.4035664      14.83795
##
##
##
## Random Forest Test MSE for  7  predictors = 2.510898
##
## Variable significance:
##
##                %IncMSE IncNodePurity
## CompPrice   21.9637554     182.65763
## Income       4.3878752     109.74517
## Advertising  7.5783907      79.41881
## Population  -2.2093457      60.19959
## Price       47.3008560     499.14167
## ShelveLoc   48.6946674     469.00933
## Age         15.7273880     162.81656
## Education   -0.8601696      45.86065
## Urban       -0.1547724      15.28609
## US           4.1744669      12.80000
##
##
##
## Random Forest Test MSE for  8  predictors = 2.481164
##
## Variable significance:
```

```
##
##                %IncMSE IncNodePurity
## CompPrice   23.9431799     184.52998
## Income       4.1404706     100.05917
## Advertising  8.1142304      82.42619
## Population  -4.5380002      57.71276
## Price       53.0957231     512.02056
## ShelveLoc   56.7333184     474.47457
## Age         15.8887698     155.42526
## Education    0.6213677      41.76509
## Urban        1.8436430      15.17217
## US           3.4148735      11.84043
##
##
##
## Random Forest Test MSE for  9  predictors = 2.576561
##
## Variable significance:
##
##                 %IncMSE IncNodePurity
## CompPrice   27.51835480    185.720630
## Income       8.51879998    102.141263
## Advertising  8.84939264     75.006995
## Population  -4.67951049     53.521368
## Price       54.52120132    518.432371
## ShelveLoc   57.49733656    483.375436
## Age         16.90599467    157.176462
## Education   -0.03251493     40.352068
## Urban        1.34363353     15.127313
## US           2.07637809      9.653227
##
##
##
## Random Forest Test MSE for  10  predictors = 2.565357
##
## Variable significance:
##
##                %IncMSE IncNodePurity
## CompPrice   27.3146403    185.546253
## Income       8.2342025    103.843257
## Advertising  8.3687203     81.245242
## Population  -4.0058187     51.219002
## Price       55.6361790    524.840700
## ShelveLoc   58.2411629    504.211819
## Age         17.0940287    142.817565
## Education    0.4838597     38.367245
## Urban        1.4349774     11.982586
## US           1.9294153      8.438386
```

As seen above, MSE tends to decrease as the number of variables considered at each split increases.

Chapter 8, Number 11

a)

```
train <- 1:1000
Caravan$Purchase <- ifelse(Caravan$Purchase == "Yes", 1, 0)
trainCaravan <- Caravan[train, ]
testCaravan <- Caravan[-train, ]
```

b)

```
library(gbm)
```

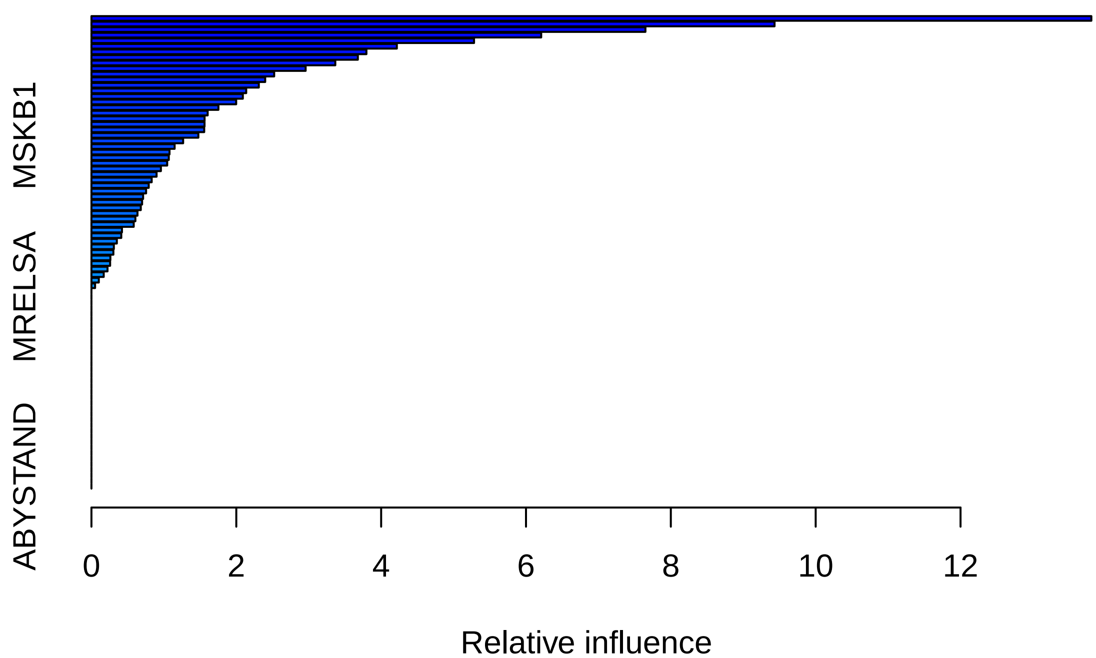
```
## Loaded gbm 2.1.5
```

```
caravanBoost <- gbm(Purchase ~ ., data = trainCaravan, distribution = "gaussian", n.trees = 1000, shrink
```

```
## Warning in gbm.fit(x = x, y = y, offset = offset, distribution =
## distribution, : variable 50: PVRAAUT has no variation.
```

```
## Warning in gbm.fit(x = x, y = y, offset = offset, distribution =
## distribution, : variable 71: AVRAAUT has no variation.
```

```
head(summary(caravanBoost))
```




```
##               var   rel.inf
## PPERSAUT PPERSAUT 13.806332
## MKOOPKLA MKOOPKLA  9.431953
```

```
## MOPLHOOG MOPLHOOG  7.649469
## MBERMIDD MBERMIDD  6.209954
## PBRAND     PBRAND  5.282896
## ABRAND     ABRAND  4.216933
```

The variables “PPERSAUT” and “MKOOPKLA” stand out as being most significant

c)

```
testProbabilities <- predict(caravanBoost, testCaravan, n.trees = 1000, type = "response")
toCategory <- ifelse(testProbabilities > 0.2, 1, 0)
caravanTable = table(testCaravan$Purchase, toCategory)
caravanTable
```

```
##    toCategory
##        0    1
##   0 4501   32
##   1  277   12
```

```
cat('Fraction of the people predicted to make a purchase do in fact make one: ', caravanTable[2,2] / sum
```

```
## Fraction of the people predicted to make a purchase do in fact make one:  0.2727273
```

```
caravanLogistic <- glm(Purchase ~ ., data = trainCaravan, family = "binomial")
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
testProbabilities2 <- predict(caravanLogistic, testCaravan, type = "response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type
## == : prediction from a rank-deficient fit may be misleading
```

```
toCategory2 <- ifelse(testProbabilities > 0.2, 1, 0)
caravanLogTable = table(testCaravan$Purchase, toCategory2)
caravanLogTable
```

```
##    toCategory2
##        0    1
##   0 4501   32
##   1  277   12
```

```
cat('Fraction of the people predicted to make a purchase do in fact make one: ', caravanLogTable[2,2] /
```

```
## Fraction of the people predicted to make a purchase do in fact make one:  0.2727273
```

```
trainKNN <- as.matrix(trainCaravan)
testKNN <- as.matrix(testCaravan)
trainClass <- Caravan[1:1000, 'Purchase']
knnPrediction <- knn(trainKNN, testKNN, trainClass, k = 1)
knnTable <- table(testCaravan$Purchase,knnPrediction)
knnTable
```

```
##    knnPrediction
##        0    1
##   0 4287  246
##   1  262   27
```

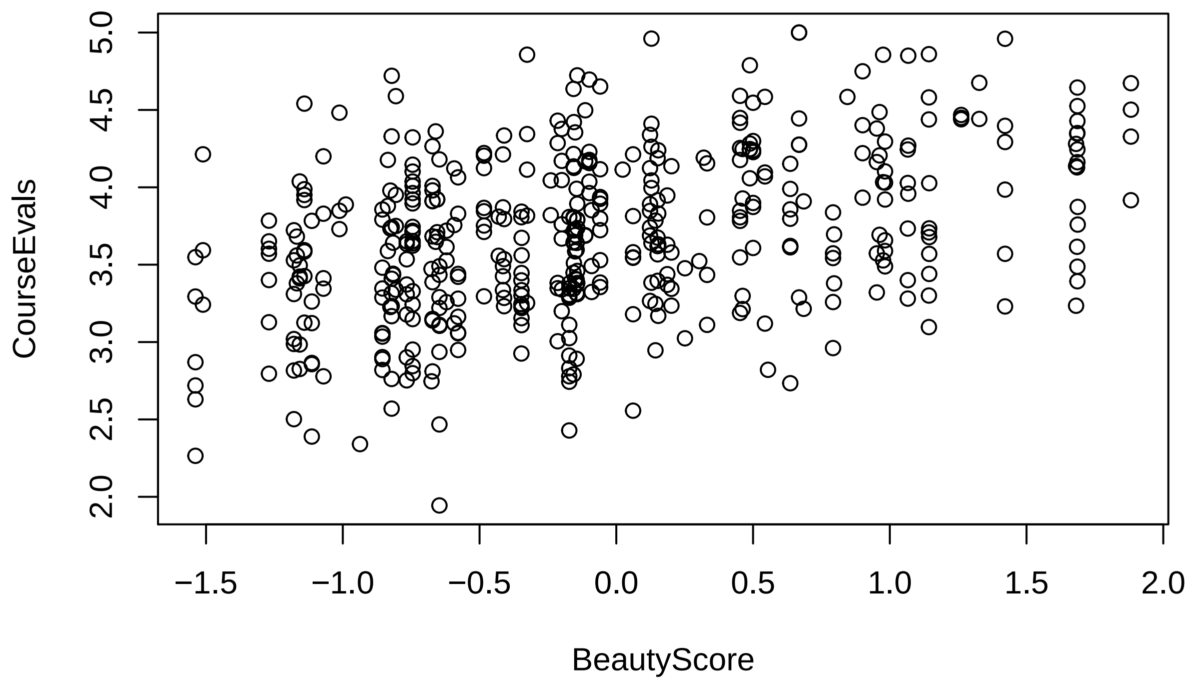
```
cat('Fraction of the people predicted to make a purchase do in fact make one: ', knnTable[2,2] / sum(kn
```

```
## Fraction of the people predicted to make a purchase do in fact make one:  0.0989011
```

While the boosting and logistic models match, KNN does a very poor job of prediction. Even moderatly raising the k value adds so much bias towards predicting a non-purchase that the model is rendered useless.

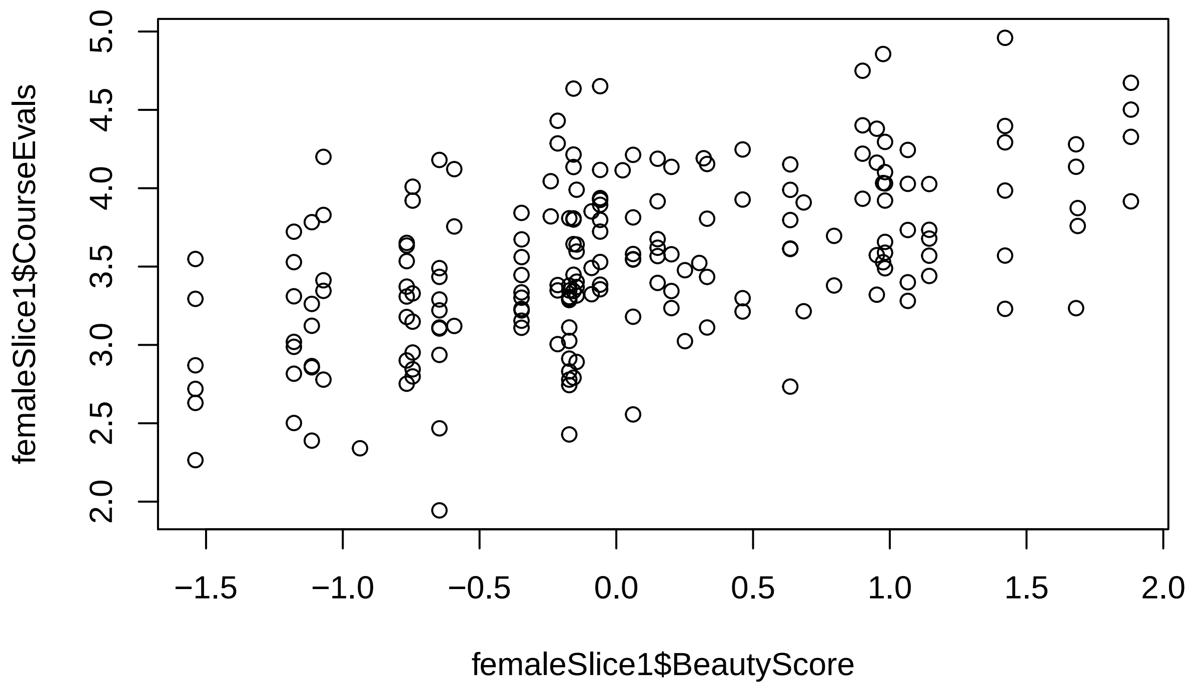
Problem 1

```
BeautyData <- read.csv("BeautyData.csv")
attach(BeautyData)
plot(BeautyScore, CourseEvals)
```

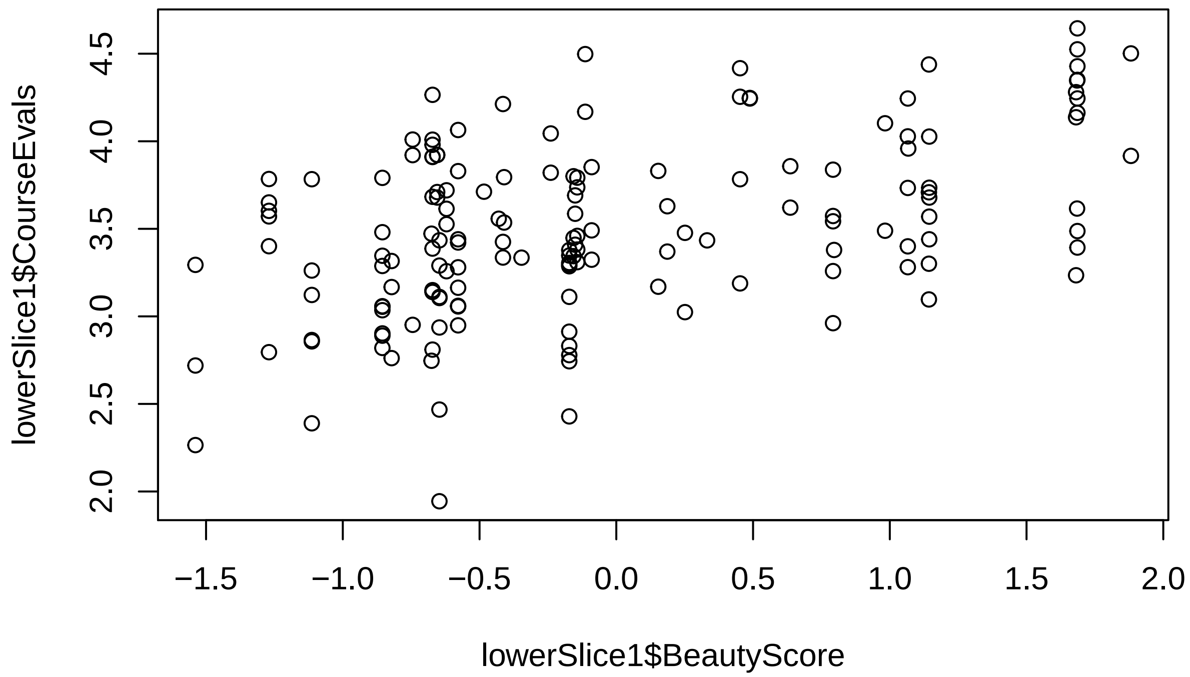


To gain some intution for each of the variable's effect on course evaluation, I made a few plots seeting each of female, low, nonenglish, and tenuretrack to sero or one.

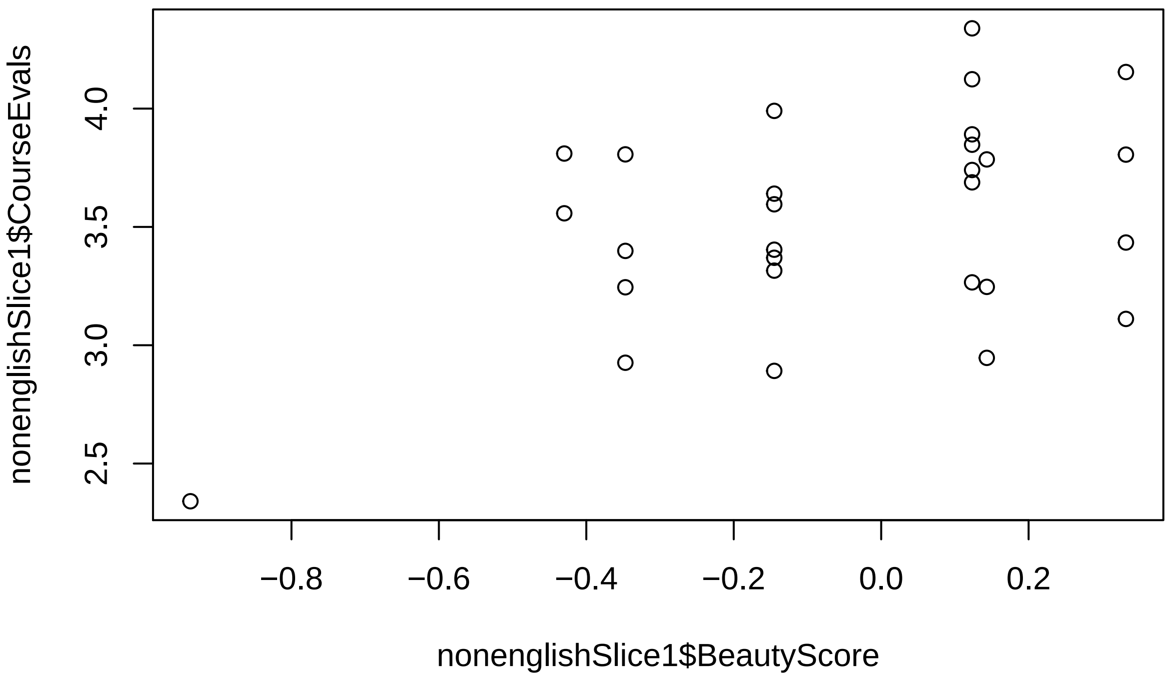
```
femaleSlice1 = BeautyData[(female == 1),]
lowerSlice1 = BeautyData[(lower == 1),]
nonenglishSlice1 = BeautyData[(nonenglish == 1),]
tenuretrackSlice1 = BeautyData[(tenuretrack == 1),]
femaleSlice0 = BeautyData[(female == 0),]
lowerSlice0 = BeautyData[(lower == 0),]
nonenglishSlice0 = BeautyData[(nonenglish == 0),]
tenuretrackSlice0 = BeautyData[(tenuretrack == 0),]

plot(femaleSlice1$BeautyScore, femaleSlice1$CourseEvals)
```



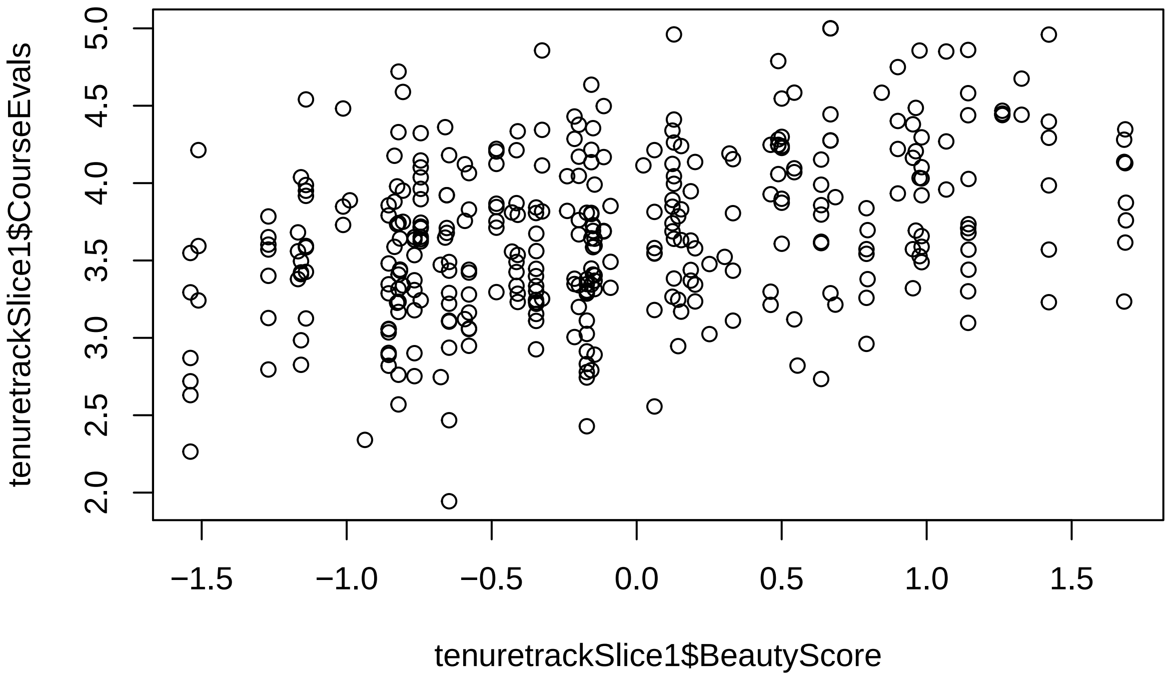

```
plot(lowerSlice1$BeautyScore, lowerSlice1$CourseEvals)
```



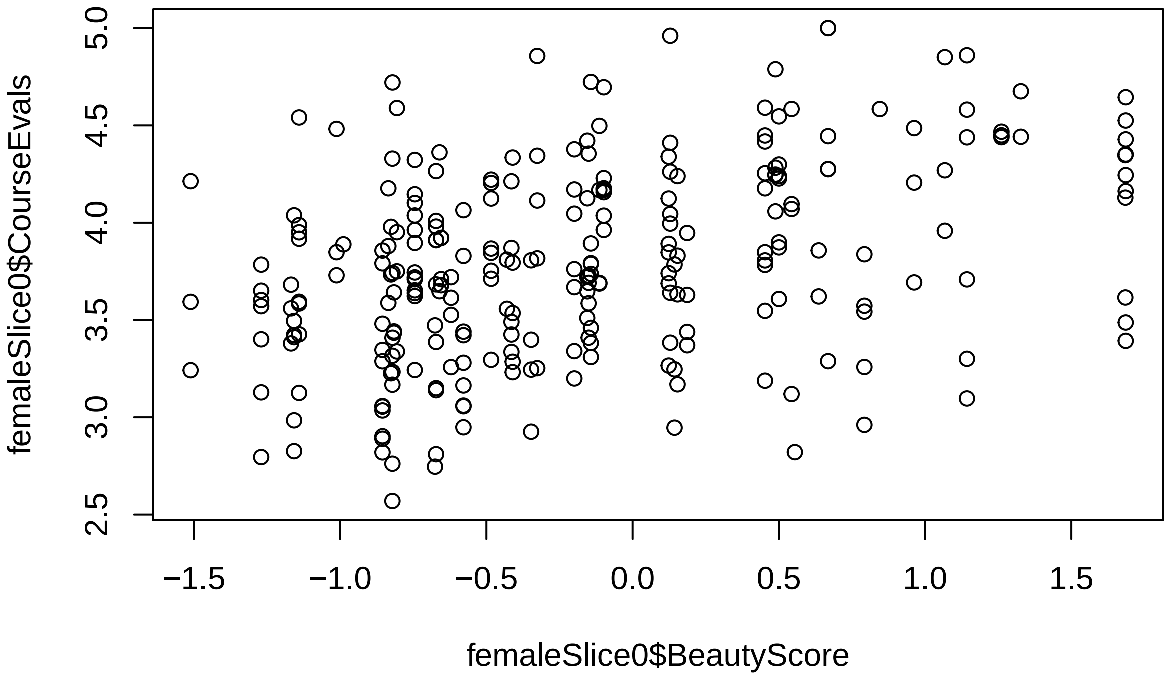

```
plot(nonenglishSlice1$BeautyScore, nonenglishSlice1$CourseEvals)
```



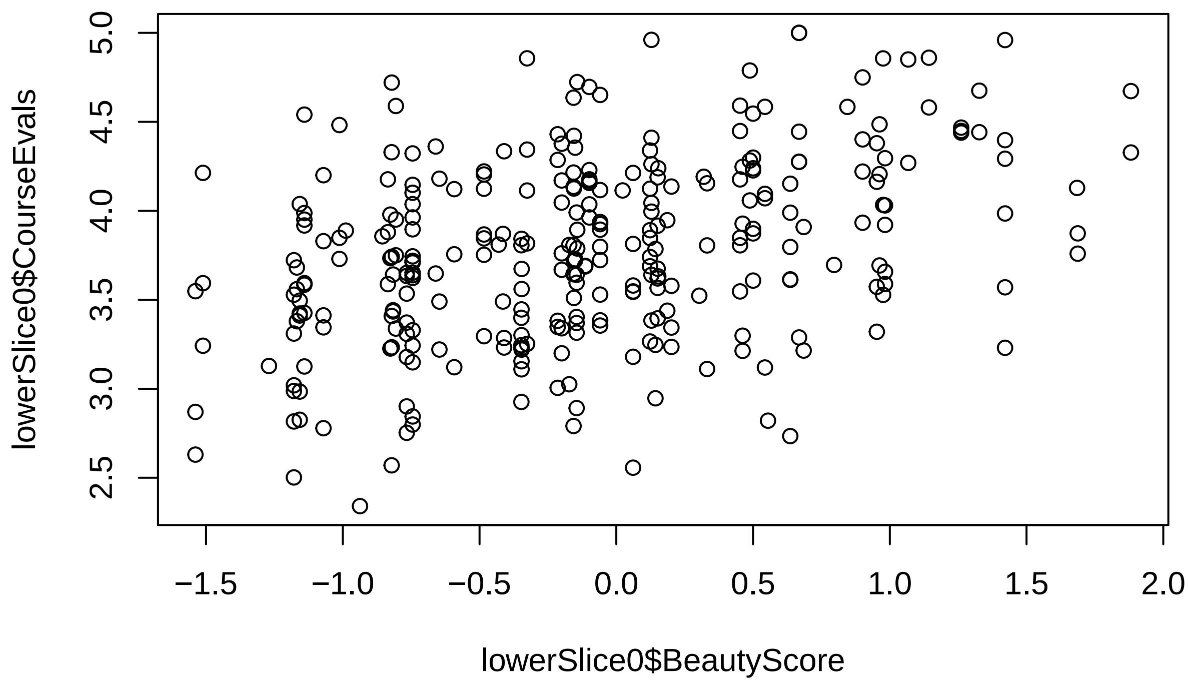

```
plot(tenuretrackSlice1$BeautyScore, tenuretrackSlice1$CourseEvals)
```



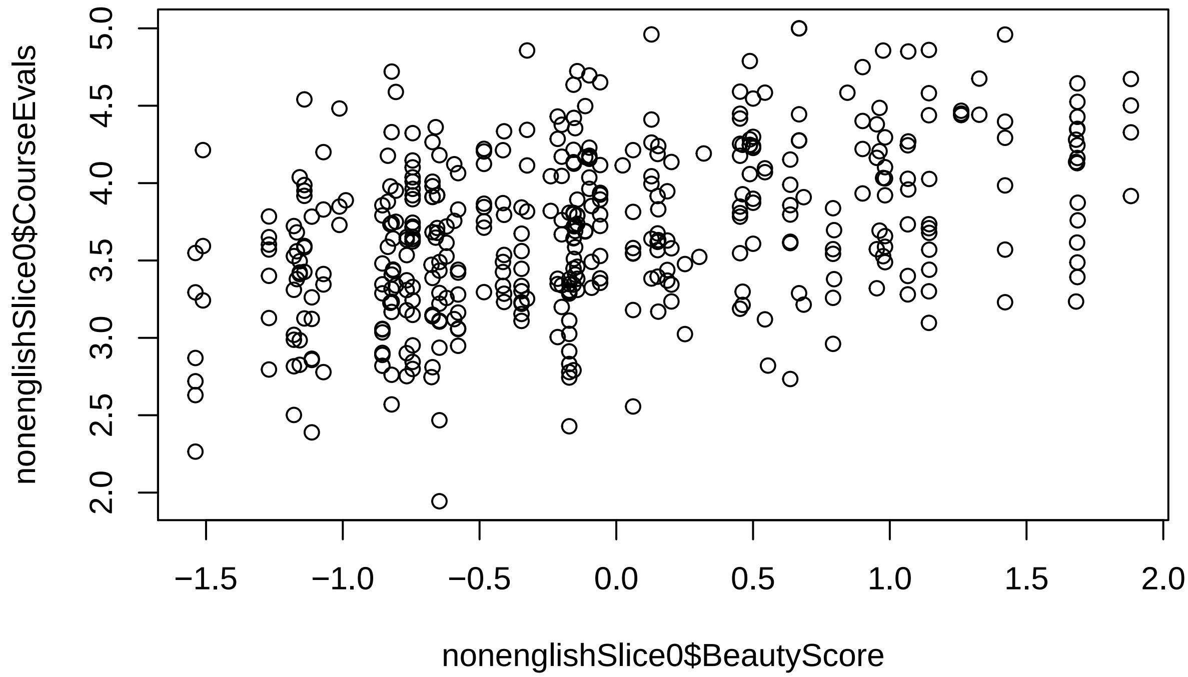

```
plot(femaleSlice0$BeautyScore, femaleSlice0$CourseEvals)
```



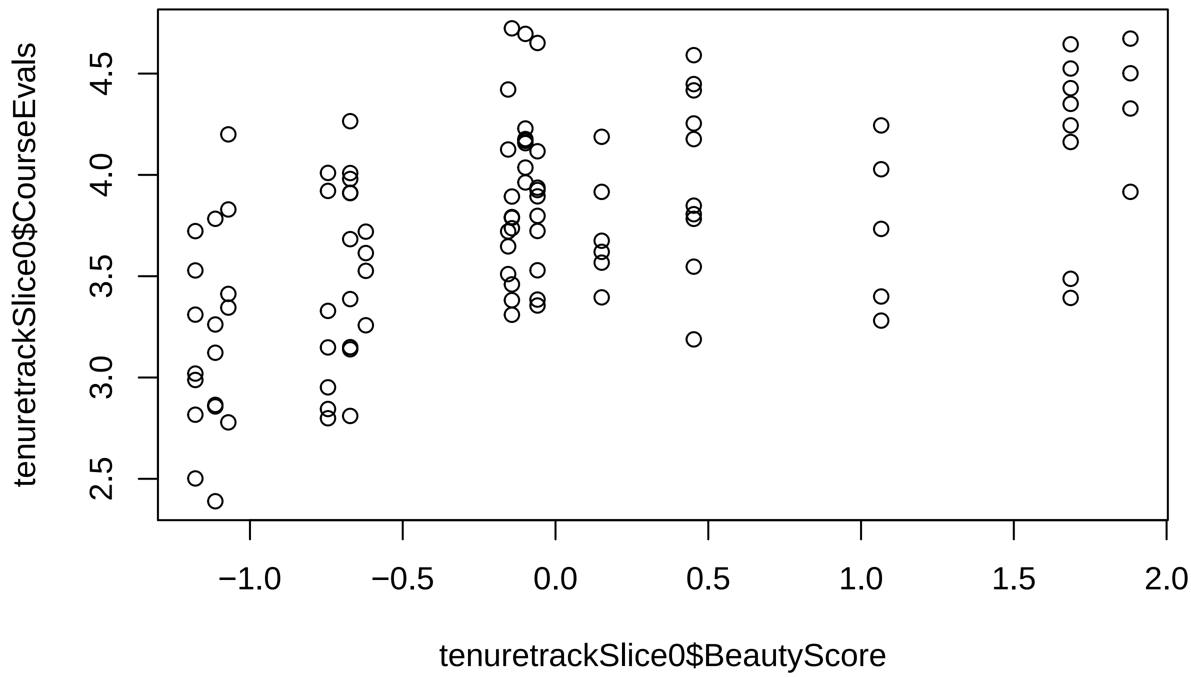

```
plot(lowerSlice0$BeautyScore, lowerSlice0$CourseEvals)
```




```
plot(nonenglishSlice0$BeautyScore, nonenglishSlice0$CourseEvals)
```




```r
plot(tenuretrackSlice0$BeautyScore, tenuretrackSlice0$CourseEvals)
```




```r
BeautyTrain = sample(1:dim(BeautyData)[1], dim(BeautyData)[1]/2)
BeautyTest <- -BeautyTrain
BeautyTrainData <- BeautyData[BeautyTrain, ]
BeautyTestData <- BeautyData[BeautyTest, ]
```

```r
BeautyTrainRegression = lm(CourseEvals ~ BeautyScore, data = BeautyTrainData)
BeautyTestPredictions = predict(BeautyTrainRegression, BeautyTestData)
print(summary(BeautyTrainRegression))
```

```
##
## Call:
## lm(formula = CourseEvals ~ BeautyScore, data = BeautyTrainData)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.59930 -0.36854  0.01082  0.36784  1.22351
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.71579    0.03281 113.257  < 2e-16 ***
## BeautyScore  0.26635    0.04022   6.623 2.48e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.497 on 229 degrees of freedom
## Multiple R-squared:  0.1608, Adjusted R-squared:  0.1571
## F-statistic: 43.86 on 1 and 229 DF,  p-value: 2.481e-10
```

```r
cat("Least Squares Test MSE =", mean((BeautyTestPredictions - BeautyTestData$CourseEvals)^2))
```

```
## Least Squares Test MSE = 0.215734
```

```r
BeautyData$female <- as.factor(BeautyData$female)
BeautyData$lower <- as.factor(BeautyData$lower)
BeautyData$nonenglish <- as.factor(BeautyData$nonenglish)
BeautyData$tenuretrack <- as.factor(BeautyData$tenuretrack)

BeautyTrainRegression = lm(CourseEvals ~ ., data = BeautyTrainData)
BeautyTestPredictions = predict(BeautyTrainRegression, BeautyTestData)
print(summary(BeautyTrainRegression))
```

```
##
## Call:
## lm(formula = CourseEvals ~ ., data = BeautyTrainData)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.16460 -0.30553  0.00116  0.27516  1.06273
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.09234    0.07222  56.665  < 2e-16 ***
## BeautyScore  0.32604    0.03574   9.122  < 2e-16 ***
## female      -0.37832    0.05833  -6.486 5.53e-10 ***
## lower       -0.40594    0.06398  -6.345 1.21e-09 ***
## nonenglish  -0.17098    0.11098  -1.541    0.125
## tenuretrack -0.09006    0.06886  -1.308    0.192
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4325 on 225 degrees of freedom
## Multiple R-squared:  0.3757, Adjusted R-squared:  0.3618
## F-statistic: 27.08 on 5 and 225 DF,  p-value: < 2.2e-16
```

```r
cat("Least Squares Test MSE =", mean((BeautyTestPredictions - BeautyTestData$CourseEvals)^2))
```

```
## Least Squares Test MSE = 0.1826809
```

```r
library(glmnet)
trainMatrix <- model.matrix(CourseEvals ~ ., data = BeautyTrainData)
testMatrix <- model.matrix(CourseEvals ~ ., data = BeautyTestData)

lasso <- glmnet(trainMatrix, BeautyTrainData$CourseEvals, alpha = 1, lambda = grid, thresh = 1e-12)
lassoCV <- cv.glmnet(trainMatrix, BeautyTrainData$CourseEvals, alpha = 1, lambda = grid, thresh = 1e-12)
lassoLambdaMin <- lassoCV$lambda.min
lassoTestPredictions <- predict(lasso, s = lassoLambdaMin, newx = testMatrix)
cat("Lasso Test MSE = ", mean((lassoTestPredictions - BeautyTestData$CourseEvals)^2), "\n\n")
```

```
## Lasso Test MSE =  0.1809954
```

```r
predict(lasso, s = lassoLambdaMin, type = "coefficients")
```

```
## 7 x 1 sparse Matrix of class "dgCMatrix"
##                      1
## (Intercept)  4.0458722
```

```
## (Intercept)  .
## BeautyScore  0.3098150
## female      -0.3519785
## lower       -0.3713525
## nonenglish  -0.1343748
## tenuretrack -0.0630716
```

```r
BeautyTrainRegression = lm(CourseEvals ~ (.)^2, data = BeautyTrainData)
BeautyTestPredictions = predict(BeautyTrainRegression, BeautyTestData)
```

```
## Warning in predict.lm(BeautyTrainRegression, BeautyTestData): prediction
## from a rank-deficient fit may be misleading
```

```r
print(summary(BeautyTrainRegression))
```

```
##
## Call:
## lm(formula = CourseEvals ~ (.)^2, data = BeautyTrainData)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.21283 -0.29432 -0.02387  0.28741  1.13293
##
## Coefficients: (1 not defined because of singularities)
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)              4.03093    0.11214  35.946  < 2e-16 ***
## BeautyScore              0.37469    0.10508   3.566 0.000447 ***
## female                  -0.33975    0.13669  -2.486 0.013695 *
## lower                   -0.35982    0.13561  -2.653 0.008564 **
## nonenglish              -0.36155    0.17394  -2.079 0.038834 *
## tenuretrack              0.04997    0.12178   0.410 0.681997
## BeautyScore:female       0.03421    0.07766   0.441 0.659993
## BeautyScore:lower       -0.11566    0.08273  -1.398 0.163556
## BeautyScore:nonenglish   0.17077    0.43365   0.394 0.694117
## BeautyScore:tenuretrack -0.01166    0.09011  -0.129 0.897175
## female:lower             0.22683    0.13034   1.740 0.083235 .
## female:nonenglish        0.28421    0.23736   1.197 0.232473
## female:tenuretrack      -0.17626    0.14551  -1.211 0.227120
## lower:nonenglish         0.37519    0.33370   1.124 0.262122
## lower:tenuretrack       -0.22166    0.14448  -1.534 0.126441
## nonenglish:tenuretrack        NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4298 on 216 degrees of freedom
## Multiple R-squared:  0.4081, Adjusted R-squared:  0.3698
## F-statistic: 10.64 on 14 and 216 DF,  p-value: < 2.2e-16
```

```r
cat("Least Squares Test MSE =", mean((BeautyTestPredictions - BeautyTestData$CourseEvals)^2))
```

```
## Least Squares Test MSE = 0.190175
```

```r
library(glmnet)
trainMatrix <- model.matrix(CourseEvals ~ (.)^2, data = BeautyTrainData)
testMatrix <- model.matrix(CourseEvals ~ (.)^2, data = BeautyTestData)
```

```r
lasso <- glmnet(trainMatrix, BeautyTrainData$CourseEvals, alpha = 1, lambda = grid, thresh = 1e-12)
lassoCV <- cv.glmnet(trainMatrix, BeautyTrainData$CourseEvals, alpha = 1, lambda = grid, thresh = 1e-12)
lassoLambdaMin <- lassoCV$lambda.min
lassoTestPredictions <- predict(lasso, s = lassoLambdaMin, newx = testMatrix)
cat("Lasso Test MSE = ", mean((lassoTestPredictions - BeautyTestData$CourseEvals)^2), "\n\n")
```

```
## Lasso Test MSE =  0.1802327
```

```r
predict(lasso, s = lassoLambdaMin, type = "coefficients")
```

```
## 17 x 1 sparse Matrix of class "dgCMatrix"
##                                 1
## (Intercept)             3.96236771
## (Intercept)             .
## BeautyScore             0.26525998
## female                 -0.25691443
## lower                  -0.22926947
## nonenglish             -0.08140653
## tenuretrack             .
## BeautyScore:female      0.05848840
## BeautyScore:lower       .
## BeautyScore:nonenglish  0.10132325
## BeautyScore:tenuretrack .
## female:lower            .
## female:nonenglish       .
## female:tenuretrack     -0.08676203
## lower:nonenglish        .
## lower:tenuretrack      -0.15920063
## nonenglish:tenuretrack  .
```

It is difficult to even determine for this data what significance the variables have on course evalations, let alone say anything about the causation of such an effect. Using concepts we have discussed in class such as interaction terms and the LASSO method prove to be decidedly inconclusive.

Problem 2

1/2)

Linear Regression and plotting of stratified histograms suggest a premium for both brick homes and homes in neighboorhood 3.

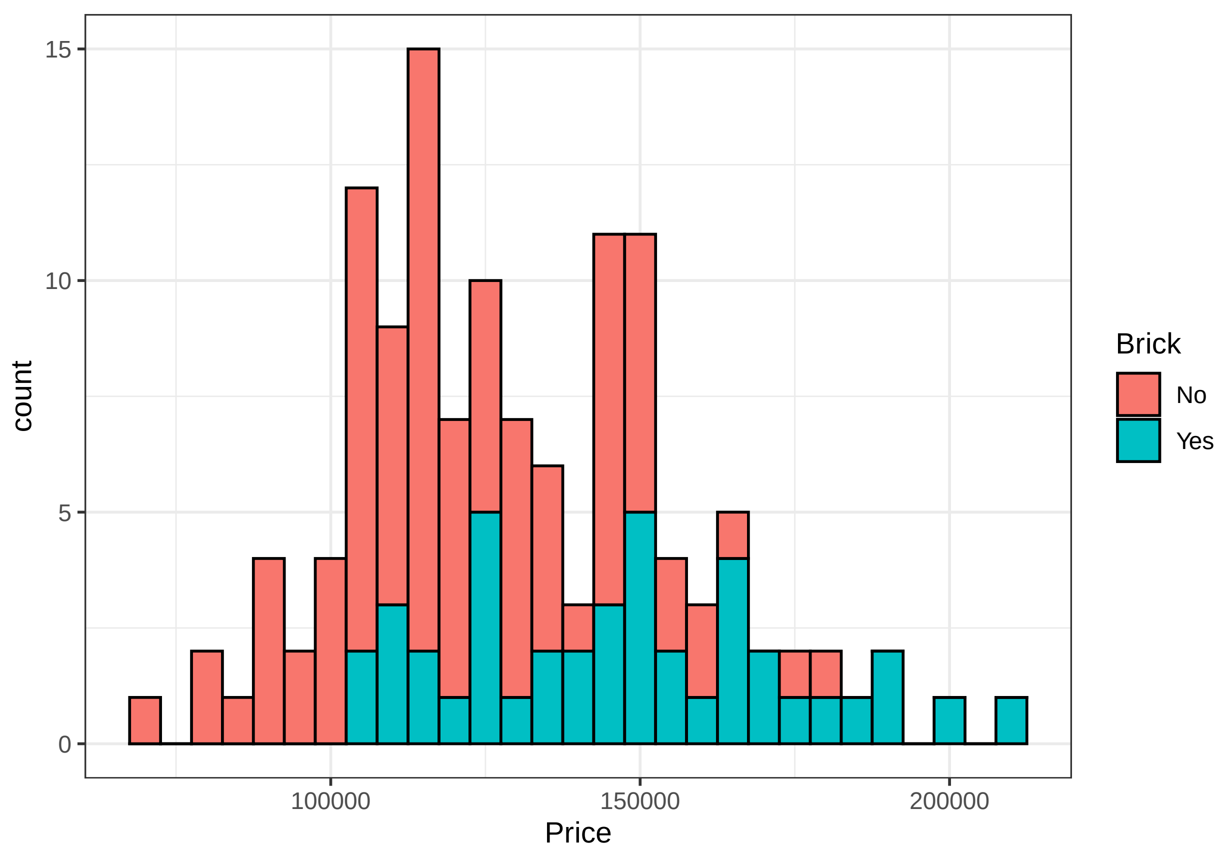
```r
MidCity <- read.csv("MidCity.csv")
```

```r
MidCity$Nbhd <- as.factor(MidCity$Nbhd)
priceModel <- lm(Price ~., data = MidCity)
summary(priceModel)
```

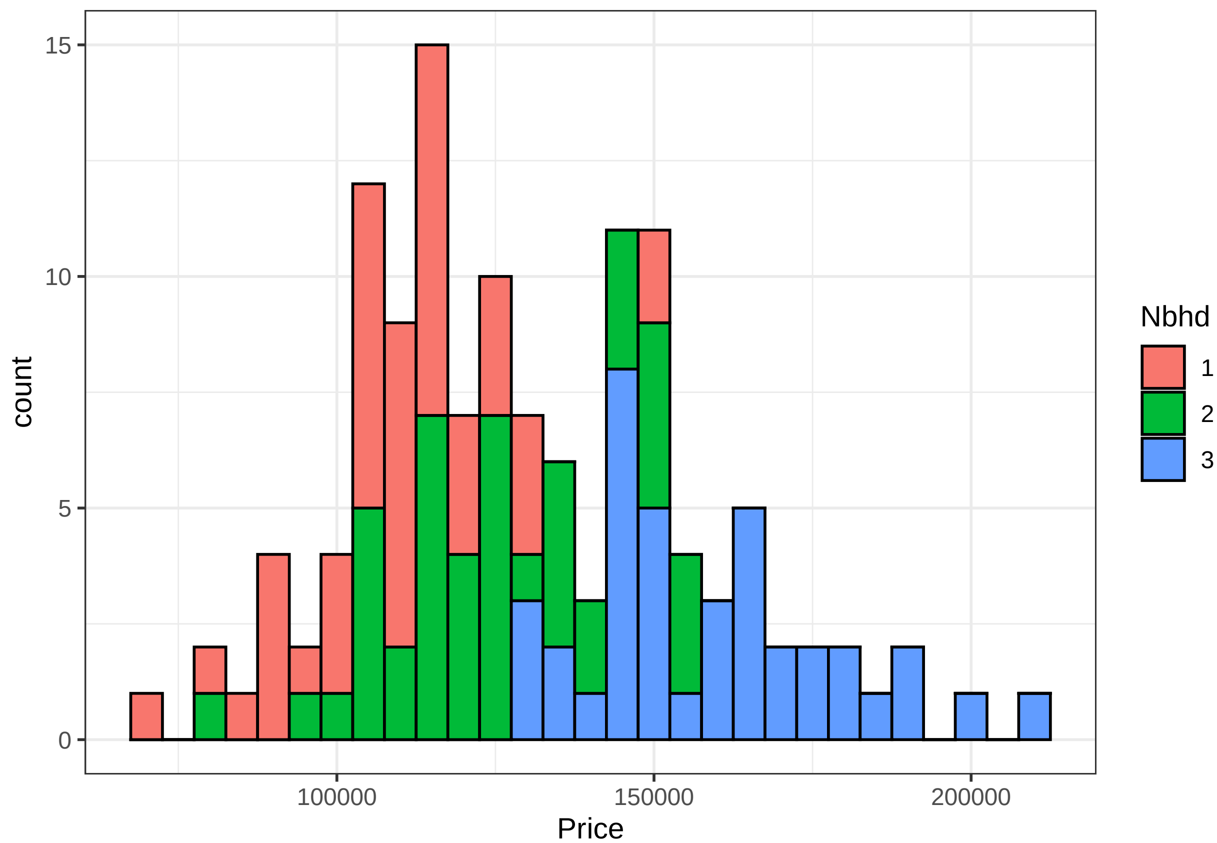
```
##
## Call:
## lm(formula = Price ~ ., data = MidCity)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -27897.8  -6074.8    -48.7   5551.8  27536.4
##
## Coefficients:
```

```
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2037.726   8911.501   0.229 0.819524
## Home           -11.456     25.387  -0.451 0.652616
## Nbhd2        -1729.613   2433.756  -0.711 0.478675
## Nbhd3        20534.706   3176.051   6.465 2.33e-09 ***
## Offers       -8350.128   1103.693  -7.566 8.96e-12 ***
## SqFt            53.634      5.926   9.051 3.30e-15 ***
## BrickYes     17313.540   1988.548   8.707 2.12e-14 ***
## Bedrooms      4136.461   1621.775   2.551 0.012023 *
## Bathrooms     7975.157   2133.831   3.737 0.000287 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10050 on 119 degrees of freedom
## Multiple R-squared:  0.8688, Adjusted R-squared:   0.86
## F-statistic: 98.54 on 8 and 119 DF,  p-value: < 2.2e-16
```

```r
library(ggplot2)

ggplot(MidCity,aes(Price)) +
    geom_histogram(aes(fill=Brick),color='black',binwidth=5000) + theme_bw()
```




```r
ggplot(MidCity,aes(Price)) +
    geom_histogram(aes(fill=Nbhd),color='black',binwidth=5000) + theme_bw()
```



3)

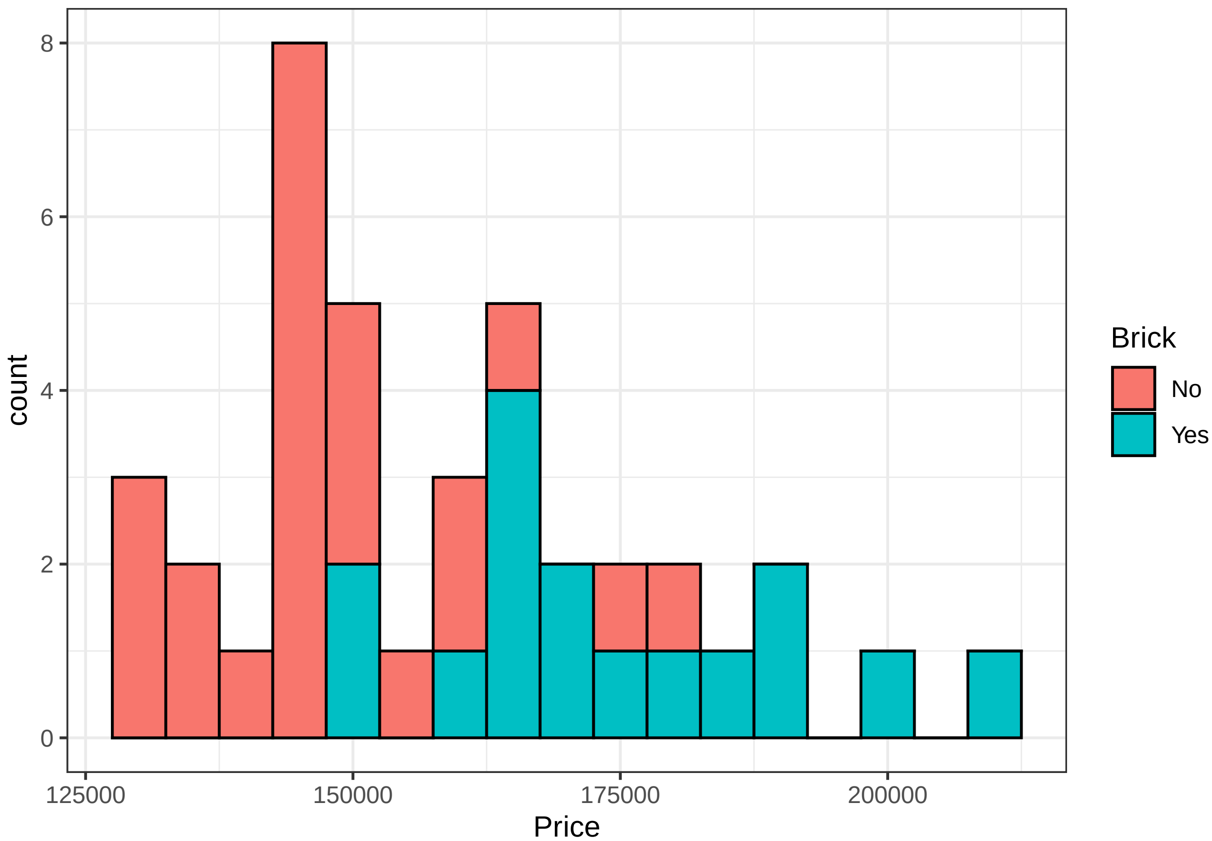
Regression for an interaction term between brick and neighboorhood suggests that brick homes in neighboorhood three carry significantly more increase in value than elsewhere.

```
MidCity$Nbhd <- as.factor(MidCity$Nbhd)
priceModel <- lm(Price ~. + Nbhd*Brick, data = MidCity)
summary(priceModel)
```

```
##
## Call:
## lm(formula = Price ~ . + Nbhd * Brick, data = MidCity)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -27843.9  -5544.3   -526.9   4167.3  28237.8
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)     3593.645   8860.065   0.406  0.68578
## Home             -12.410     24.975  -0.497  0.62020
## Nbhd2          -1527.046   2721.268  -0.561  0.57577
## Nbhd3          16807.264   3466.191   4.849 3.86e-06 ***
## Offers         -8470.621   1086.489  -7.796 2.91e-12 ***
## SqFt              54.427      5.866   9.278 1.10e-15 ***
## BrickYes       12033.113   4097.033   2.937  0.00399 **
## Bedrooms        4660.752   1608.651   2.897  0.00449 **
```

```
## Bathrooms        6554.909   2176.681   3.011  0.00319 **
## Nbhd2:BrickYes  2781.540   5090.237   0.546  0.58580
## Nbhd3:BrickYes 12019.217   5360.949   2.242  0.02685 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9879 on 117 degrees of freedom
## Multiple R-squared:  0.8755, Adjusted R-squared:  0.8648
## F-statistic: 82.24 on 10 and 117 DF,  p-value: < 2.2e-16
```

```
ggplot(subset(MidCity, Nbhd == 3),aes(Price)) +
    geom_histogram(aes(fill=Brick),color='black',binwidth=5000) + theme_bw()
```



4)

This is possible, and the coefficient for "NbhdAgeOld" suggests that there is not much of a difference (in terms of these variables) between neighborhood one and two.

```
MidCity$NbhdAge <- ifelse(MidCity$Nbhd == 3, 'New', 'Old')

NewOldModel <- lm(Price ~. - Nbhd, data = MidCity)
summary(NewOldModel)
```

```
##
## Call:
## lm(formula = Price ~ . - Nbhd, data = MidCity)
##
```

```
## Residuals:
##      Min       1Q   Median       3Q      Max
## -27191.9  -6372.7   -154.2   5739.9  26880.7
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  24979.54    9693.99   2.577  0.01118 *
## Home            -8.68      25.03  -0.347  0.72940
## Offers       -8061.22    1023.98  -7.872 1.73e-12 ***
## SqFt            52.56       5.72   9.190 1.46e-15 ***
## BrickYes     17051.46    1950.02   8.744 1.64e-14 ***
## Bedrooms      3971.91    1601.85   2.480  0.01454 *
## Bathrooms     7874.35    2124.72   3.706  0.00032 ***
## NbhdAgeOld  -21929.82    2491.57  -8.802 1.20e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10030 on 120 degrees of freedom
## Multiple R-squared:  0.8683, Adjusted R-squared:  0.8606
## F-statistic:   113 on 7 and 120 DF,  p-value: < 2.2e-16
```

Problem 3

1)

While there may be a significant correlation found between a regression of "Crime" and "Police", this does not account cor other causal factors that are correlated with either of these. In other words, there may be another variable, highly correlated with our predictor, that is actually causing the response variable.

2)

The researchers from UPenn were trying to determine if crime were simply determined by the avilible number of victims, measured by the addition of the additional variable of METRO ridership. Column one of the table indicates that the category of "High Alert" considered on its own is significant at the alpha = .05 level.

Column two of the table considers both the category of "High Alert" and the logarithim of "midday ridership" as predictors. While the riddership variable is significant at the higher alpha = .01 level, the alert category is still significant at the alpha = .05 level.

3)

Controlling for METRO ridership allows us to stratify our predicts based upon the number of people utilizing public transport, an indicator of how many people could potentially be victims of crime. The inital pypothesis was that fewer people would utilize public transport under a high alert, which showed that that this was actually incorrect. In Wheelan's words "They checked that hypothesis by looking at ridership levels on the Metro system, and they actually were not diminished on high-terror days, so they suggested the number of victims was largely unchanged."

Considering the results of the second column of table two, it appears that the presumed presence of more police under an increased alert did significantly reduce crime, even when controling for a measure of availible victims by including METRO ridership.

4)

This table further quantifies the effect of a high alert on the stratification of D.C's first district and all other areas. Here we see that controling for district by the interaction terms "High Alert x District 1" and "High Alert x Other Districts" that it appears that the effect of a high alert, while having a negative association in both cases, is only statistically significant for District 1 at the alpha = .01 level. Again note that midday ridership remains significant at the alpha = .01 level.

Problem 4

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

```
library(BART)
```

```
## Loading required package: nlme
```

```
## Loading required package: nnet
```

```
## Loading required package: survival
```

```
x = CAhousing[,1:8]
y = log(CAhousing$medianHouseValue)

set.seed(99) #MCMC, so set the seed
nd=200 # number of kept draws
burn=50 # number of burn in draws

n=length(y) #total sample size
set.seed(14) #
ii = sample(1:n,floor(.75*n)) # indices for train data, 75% of data
xtrain=x[ii,]; ytrain=y[ii] # training data
xtest=x[-ii,]; ytest=y[-ii] # test data
cat("train sample size is ",length(ytrain)," and test sample size is ",length(ytest),"\n")
```

```
## train sample size is  15480  and test sample size is  5160
```

```
set.seed(99)
bf_train = wbart(xtrain,ytrain)
```

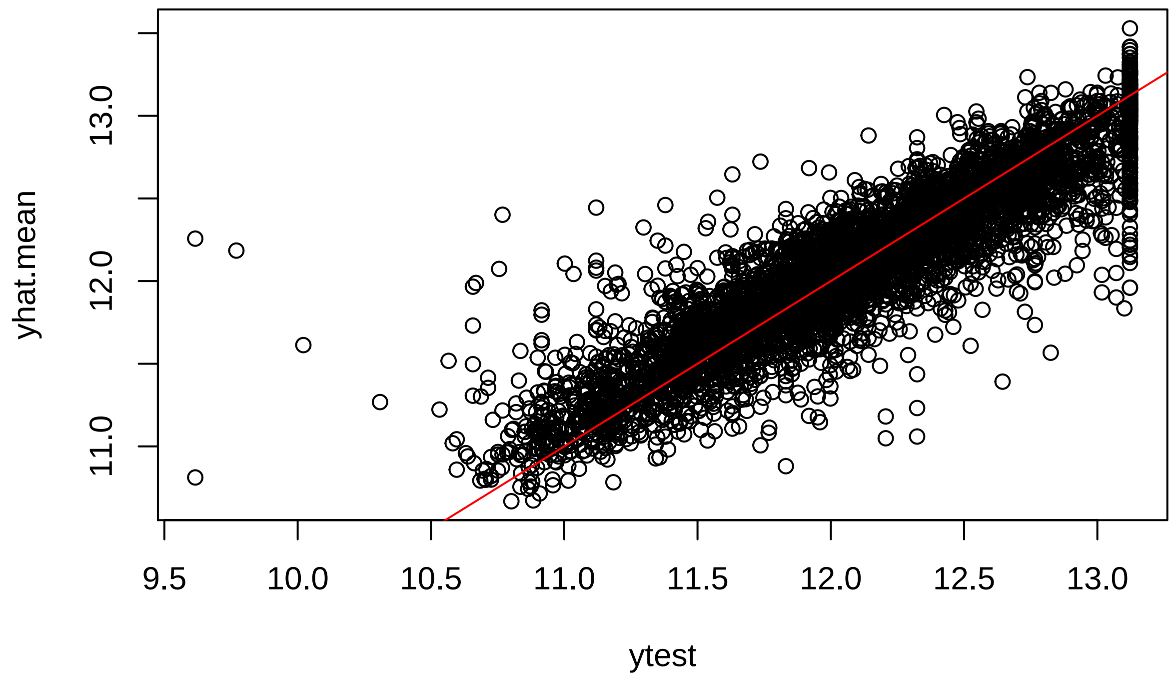
```
## *****Into main of wbart
## *****Data:
## data:n,p,np: 15480, 8, 0
## y1,yn: 0.245305, 0.307517
## x1,x[n*p]: -120.650000, 6.049400
## *****Number of Trees: 200
## *****Number of Cut Points: 100 ... 100
## *****burn and ndpost: 100, 1000
## *****Prior:beta,alpha,tau,nu,lambda: 2.000000,0.950000,0.061989,3.000000,0.022259
## *****sigma: 0.338036
## *****w (weights): 1.000000 ... 1.000000
## *****Dirichlet:sparse,theta,omega,a,b,rho,augment: 0,0,1,0.5,1,8,0
## *****nkeeptrain,nkeeptest,nkeeptestme,nkeeptreedraws: 1000,1000,1000,1000
## *****printevery: 100
## *****skiptr,skipte,skipteme,skiptreedraws: 1,1,1,1
##
## MCMC
```

```
## done 0 (out of 1100)
## done 100 (out of 1100)
## done 200 (out of 1100)
## done 300 (out of 1100)
## done 400 (out of 1100)
## done 500 (out of 1100)
## done 600 (out of 1100)
## done 700 (out of 1100)
## done 800 (out of 1100)
## done 900 (out of 1100)
## done 1000 (out of 1100)
## time: 81s
## check counts
## trcnt,tecnt,temecnt,treedrawscnt: 1000,0,0,1000
```

```
yhat = predict(bf_train,as.matrix(xtest))
```

```
## *****In main of C++ for bart prediction
## tc (threadcount): 1
## number of bart draws: 1000
## number of trees in bart sum: 200
## number of x columns: 8
## from x,np,p: 8, 5160
## ***using serial code
```

```
yhat.mean = apply(yhat,2,mean)



plot(ytest,yhat.mean)
abline(0,1,col=2)
```




```
comparisonBART <- data.frame(cbind(ytest, yhat.mean))
cat('BART RMSE = ', sqrt(mean((comparisonBART$ytest - comparisonBART$yhat.mean)**2)))
```

```
## BART RMSE =  0.2493312
```

It appears that the BART solution has a slightly higher RMSE than the random forest or boosting method provided in the notes.

Problem 5

```
library(MASS)
netData <- Boston
netData$chas <- as.numeric(netData$chas)

netTrain = sample(1:dim(netData)[1], dim(netData)[1]*.75)
netTest <- -netTrain
netTrainData <- netData[netTrain, ]
netTestData <- netData[-netTest, ]

maxs <- as.numeric(apply(netData, 2, max))
mins <- as.numeric(apply(netData, 2, min))

netScale <- scale(netData, center = mins, scale = maxs - mins)

scaledTrain <- netScale[netTrain,]
scaledTest <- as.data.frame(netScale[-netTest,])
```

```r
f <- as.formula(medv ~ crim + zn + indus + chas + nox + rm + age + dis + rad + tax + ptratio + black + l
```

Single Layer Net

```r
#import function from Github so we can view a graph
library(devtools)
```

```
## Loading required package: usethis

## Registered S3 method overwritten by 'cli':
##   method     from
##   print.tree tree
```

```r
source_url('https://gist.githubusercontent.com/fawda123/7471137/raw/466c1474d0a505ff044412703516c34f1a4
```

```
## SHA-1 hash of file is 74c80bd5ddbc17ab3ae5ece9c0ed9beb612e87ef
```

```r
library(nnet)
library(reshape)
```

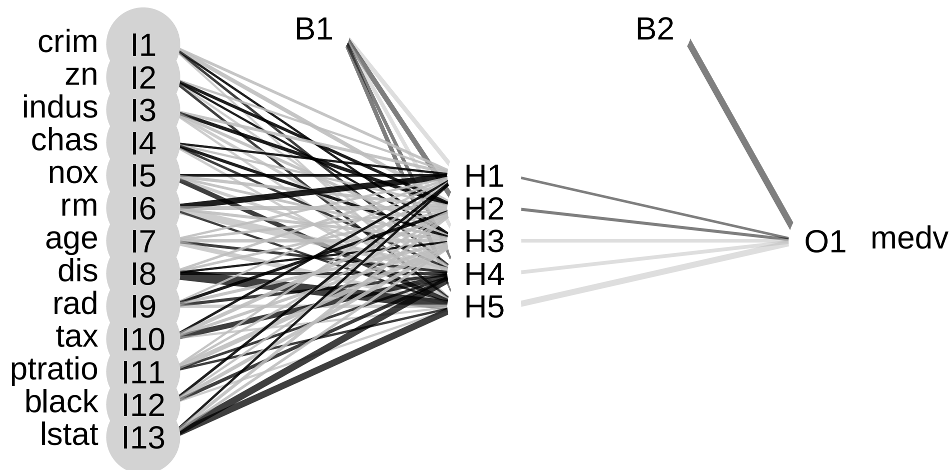
```
##
## Attaching package: 'reshape'

## The following object is masked from 'package:class':
##
##     condense

## The following object is masked from 'package:Matrix':
##
##     expand
```

```r
#arbitrary size selection, we'll cross validate to select properly
oneLayerNet <- nnet(formula = f, data = scaledTrain, size = 5)
```

```
## # weights:  76
## initial  value 45.640136
## iter  10 value 4.347692
## iter  20 value 2.894910
## iter  30 value 2.018930
## iter  40 value 1.530016
## iter  50 value 1.360005
## iter  60 value 1.228992
## iter  70 value 1.151390
## iter  80 value 1.125259
## iter  90 value 1.111241
## iter 100 value 1.097115
## final  value 1.097115
## stopped after 100 iterations
```

```r
plot.nnet(oneLayerNet, alpha.val = 0.5, circle.col = list('lightgray', 'white'))
```

```
## Loading required package: scales
```



Cross validation seems to suggest approximately seven hidden units and a weight decay of .005

```
library(caret)
```

```
## Loading required package: lattice
```

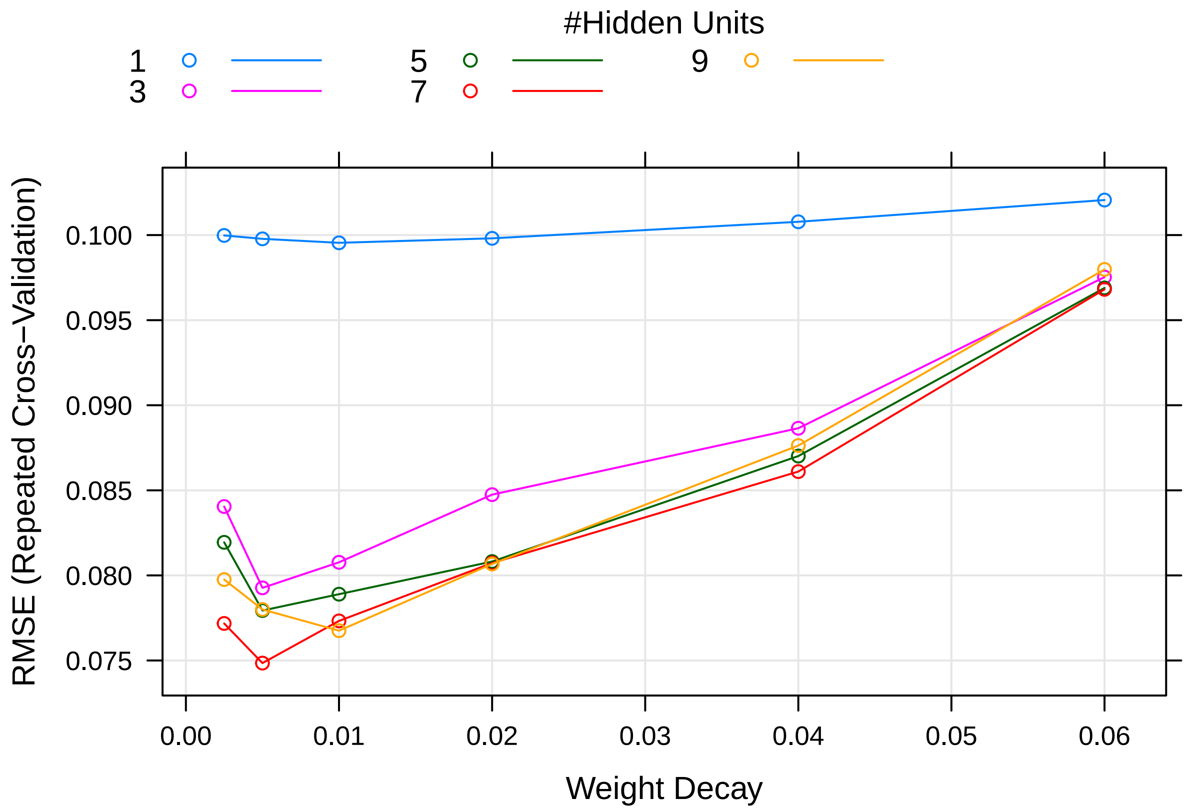
```
##
## Attaching package: 'caret'
```

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

```
## The following object is masked from 'package:pls':
##
##     R2
```

```
cvCtrl <- trainControl(method="repeatedcv", repeats=3)

nnetGrid <- expand.grid(.decay = c(.0025, .005, .01, .02, .04, .06), .size = c(1, 3, 5, 7, 9))

modFit_1 <- train(f, method="nnet", trControl=cvCtrl, data=scaledTrain, trace=FALSE, maxit=1000, linout
```

```
plot(modFit_1)
```



Multiple Layer Net (just for fun!)

```
library(neuralnet)
nn <- neuralnet(f,data=scaledTrain,hidden=c(5, 5),linear.output=T)
```

```
plot(nn)
```

```
#linear for comparison
BostonLinear <- glm(medv~., data=netTrainData)
BostonLinearPredict <- predict(BostonLinear,netTrainData)
LinearMSE <- sum((BostonLinearPredict - netTestData$medv)^2)/nrow(netTestData)

nnPredict <- compute(nn,scaledTest[,1:13])
nnPredictScaled <- nnPredict$net.result*(max(netData$medv)-min(netData$medv))+min(netData$medv)
testNNr <- (scaledTest$medv)*(max(netData$medv)-min(netData$medv))+min(netData$medv)
nnMSE <- sum((testNNr - nnPredictScaled)^2)/nrow(scaledTest)

print(paste(LinearMSE,nnMSE))
```

```
## [1] "24.5288325402381 4.89639151458281"
```

```
df <- data.frame('Layer1'=integer(), 'Layer2' = integer(), 'NetMSE' = numeric())

for (x in 1:5){
  for (y in 1:5){
    nn <- neuralnet(f,data=scaledTrain,hidden=c(x, y),linear.output=T)
```

```r
    nnPredict <- compute(nn,scaledTest[,1:13])
    nnPredictScaled <- nnPredict$net.result*(max(netData$medv)-min(netData$medv))+min(netData$medv)
    testNNr <- (scaledTest$medv)*(max(netData$medv)-min(netData$medv))+min(netData$medv)
    nnMSE <- sum((testNNr - nnPredictScaled)^2)/nrow(scaledTest)

    df[nrow(df) + 1,] = list(x, y, nnMSE)

    #print(nnMSE)
  }
}

print(df[which.min(df$NetMSE),])
```

```
##    Layer1 Layer2   NetMSE
## 24      5      4 3.926051
```

Problem 6

My main contribution to our group project was the creation of a LASSO model and several corresponding confusion matrices. After an initial logistic regression by one of my group members, I created a LASSO medel, cross validated across a range of lanbda values. This LASSO regression resulted in a model that had a similiar overall accuracy, but made significant improvements to the proportion of false negative and false positive results.

We also used this LASSO as an indicator of what factors were most significant in our regression. I then used this to create a second logisitic model, but did not suceed in producing better results than the LASSO I created.

Finally, I also calulated the accuracy and confusion matrix for a random forest model that one of my team members created.

Outside of our code, I drafted the slides and portion of the report correspondiong to the particuliar analysis I had done for the project.