Predictive_Models_Exam_Truett_Bloxsom

Chapter 2 Question 10

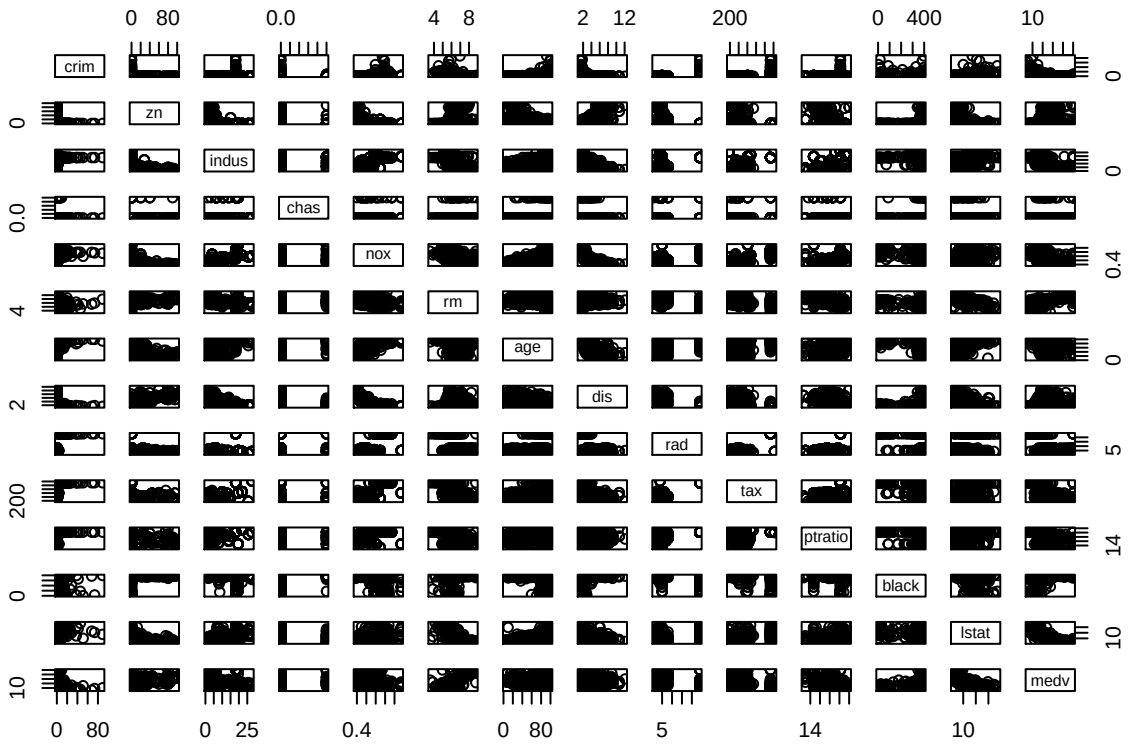
```
library(MASS)
dim(Boston)
```

```
## [1] 506  14
```

```
attach(Boston)
#?Boston
```

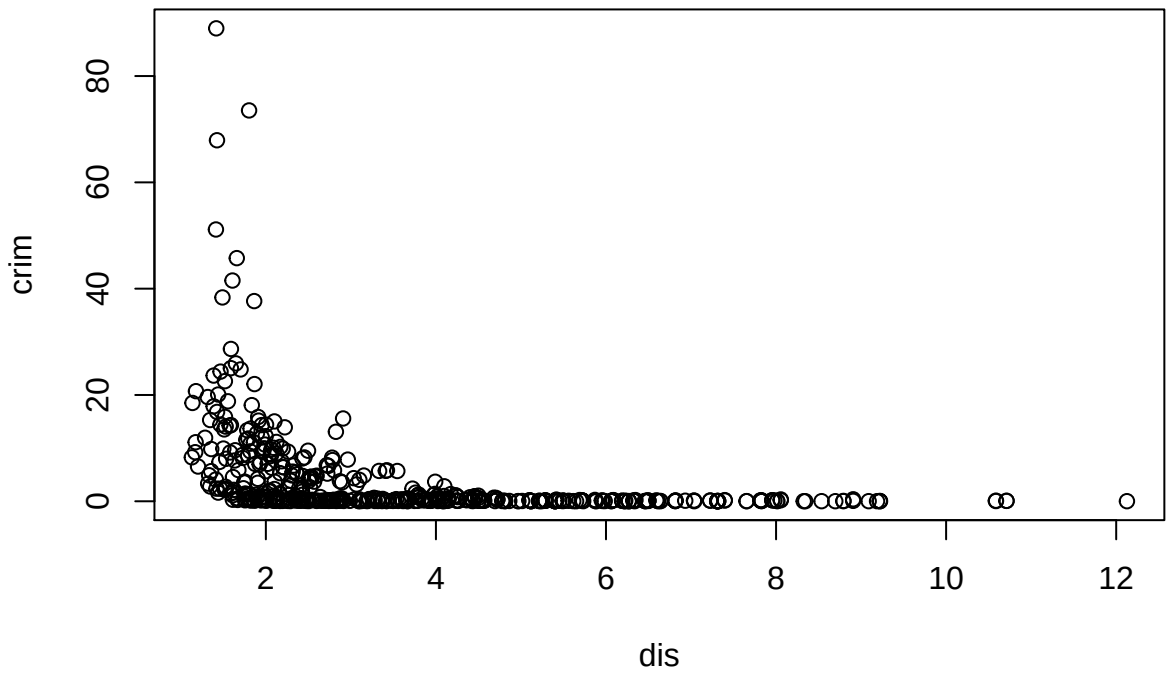
(a) there are 506 rows and 14 columns. Each row represents data on a particular suburb, and the columns represent data for one variable for all the suburbs.

```
pairs(~., data = Boston)
```

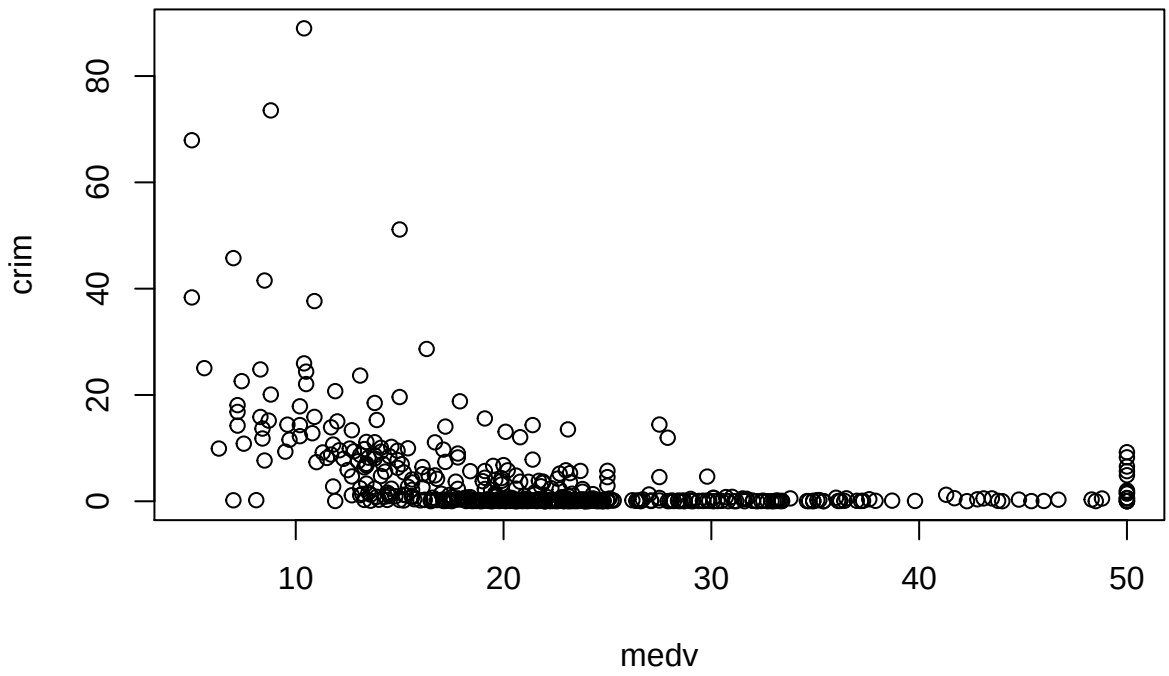


(b) there seems to be a few pairs of variables that are associated with eachother like lstat and medv or dis and nox

```
plot(dis, crim)
```

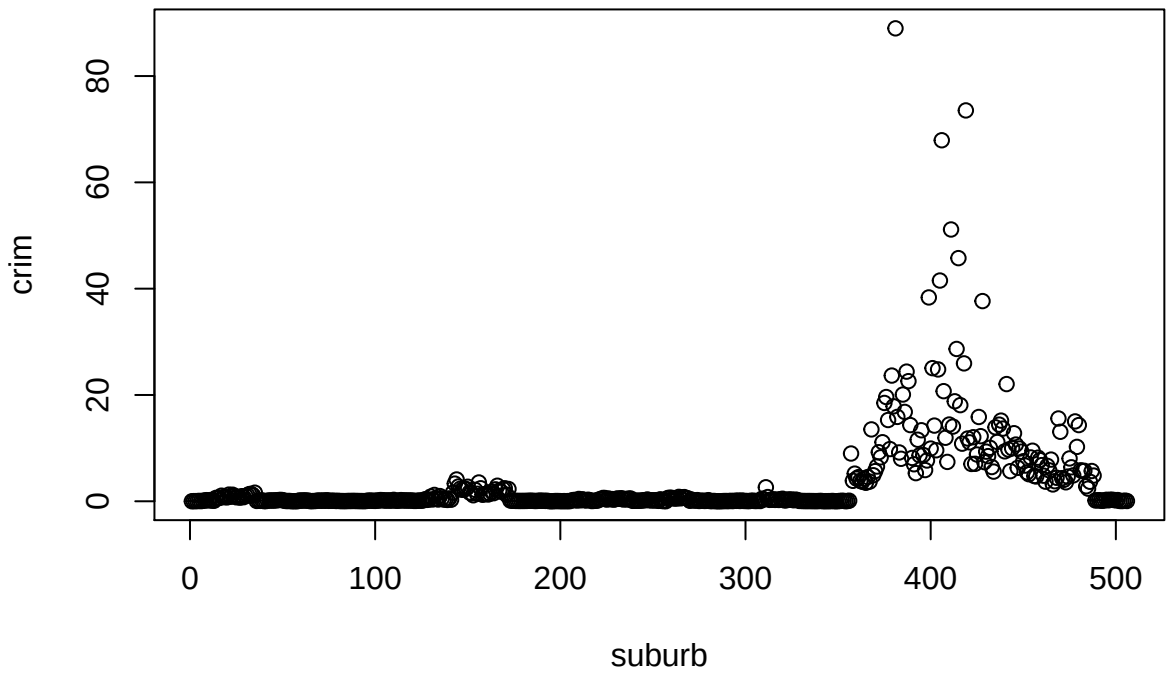



```
plot(medv, crim)
```

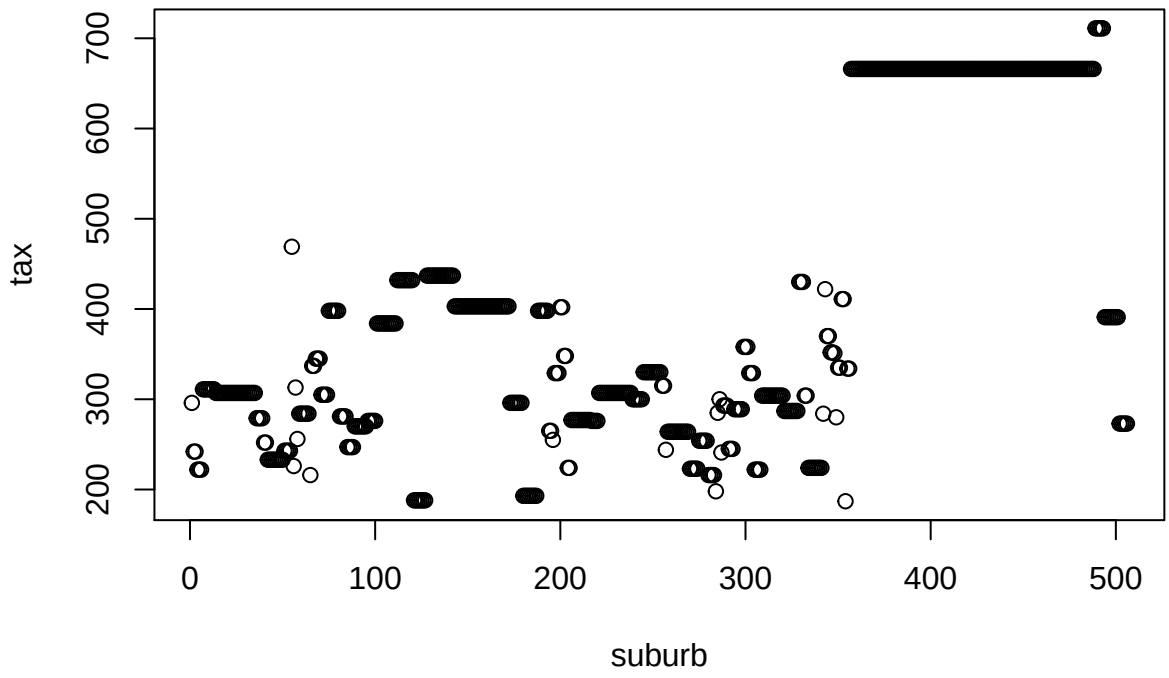


(c) There seems to be a relationship between crime rate and distance from epmloyments centers with crime increasing as the distance to the employments centers decreases there is also a relationship between crime rate and median house value with crime decreasing as the median house value increases.

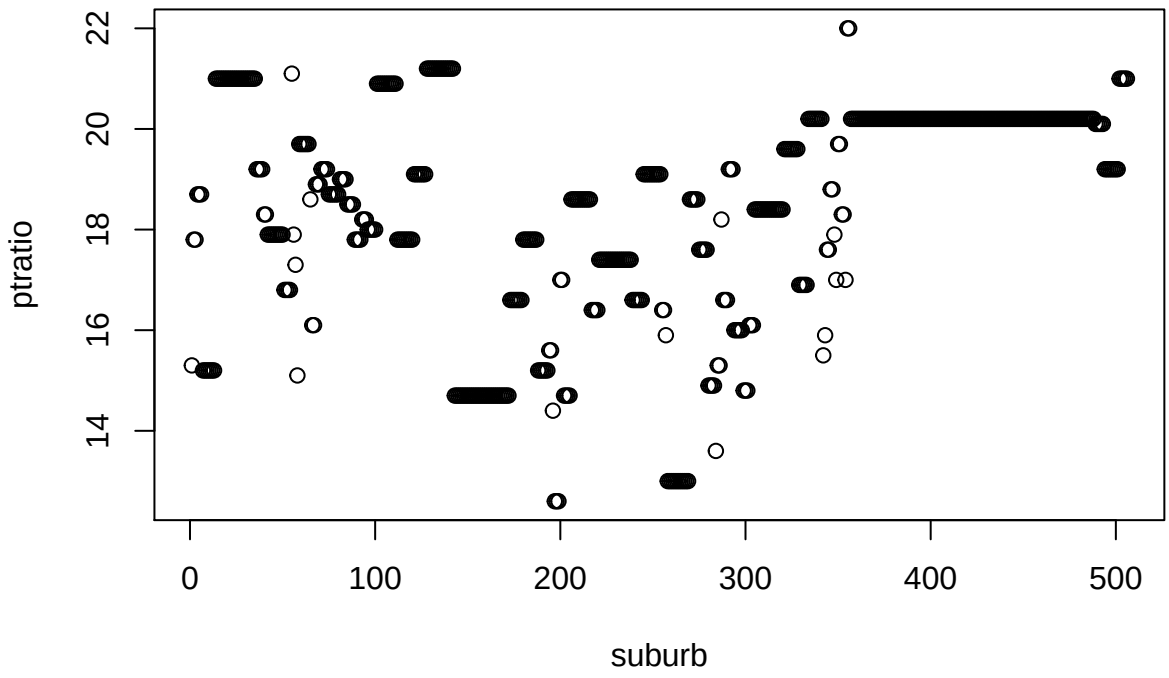
```
plot(crim, xlab = "suburb")
```




```
plot(tax, xlab = "suburb")
```




```
plot(ptratio, xlab = "suburb")
```



(d) the suburbs indexed from around 375 to 475 seem to have higher crime rates than the rest of the dataset. For these same suburbs, the tax rate is high and the pupil-teacher ratio is also high. the crime rate ranges from 0 to 100. The tax rate ranges from 0 to 700+. the pupil-teacher raion ranges from 12 to 22.

```
length(which(chas == 1))
```

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

(e) 35 suburbs bound the charles river

```
mean(ptratio)
```

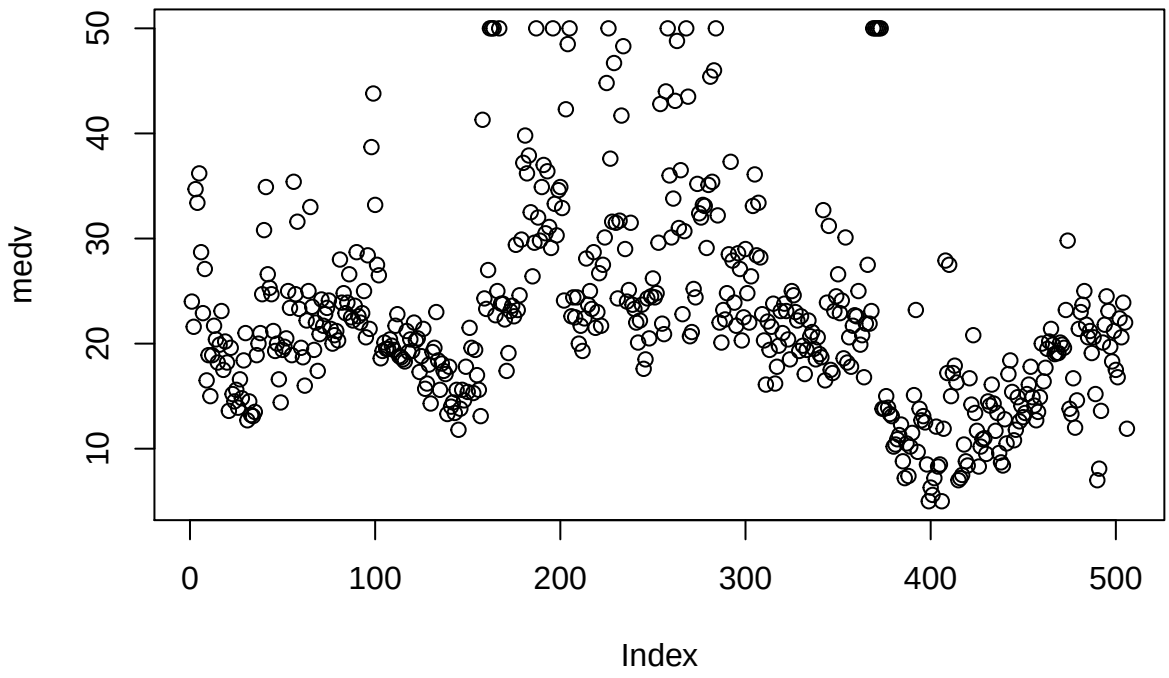
```
## [1] 18.45553
```

(f) the mean pupil-teacher ratio is 18.46

```
min(Boston[,14])
```

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

```
plot(medv)
```




```
Boston[406,]
```

```
##        crim zn indus chas   nox    rm age    dis rad tax ptratio  black
## 406 67.9208  0  18.1    0 0.693 5.683 100 1.4254  24 666    20.2 384.97
##     lstat medv
## 406 22.98    5
```

```
summary(Boston)
```

```
##       crim                zn              indus            chas
##  Min.   : 0.00632   Min.   :  0.00   Min.   : 0.46   Min.   :0.00000
##  1st Qu.: 0.08204   1st Qu.:  0.00   1st Qu.: 5.19   1st Qu.:0.00000
##  Median : 0.25651   Median :  0.00   Median : 9.69   Median :0.00000
##  Mean   : 3.61352   Mean   : 11.36   Mean   :11.14   Mean   :0.06917
##  3rd Qu.: 3.67708   3rd Qu.: 12.50   3rd Qu.:18.10   3rd Qu.:0.00000
##  Max.   :88.97620   Max.   :100.00   Max.   :27.74   Max.   :1.00000
##       nox               rm             age              dis
##  Min.   :0.3850   Min.   :3.561   Min.   :  2.90   Min.   : 1.130
##  1st Qu.:0.4490   1st Qu.:5.886   1st Qu.: 45.02   1st Qu.: 2.100
##  Median :0.5380   Median :6.208   Median : 77.50   Median : 3.207
##  Mean   :0.5547   Mean   :6.285   Mean   : 68.57   Mean   : 3.795
##  3rd Qu.:0.6240   3rd Qu.:6.623   3rd Qu.: 94.08   3rd Qu.: 5.188
##  Max.   :0.8710   Max.   :8.780   Max.   :100.00   Max.   :12.127
##       rad              tax           ptratio          black
##  Min.   : 1.000   Min.   :187.0   Min.   :12.60   Min.   :  0.32
```

```
##  1st Qu.: 4.000   1st Qu.:279.0   1st Qu.:17.40   1st Qu.:375.38
##  Median : 5.000   Median :330.0   Median :19.05   Median :391.44
##  Mean   : 9.549   Mean   :408.2   Mean   :18.46   Mean   :356.67
##  3rd Qu.:24.000   3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:396.23
##  Max.   :24.000   Max.   :711.0   Max.   :22.00   Max.   :396.90
##      lstat            medv
##  Min.   : 1.73   Min.   : 5.00
##  1st Qu.: 6.95   1st Qu.:17.02
##  Median :11.36   Median :21.20
##  Mean   :12.65   Mean   :22.53
##  3rd Qu.:16.95   3rd Qu.:25.00
##  Max.   :37.97   Max.   :50.00
```

(g) the 406th suburb has the lowest median value of owner-occupied homes with a value of 5. The Boston[406,] output summaizes the values of the other predictors. The crime in suburb 406, 67, is much higher than the average, 3.6. 406 has more non-retial busniess, 18.1, than the average at 11.1. 406 has a higher nitrogen oxides consentration, .69, than the avergae at .55. 406 has fewer rooms, 5.6, than the average at 6.2. 406 has a higher age of housing, 100, than the average at 68. 406 is closer to employment centers, 1.4, than the average at 3.8. 406 is much farther away from highways, 24, than the average at 9.5. 406 has much higher tax rate, 666, than the average at 408. 406 has a higher pupil-teacher ratio, 20.2, than the average at 18.46 406 has a higher porportion of blacks, 385, than the average at 357. 406 has a much higher lower status, 23, than the average at 13. The 406th suburb seems to be a low-income/poor neighborhood with high pollution, high crime, and older population compered to the average suburb in boston.

```
sum(rm > 7)
```

```
## [1] 64
```

```
sum(rm > 8)
```

```
## [1] 13
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following object is masked from 'package:MASS':
##
##     select
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
rm8 = filter(Boston, rm > 8)
summary(rm8)
```

```
##       crim               zn            indus             chas
##  Min.   :0.02009   Min.   : 0.00   Min.   : 2.680   Min.   :0.0000
##  1st Qu.:0.33147   1st Qu.: 0.00   1st Qu.: 3.970   1st Qu.:0.0000
##  Median :0.52014   Median : 0.00   Median : 6.200   Median :0.0000
##  Mean   :0.71879   Mean   :13.62   Mean   : 7.078   Mean   :0.1538
##  3rd Qu.:0.57834   3rd Qu.:20.00   3rd Qu.: 6.200   3rd Qu.:0.0000
##  Max.   :3.47428   Max.   :95.00   Max.   :19.580   Max.   :1.0000
##       nox               rm             age              dis
##  Min.   :0.4161   Min.   :8.034   Min.   : 8.40   Min.   :1.801
##  1st Qu.:0.5040   1st Qu.:8.247   1st Qu.:70.40   1st Qu.:2.288
##  Median :0.5070   Median :8.297   Median :78.30   Median :2.894
##  Mean   :0.5392   Mean   :8.349   Mean   :71.54   Mean   :3.430
##  3rd Qu.:0.6050   3rd Qu.:8.398   3rd Qu.:86.50   3rd Qu.:3.652
##  Max.   :0.7180   Max.   :8.780   Max.   :93.90   Max.   :8.907
##       rad              tax           ptratio          black
##  Min.   : 2.000   Min.   :224.0   Min.   :13.00   Min.   :354.6
##  1st Qu.: 5.000   1st Qu.:264.0   1st Qu.:14.70   1st Qu.:384.5
##  Median : 7.000   Median :307.0   Median :17.40   Median :386.9
##  Mean   : 7.462   Mean   :325.1   Mean   :16.36   Mean   :385.2
##  3rd Qu.: 8.000   3rd Qu.:307.0   3rd Qu.:17.40   3rd Qu.:389.7
##  Max.   :24.000   Max.   :666.0   Max.   :20.20   Max.   :396.9
##      lstat            medv
##  Min.   :2.47   Min.   :21.9
##  1st Qu.:3.32   1st Qu.:41.7
##  Median :4.14   Median :48.3
##  Mean   :4.31   Mean   :44.2
##  3rd Qu.:5.12   3rd Qu.:50.0
##  Max.   :7.44   Max.   :50.0
```

```
summary(Boston)
```

```
##       crim                zn              indus             chas
##  Min.   : 0.00632   Min.   :  0.00   Min.   : 0.46   Min.   :0.00000
##  1st Qu.: 0.08204   1st Qu.:  0.00   1st Qu.: 5.19   1st Qu.:0.00000
##  Median : 0.25651   Median :  0.00   Median : 9.69   Median :0.00000
##  Mean   : 3.61352   Mean   : 11.36   Mean   :11.14   Mean   :0.06917
##  3rd Qu.: 3.67708   3rd Qu.: 12.50   3rd Qu.:18.10   3rd Qu.:0.00000
##  Max.   :88.97620   Max.   :100.00   Max.   :27.74   Max.   :1.00000
##       nox               rm              age              dis
##  Min.   :0.3850   Min.   :3.561   Min.   :  2.90   Min.   : 1.130
##  1st Qu.:0.4490   1st Qu.:5.886   1st Qu.: 45.02   1st Qu.: 2.100
##  Median :0.5380   Median :6.208   Median : 77.50   Median : 3.207
##  Mean   :0.5547   Mean   :6.285   Mean   : 68.57   Mean   : 3.795
##  3rd Qu.:0.6240   3rd Qu.:6.623   3rd Qu.: 94.08   3rd Qu.: 5.188
##  Max.   :0.8710   Max.   :8.780   Max.   :100.00   Max.   :12.127
##       rad              tax           ptratio          black
##  Min.   : 1.000   Min.   :187.0   Min.   :12.60   Min.   :  0.32
##  1st Qu.: 4.000   1st Qu.:279.0   1st Qu.:17.40   1st Qu.:375.38
##  Median : 5.000   Median :330.0   Median :19.05   Median :391.44
```

```
##  Mean   : 9.549   Mean   :408.2   Mean   :18.46   Mean   :356.67
##  3rd Qu.:24.000   3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:396.23
##  Max.   :24.000   Max.   :711.0   Max.   :22.00   Max.   :396.90
##      lstat            medv
##  Min.   : 1.73   Min.   : 5.00
##  1st Qu.: 6.95   1st Qu.:17.02
##  Median :11.36   Median :21.20
##  Mean   :12.65   Mean   :22.53
##  3rd Qu.:16.95   3rd Qu.:25.00
##  Max.   :37.97   Max.   :50.00
```

(h) 64 suburbs average more than 7 rooms per dwelling and 13 average more than 8 rooms per dwelling. The dwellings that average more than 8 rooms per dwelling have lower crime rates, lower non-retial, higher age, lower distance to highways, lower property tax rates, lower pupil-teacher ratio, higher porportion of blacks, lower lower status of the population, and higher median value of homes than the average suburb.

Chapter 3 Question 15

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

```
##
## Call:
## lm(formula = crim ~ zn, data = Boston)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -4.429 -4.222 -2.620  1.250 84.523
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.45369    0.41722  10.675  < 2e-16 ***
## zn          -0.07393    0.01609  -4.594 5.51e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.435 on 504 degrees of freedom
## Multiple R-squared:  0.04019,    Adjusted R-squared:  0.03828
## F-statistic:  21.1 on 1 and 504 DF,  p-value: 5.506e-06
```

```
lm.fit1 = lm(crim~indus, data = Boston)
summary(lm.fit1)
```

```
##
## Call:
## lm(formula = crim ~ indus, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -11.972  -2.698  -0.736   0.712  81.813
##
```

```
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.06374    0.66723  -3.093  0.00209 **
## indus        0.50978    0.05102   9.991  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.866 on 504 degrees of freedom
## Multiple R-squared:  0.1653, Adjusted R-squared:  0.1637
## F-statistic: 99.82 on 1 and 504 DF,  p-value: < 2.2e-16
```

```
lm.fit2 = lm(crim~chas, data = Boston)
summary(lm.fit2)
```

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

```
lm.fit3 = lm(crim~nox, data = Boston)
summary(lm.fit3)
```

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

```r
lm.fit4 = lm(crim~rm, data = Boston)
summary(lm.fit4)
```

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

```r
lm.fit5 = lm(crim~age, data = Boston)
summary(lm.fit5)
```

```
##
## Call:
## lm(formula = crim ~ age, data = Boston)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -6.789 -4.257 -1.230  1.527 82.849
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.77791    0.94398  -4.002 7.22e-05 ***
## age          0.10779    0.01274   8.463 2.85e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.057 on 504 degrees of freedom
## Multiple R-squared:  0.1244, Adjusted R-squared:  0.1227
## F-statistic: 71.62 on 1 and 504 DF,  p-value: 2.855e-16
```

```r
lm.fit6 = lm(crim~dis, data = Boston)
summary(lm.fit6)
```

```
##
## Call:
## lm(formula = crim ~ dis, data = Boston)
##
## Residuals:
```

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

```
lm.fit7 = lm(crim~rad, data = Boston)
summary(lm.fit7)
```

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

```
lm.fit8 = lm(crim~tax, data = Boston)
summary(lm.fit8)
```

```
##
## Call:
## lm(formula = crim ~ tax, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -12.513  -2.738  -0.194   1.065  77.696
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -8.528369   0.815809  -10.45   <2e-16 ***
## tax          0.029742   0.001847   16.10   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 6.997 on 504 degrees of freedom
## Multiple R-squared:  0.3396, Adjusted R-squared:  0.3383
## F-statistic: 259.2 on 1 and 504 DF,  p-value: < 2.2e-16
```

```
lm.fit9 = lm(crim~ptratio, data = Boston)
summary(lm.fit9)
```

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

```
lm.fit10 = lm(crim~black, data = Boston)
summary(lm.fit10)
```

```
##
## Call:
## lm(formula = crim ~ black, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -13.756  -2.299  -2.095  -1.296  86.822
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 16.553529   1.425903  11.609   <2e-16 ***
## black       -0.036280   0.003873  -9.367   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.946 on 504 degrees of freedom
## Multiple R-squared:  0.1483, Adjusted R-squared:  0.1466
## F-statistic: 87.74 on 1 and 504 DF,  p-value: < 2.2e-16
```

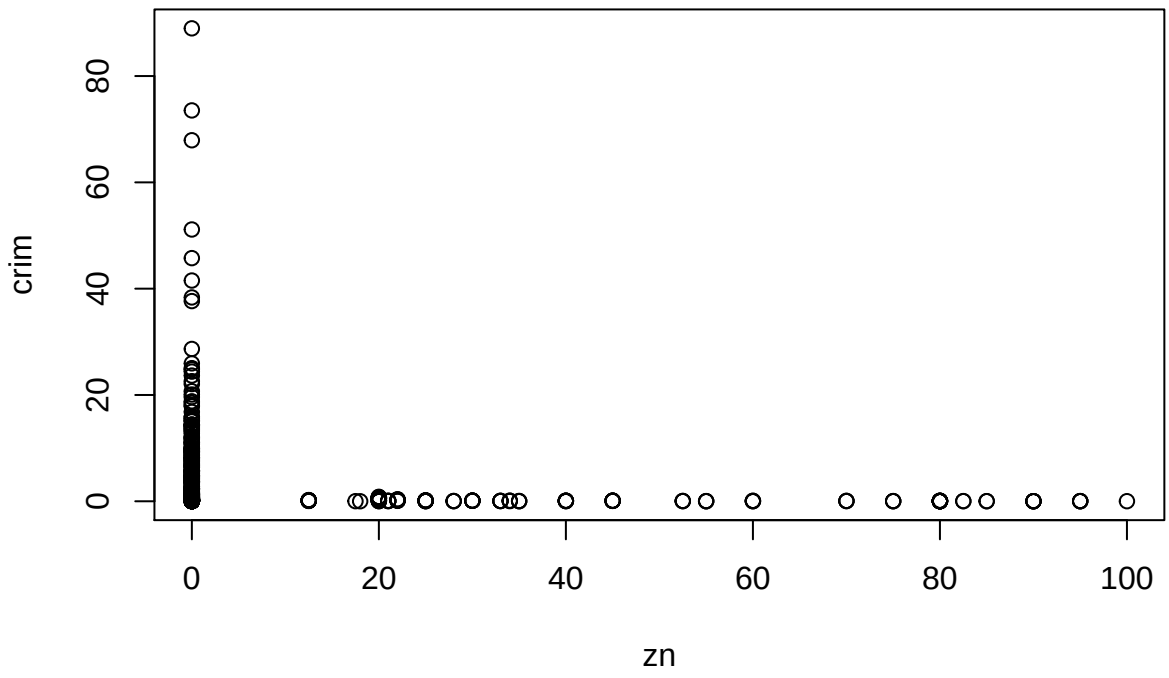
```
lm.fit11 = lm(crim~lstat, data = Boston)
summary(lm.fit11)
```

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

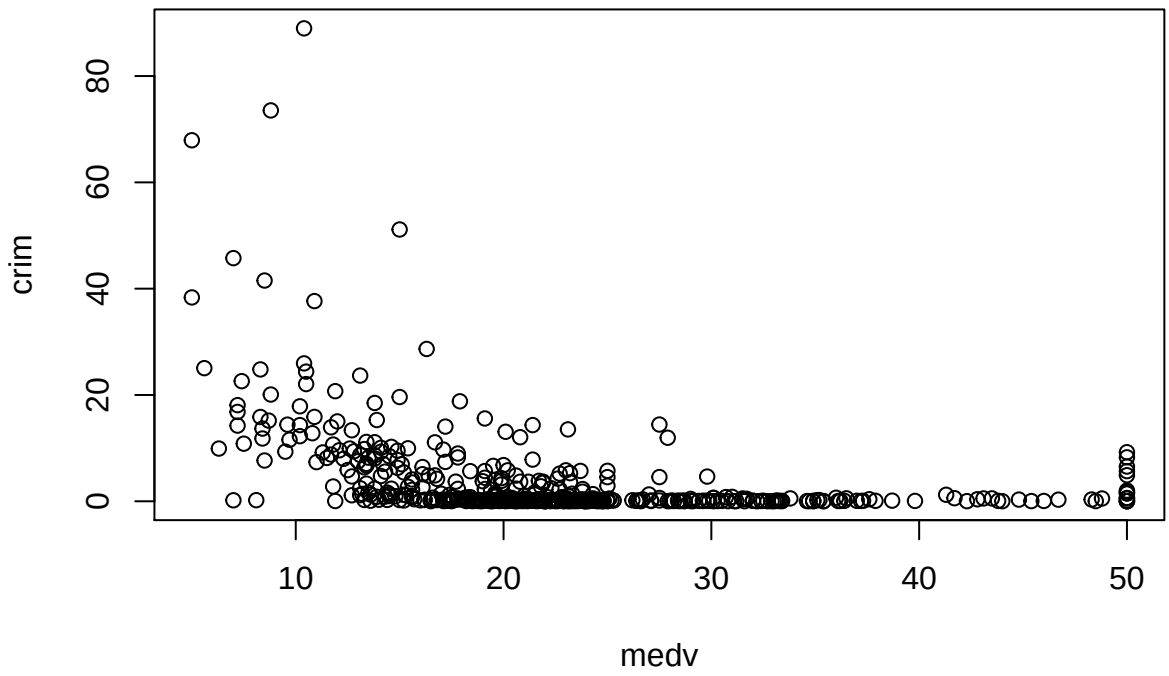
```
lm.fit12 = lm(crim~medv, data = Boston)
summary(lm.fit12)
```

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

```
plot(zn, crim)
```




```
plot(medv, crim)
```



(a) most of the predictors are highly significant with one exception, being by the Charles river is not significant. both of the plots show as zn or medv increase, crim decreases substantially.

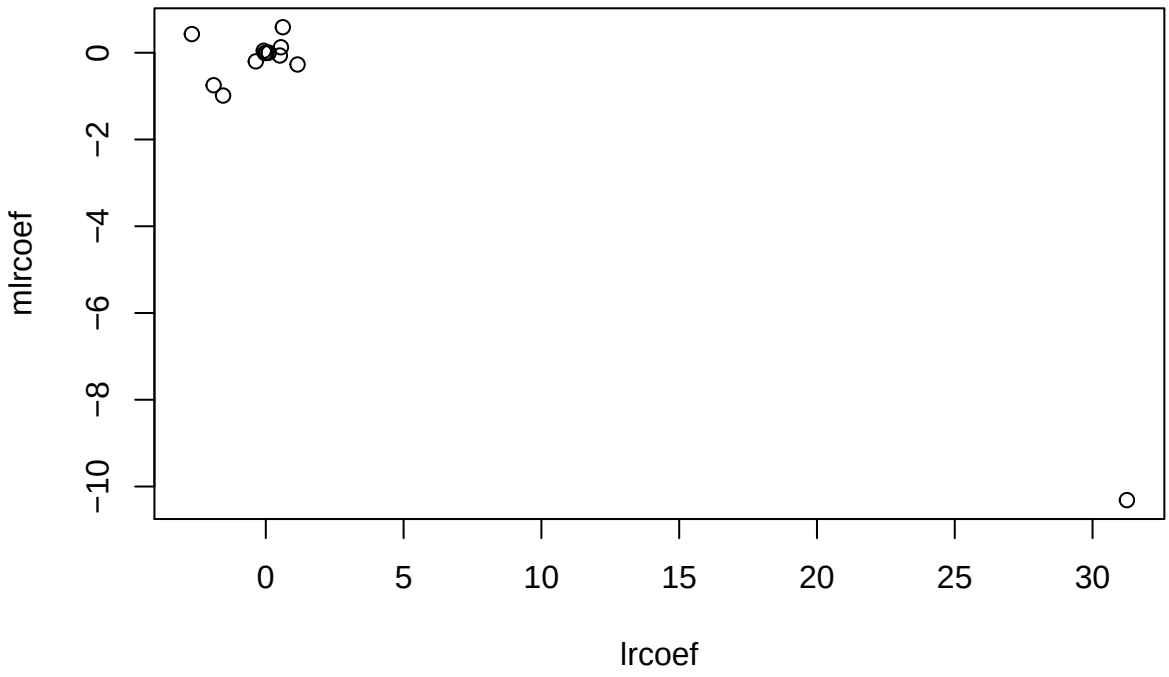
```
lm.fitm = lm(crim~., data = Boston)
summary(lm.fitm)
```

```
##
## Call:
## lm(formula = crim ~ ., data = Boston)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -9.924 -2.120 -0.353  1.019 75.051
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  17.033228   7.234903   2.354 0.018949 *
## zn            0.044855   0.018734   2.394 0.017025 *
## indus        -0.063855   0.083407  -0.766 0.444294
## chas         -0.749134   1.180147  -0.635 0.525867
## nox         -10.313535   5.275536  -1.955 0.051152 .
## rm            0.430131   0.612830   0.702 0.483089
## age           0.001452   0.017925   0.081 0.935488
## dis          -0.987176   0.281817  -3.503 0.000502 ***
## rad           0.588209   0.088049   6.680 6.46e-11 ***
## tax          -0.003780   0.005156  -0.733 0.463793
```

```
## ptratio      -0.271081   0.186450  -1.454 0.146611
## black        -0.007538   0.003673  -2.052 0.040702 *
## lstat         0.126211   0.075725   1.667 0.096208 .
## medv         -0.198887   0.060516  -3.287 0.001087 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.439 on 492 degrees of freedom
## Multiple R-squared:  0.454,  Adjusted R-squared:  0.4396
## F-statistic: 31.47 on 13 and 492 DF,  p-value: < 2.2e-16
```

(b)Many of the predictors are now insignificant. Distance from employment centers has a highly negative association with crime rate, and accessibility to highways has a highly positive association with crime rate. We can reject the null hypothesis for zn, nox, dis, rad, black, lstat, and medv at the 10% level.

```
lrcoef = c(coefficients(lm.fit)[2], coefficients(lm.fit1)[2],
           coefficients(lm.fit2)[2], coefficients(lm.fit3)[2],
           coefficients(lm.fit4)[2], coefficients(lm.fit5)[2],
           coefficients(lm.fit6)[2], coefficients(lm.fit7)[2],
           coefficients(lm.fit8)[2], coefficients(lm.fit9)[2],
           coefficients(lm.fit10)[2], coefficients(lm.fit11)[2],
           coefficients(lm.fit12)[2])
mlrcoef = coefficients(lm.fitm)[2:14]
plot(lrcoef, mlrcoef)
```



(c) Most of the simple and multiple regression coefficients are close to 0 which means they are similar to

each other. nox is the only outlier with the simple linear coefficient value being 31 and the multiple linear regression coefficent value being -10.

```
lm.fitq = lm(crim~poly(zn, 3), data = Boston)
summary(lm.fitq)
```

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

```
anova(lm.fit, lm.fitq)
```

```
## Analysis of Variance Table
##
## Model 1: crim ~ zn
## Model 2: crim ~ poly(zn, 3)
##   Res.Df   RSS Df Sum of Sq      F   Pr(>F)
## 1    504 35862
## 2    502 35187  2    674.56 4.8118 0.008512 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
lm.fitq1 = lm(crim~poly(indus, 3), data = Boston)
summary(lm.fitq1)
```

```
##
## Call:
## lm(formula = crim ~ poly(indus, 3), data = Boston)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -8.278 -2.514  0.054  0.764 79.713
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)        3.614      0.330  10.950  < 2e-16 ***
## poly(indus, 3)1   78.591      7.423  10.587  < 2e-16 ***
## poly(indus, 3)2  -24.395      7.423  -3.286  0.00109 **
## poly(indus, 3)3  -54.130      7.423  -7.292  1.2e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.423 on 502 degrees of freedom
## Multiple R-squared:  0.2597, Adjusted R-squared:  0.2552
## F-statistic: 58.69 on 3 and 502 DF,  p-value: < 2.2e-16
```

```
anova(lm.fit1, lm.fitq1)
```

```
## Analysis of Variance Table
##
## Model 1: crim ~ indus
## Model 2: crim ~ poly(indus, 3)
##   Res.Df   RSS Df Sum of Sq      F    Pr(>F)
## 1    504 31187
## 2    502 27662  2    3525.1 31.987 8.409e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
lm.fitq3 = lm(crim~poly(nox, 3), data = Boston)
summary(lm.fitq3)
```

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

```
anova(lm.fit3, lm.fitq3)
```

```
## Analysis of Variance Table
##
## Model 1: crim ~ nox
```

```
## Model 2: crim ~ poly(nox, 3)
##   Res.Df   RSS Df Sum of Sq      F    Pr(>F)
## 1    504 30742
## 2    502 26267  2    4474.6 42.758 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
lm.fitq4 = lm(crim~poly(rm, 3), data = Boston)
summary(lm.fitq4)
```

```
##
## Call:
## lm(formula = crim ~ poly(rm, 3), data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -18.485  -3.468  -2.221  -0.015  87.219
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    3.6135     0.3703   9.758  < 2e-16 ***
## poly(rm, 3)1 -42.3794     8.3297  -5.088 5.13e-07 ***
## poly(rm, 3)2  26.5768     8.3297   3.191  0.00151 **
## poly(rm, 3)3  -5.5103     8.3297  -0.662  0.50858
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.33 on 502 degrees of freedom
## Multiple R-squared:  0.06779,    Adjusted R-squared:  0.06222
## F-statistic: 12.17 on 3 and 502 DF,  p-value: 1.067e-07
```

```
anova(lm.fit4, lm.fitq4)
```

```
## Analysis of Variance Table
##
## Model 1: crim ~ rm
## Model 2: crim ~ poly(rm, 3)
##   Res.Df   RSS Df Sum of Sq      F   Pr(>F)
## 1    504 35567
## 2    502 34831  2    736.69 5.3088 0.005229 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
lm.fitq5 = lm(crim~poly(age, 3), data = Boston)
summary(lm.fitq5)
```

```
##
## Call:
## lm(formula = crim ~ poly(age, 3), data = Boston)
##
## Residuals:
##    Min     1Q Median     3Q    Max
```

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

```
anova(lm.fit5, lm.fitq5)
```

```
## Analysis of Variance Table
##
## Model 1: crim ~ age
## Model 2: crim ~ poly(age, 3)
##   Res.Df   RSS Df Sum of Sq     F    Pr(>F)
## 1    504 32714
## 2    502 30853  2      1861 15.14 4.125e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
lm.fitq6 = lm(crim~poly(dis, 3), data = Boston)
summary(lm.fitq6)
```

```
##
## Call:
## lm(formula = crim ~ poly(dis, 3), data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -10.757  -2.588   0.031   1.267  76.378
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)     3.6135     0.3259  11.087  < 2e-16 ***
## poly(dis, 3)1 -73.3886     7.3315 -10.010  < 2e-16 ***
## poly(dis, 3)2  56.3730     7.3315   7.689 7.87e-14 ***
## poly(dis, 3)3 -42.6219     7.3315  -5.814 1.09e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.331 on 502 degrees of freedom
## Multiple R-squared:  0.2778, Adjusted R-squared:  0.2735
## F-statistic: 64.37 on 3 and 502 DF,  p-value: < 2.2e-16
```

```r
anova(lm.fit6, lm.fitq6)
```

```
## Analysis of Variance Table
##
## Model 1: crim ~ dis
## Model 2: crim ~ poly(dis, 3)
##   Res.Df   RSS Df Sum of Sq     F    Pr(>F)
## 1    504 31977
## 2    502 26983  2    4994.5 46.46 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
lm.fitq7 = lm(crim~poly(rad, 3), data = Boston)
summary(lm.fitq7)
```

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

```r
anova(lm.fit7, lm.fitq7)
```

```
## Analysis of Variance Table
##
## Model 1: crim ~ rad
## Model 2: crim ~ poly(rad, 3)
##   Res.Df   RSS Df Sum of Sq      F  Pr(>F)
## 1    504 22745
## 2    502 22417  2    328.06 3.6733 0.02608 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
lm.fitq8 = lm(crim~poly(tax, 3), data = Boston)
summary(lm.fitq8)
```

```
##
## Call:
## lm(formula = crim ~ poly(tax, 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(tax, 3)1 112.6458     6.8537  16.436  < 2e-16 ***
## poly(tax, 3)2  32.0873     6.8537   4.682 3.67e-06 ***
## poly(tax, 3)3  -7.9968     6.8537  -1.167    0.244
## ---
## 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
```

```
anova(lm.fit8, lm.fitq8)
```

```
## Analysis of Variance Table
##
## Model 1: crim ~ tax
## Model 2: crim ~ poly(tax, 3)
##   Res.Df   RSS Df Sum of Sq     F    Pr(>F)
## 1    504 24674
## 2    502 23581  2    1093.5 11.64 1.144e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
lm.fitq9 = lm(crim~poly(ptratio, 3), data = Boston)
summary(lm.fitq9)
```

```
##
## Call:
## lm(formula = crim ~ poly(ptratio, 3), data = Boston)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -6.833 -4.146 -1.655  1.408 82.697
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)          3.614      0.361  10.008  < 2e-16 ***
## poly(ptratio, 3)1   56.045      8.122   6.901 1.57e-11 ***
## poly(ptratio, 3)2   24.775      8.122   3.050  0.00241 **
## poly(ptratio, 3)3  -22.280      8.122  -2.743  0.00630 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 8.122 on 502 degrees of freedom
## Multiple R-squared:  0.1138, Adjusted R-squared:  0.1085
## F-statistic: 21.48 on 3 and 502 DF,  p-value: 4.171e-13
```

```
anova(lm.fit9, lm.fitq9)
```

```
## Analysis of Variance Table
##
## Model 1: crim ~ ptratio
## Model 2: crim ~ poly(ptratio, 3)
##   Res.Df   RSS Df Sum of Sq      F    Pr(>F)
## 1    504 34222
## 2    502 33112  2    1110.2 8.4155 0.0002542 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
lm.fitq10 = lm(crim~poly(black, 3), data = Boston)
summary(lm.fitq10)
```

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

```
anova(lm.fit10, lm.fitq10)
```

```
## Analysis of Variance Table
##
## Model 1: crim ~ black
## Model 2: crim ~ poly(black, 3)
##   Res.Df   RSS Df Sum of Sq      F Pr(>F)
## 1    504 31823
## 2    502 31765  2    58.495 0.4622 0.6302
```

```r
lm.fitq11 = lm(crim~poly(lstat, 3), data = Boston)
summary(lm.fitq11)
```

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

```r
anova(lm.fit11, lm.fitq11)
```

```
## Analysis of Variance Table
##
## Model 1: crim ~ lstat
## Model 2: crim ~ poly(lstat, 3)
##   Res.Df   RSS Df Sum of Sq     F  Pr(>F)
## 1    504 29607
## 2    502 29221  2    386.39 3.319 0.03698 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
lm.fitq12 = lm(crim~poly(medv, 3), data = Boston)
summary(lm.fitq12)
```

```
##
## Call:
## lm(formula = crim ~ poly(medv, 3), data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -24.427  -1.976  -0.437   0.439  73.655
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)       3.614      0.292  12.374  < 2e-16 ***
## poly(medv, 3)1  -75.058      6.569 -11.426  < 2e-16 ***
```

```
## poly(medv, 3)2   88.086      6.569  13.409  < 2e-16 ***
## poly(medv, 3)3  -48.033      6.569  -7.312 1.05e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.569 on 502 degrees of freedom
## Multiple R-squared:  0.4202, Adjusted R-squared:  0.4167
## F-statistic: 121.3 on 3 and 502 DF,  p-value: < 2.2e-16
```

```
anova(lm.fit12, lm.fitq12)
```

```
## Analysis of Variance Table
##
## Model 1: crim ~ medv
## Model 2: crim ~ poly(medv, 3)
##   Res.Df   RSS Df Sum of Sq      F    Pr(>F)
## 1    504 31730
## 2    502 21663  2     10066 116.63 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

(d) there is evidence of non-linear association between the predictors and the response. the only predictors that do not have a significant f stat from the anova test is the binary variables black and indus.

Chapter 6 Question 9

```
set.seed(1)
library(ISLR)
sample = sample(nrow(College),nrow(College)*0.75)
train = College[sample,]
test = College[-sample,]
```

(a) the above commands answer part (a)

```
lm.fit = lm(Apps~., data = train)
lm.pred = predict(lm.fit,test)
cat("linear reg test MSE =",mean((test$Apps-lm.pred)^2))
```

```
## linear reg test MSE = 1384604
```

(b) the above commands answer part (b)

```
library(glmnet)
```

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

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

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

```
grid = 10^seq(10, -2, length = 100)
train.m = model.matrix(Apps~., data = train)
test.m = model.matrix(Apps~., data = test)
ridge.mod = glmnet(train.m, train$Apps, alpha = 0, lambda = grid)
cv.ridge = cv.glmnet(train.m, train$Apps, alpha = 0, lambda = grid)
bestlam.r = cv.ridge$lambda.min
ridge.pred = predict(ridge.mod, s = bestlam.r, newx = test.m)
cat("ridge test MSE =",mean((test$Apps - ridge.pred)^2) )
```

```
## ridge test MSE = 1384165
```

(c) the above commands answer part (c)

```
lasso.mod = glmnet(train.m, train$Apps, alpha = 1, lambda = grid)
cv.lasso = cv.glmnet(train.m, train$Apps, alpha = 1, lambda = grid)
bestlam.l = cv.lasso$lambda.min
lasso.pred = predict(lasso.mod, s = bestlam.l, newx = test.m)
cat("lasso test MSE =",mean((test$Apps - lasso.pred)^2))
```

```
## lasso test MSE = 1383288
```

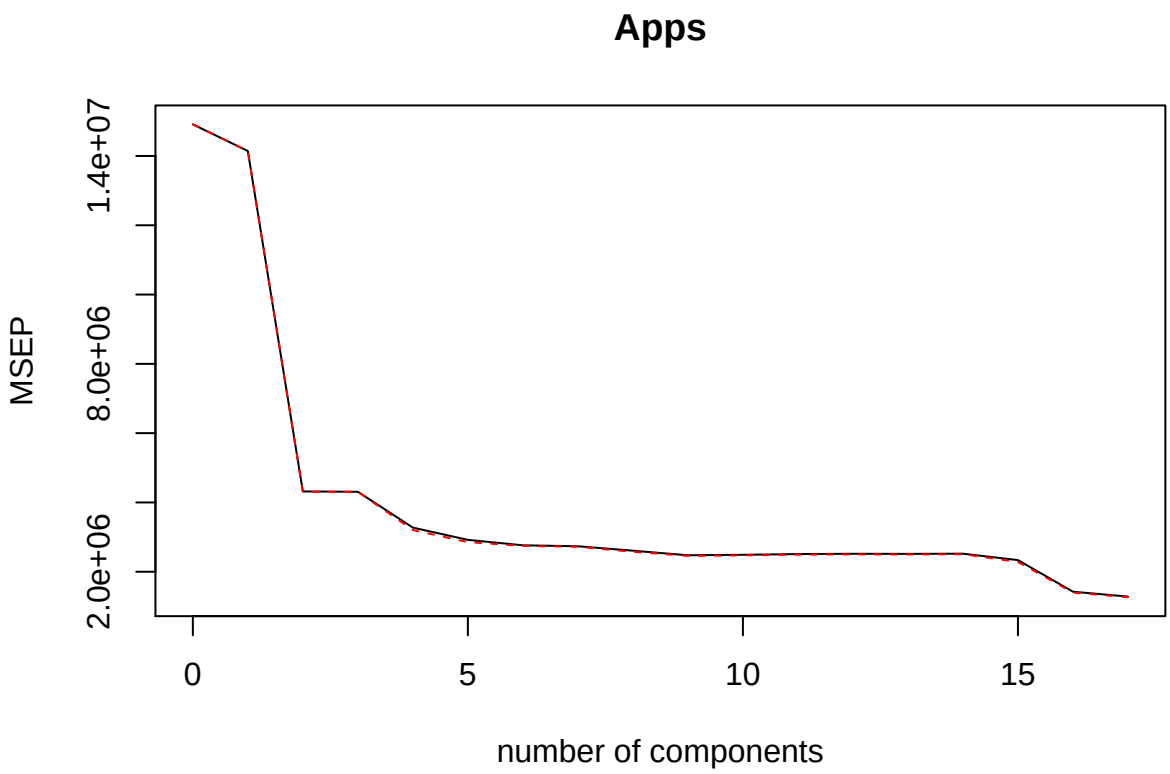
(d) the above commands answer part (d)

```
library(pls)
```

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

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

```
pcr.fit = pcr(Apps~., data = train, scale = TRUE, validation = "CV")
validationplot(pcr.fit, val.type = "MSEP")
```

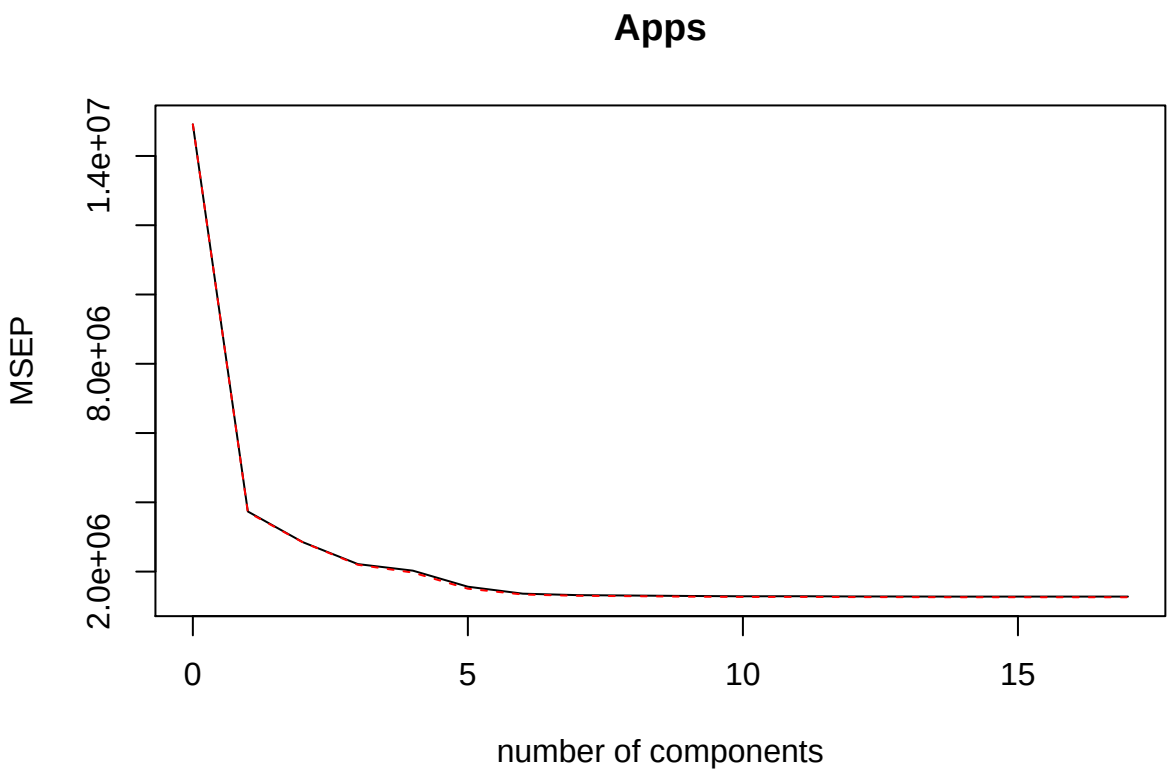



```
pcr.pred = predict(pcr.fit, test, ncomp = 17)
cat("PCR test MSE =",mean((test$Apps - pcr.pred)^2))
```

```
## PCR test MSE = 1384604
```

(e) the above commands answer part (e)

```
pls.fit = plsr(Apps~., data = train, scale = TRUE, validation = "CV")
validationplot(pls.fit, val.type = "MSEP")
```



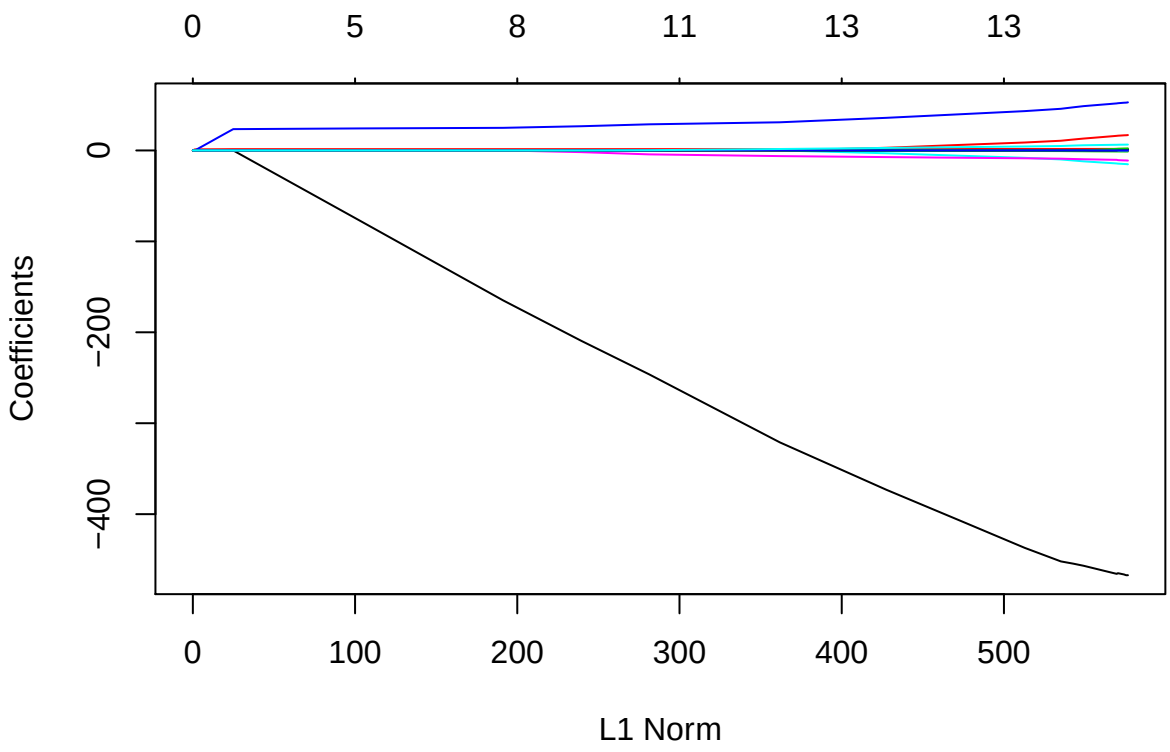

```
pls.pred = predict(pls.fit, test, ncomp = 9)
cat("PLS test MSE =",mean((test$Apps - pls.pred)^2))
```

```
## PLS test MSE = 1381335
```

(f) the above commands answer part (f)

```
plot(lasso.mod)
```

```
## Warning in regularize.values(x, y, ties, missing(ties)): collapsing to
## unique 'x' values
```




```
bestlam.l
```

```
## [1] 0.01
```

```
predict(cv.lasso, s = bestlam.l, type = "coefficients")
```

```
## 19 x 1 sparse Matrix of class "dgCMatrix"
##                          1
## (Intercept) -581.01303388
## (Intercept)    .
## PrivateYes  -467.18115486
## Accept         1.71102178
## Enroll        -1.18753033
## Top10perc     52.89600472
## Top25perc    -15.22169939
## F.Undergrad    0.06976949
## P.Undergrad    0.05772324
## Outstate      -0.08136889
## Room.Board     0.16118771
## Books          0.23357448
## Personal       0.00658961
## PhD          -11.12956791
## Terminal       0.90737997
## S.F.Ratio     16.89076321
## perc.alumni    2.23023594
```

```
## Expend          0.05569927
## Grad.Rate       6.42662614
```

```
test.average = mean(test[, "Apps"])
lasso.r2 = 1 - mean((test[, "Apps"] - lasso.pred)^2)/mean((test[, "Apps"] - test.average)^2)
print(paste0("lasso r^2 = ",lasso.r2))
```

```
## [1] "lasso r^2 = 0.908730036354661"
```

(g) PLS has the smallest CV MSE at 1376378. lasso, MLR, ridge, and PCR having slighly higher MSE's than PLS. Using lasso, the r^2 is .909 which means that we can explain 91% of the variation in applications recieved with the variation in the independent variables. there is not much difference between the the test error results of the five approaches.

Chapter 6 Question 11

```
library(MASS)
library(glmnet)
library(ISLR)
library(leaps)
library(pls)
summary(Boston)
```

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

```r
#?Boston
```

```r
regfit.full = regsubsets(crim~., Boston, nvmax = 13)
reg.summary = summary(regfit.full)
which.max(reg.summary$adjr2)
```

```
## [1] 9
```

```r
which.min(reg.summary$cp)
```

```
## [1] 8
```

```r
which.min(reg.summary$bic)
```

```
## [1] 3
```

```r
coef(regfit.full, 9)
```

```
##   (Intercept)            zn         indus           nox           dis
##  19.124636156   0.042788127  -0.099385948 -10.466490364  -1.002597606
##           rad       ptratio         black         lstat          medv
##   0.539503547  -0.270835584  -0.008003761   0.117805932  -0.180593877
```

```r
coef(regfit.full, 8)
```

```
##   (Intercept)            zn           nox           dis           rad
##  19.683127801   0.043293393 -12.753707757  -0.918318253   0.532616533
##       ptratio         black         lstat          medv
##  -0.310540942  -0.007922426   0.110173124  -0.174207166
```

```r
coef(regfit.full, 3)
```

```
##  (Intercept)          rad        black        lstat
## -0.372585457  0.488172386 -0.009471639  0.213595700
```

```r
set.seed(1)
sample = sample(nrow(Boston),nrow(Boston)*0.5)
train = Boston[sample,]
test = Boston[-sample,]
grid = 10^seq(10, -2, length = 100)
train.m = model.matrix(crim~., data = train)
test.m = model.matrix(crim~., data = test)
```

```r
ridge.mod = glmnet(train.m, train$crim, alpha = 0, lambda = grid)
cv.ridge = cv.glmnet(train.m, train$crim, alpha = 0, lambda = grid)
bestlam.r = cv.ridge$lambda.min
ridge.pred = predict(ridge.mod, s = bestlam.r, newx = test.m)
cat("ridge test RMSE =",sqrt(mean((test$crim - ridge.pred)^2)))
```

```
## ridge test RMSE = 6.411739
```

```r
library(plotmo)
```

```
## Loading required package: Formula
```

```
## Loading required package: plotrix
```

```
## Loading required package: TeachingDemos
```

```r
lasso.mod = glmnet(train.m, train$crim, alpha = 1, lambda = grid)
cv.lasso = cv.glmnet(train.m, train$crim, alpha = 1, lambda = grid)
bestlam.l = cv.lasso$lambda.min
bestlam.l
```

```
## [1] 0.07054802
```

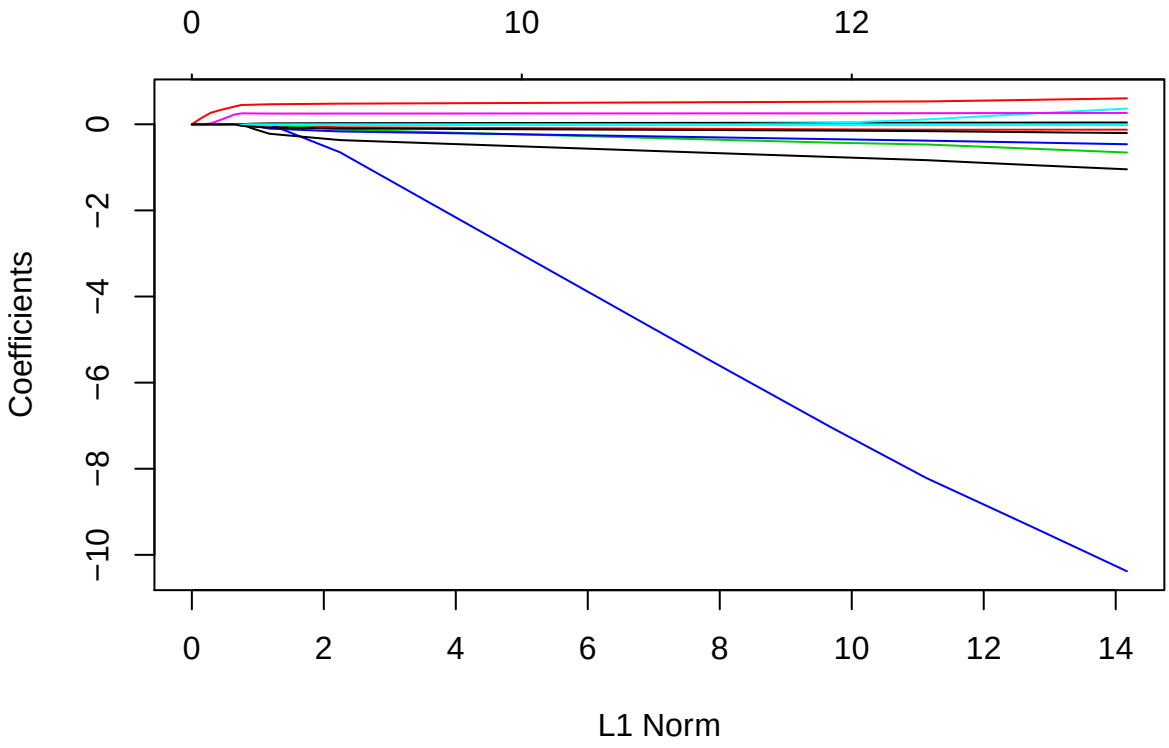
```r
lasso.pred = predict(lasso.mod, s = bestlam.l, newx = test.m)
cat("lasso test RMSE =",sqrt(mean((test$crim - lasso.pred)^2)))
```

```
## lasso test RMSE = 6.394322
```

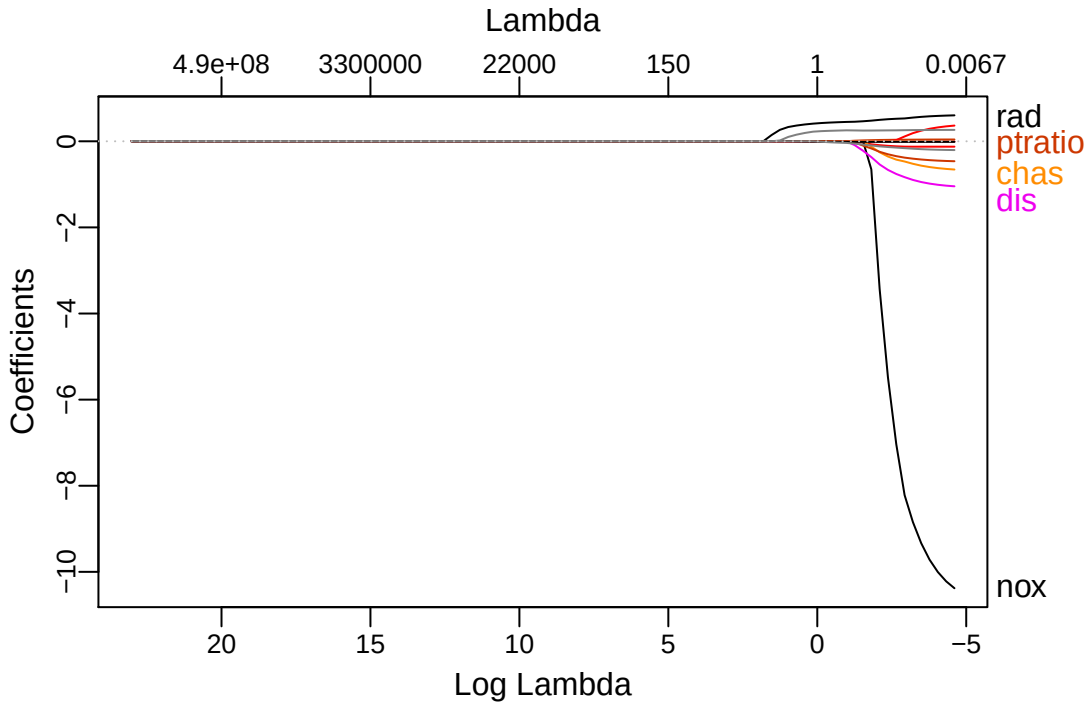
```r
x = model.matrix(crim~., Boston)[,-1]
out = glmnet(x, Boston$crim, alpha = 1, lambda = grid)
lasso.coef = predict(out, type = "coefficients", s = bestlam.l)
lasso.coef
```

```
## 14 x 1 sparse Matrix of class "dgCMatrix"
##                         1
## (Intercept) 11.377041716
## zn           0.034373111
## indus       -0.062605969
## chas        -0.555023861
## nox         -5.721122105
## rm           0.151927338
## age          .
## dis         -0.715085799
## rad          0.506621375
## tax          .
## ptratio     -0.156825685
## black       -0.007550409
## lstat        0.122010397
## medv        -0.145585637
```

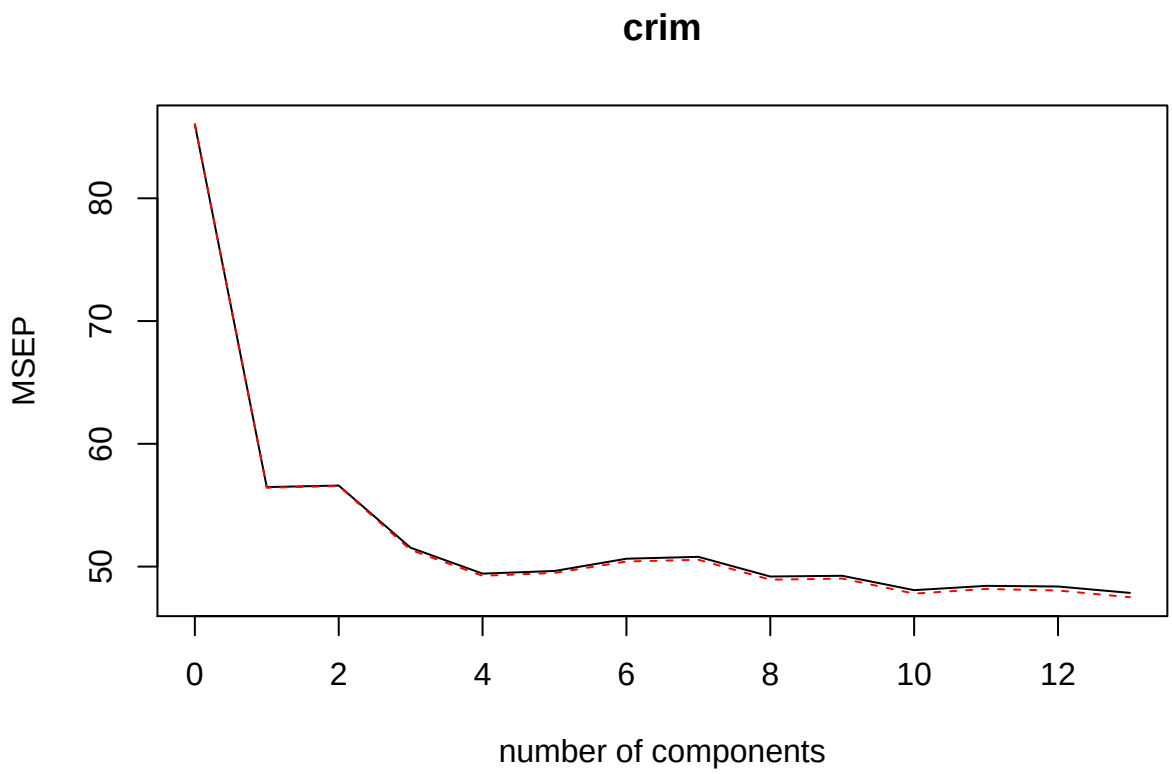
```r
plot(lasso.mod)
```




```
plot_glmnet(lasso.mod, label=5)
```




```
pcr.fit = pcr(crim~., data = Boston, subset = sample, scale = TRUE, validation = "CV")
validationplot(pcr.fit, val.type = "MSEP")
```




```
summary(pcr.fit)
```

```
## Data:    X dimension: 253 13
##  Y dimension: 253 1
## Fit method: svdpc
## Number of components considered: 13
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           9.275    7.515    7.523    7.179    7.030    7.046    7.116
## adjCV        9.275    7.511    7.521    7.166    7.018    7.034    7.101
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       7.127    7.014    7.018     6.934     6.959     6.955     6.918
## adjCV    7.110    6.995    7.002     6.914     6.941     6.932     6.892
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X       48.51     60.4    69.86    77.08    82.80    87.68    91.24
## crim    34.94     35.2    42.83    45.47    45.57    45.58    45.75
##       8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## X       93.56    95.47     97.08     98.48     99.54    100.00
## crim    47.59    47.68     48.75     49.31     50.14     51.37
```

(a) Using best subset selection, the adjusted R^2 estimated that a 9 predictor model was optimal, the AIC estimated that a 8 predictor model is optimal, and the BIC estimated that a 3 variable model is

optimal. Using CV ridge, the test MSE was 6.41. Using CV lasso, the test MSE was 6.39 and variables age and tax coefficients were pushed to 0. Using CV PCR, the lowest adjusted MSE was 6.89 with the 13 variable model but the 4 variable model was close behind with 6.96.

(b) I would choose the lasso model with lambda = .0705. I choose this model since it had the lowest test MSE compared to the other models. Even though the ridge model had a very similar test MSE, lasso is more interpretable than ridge is.

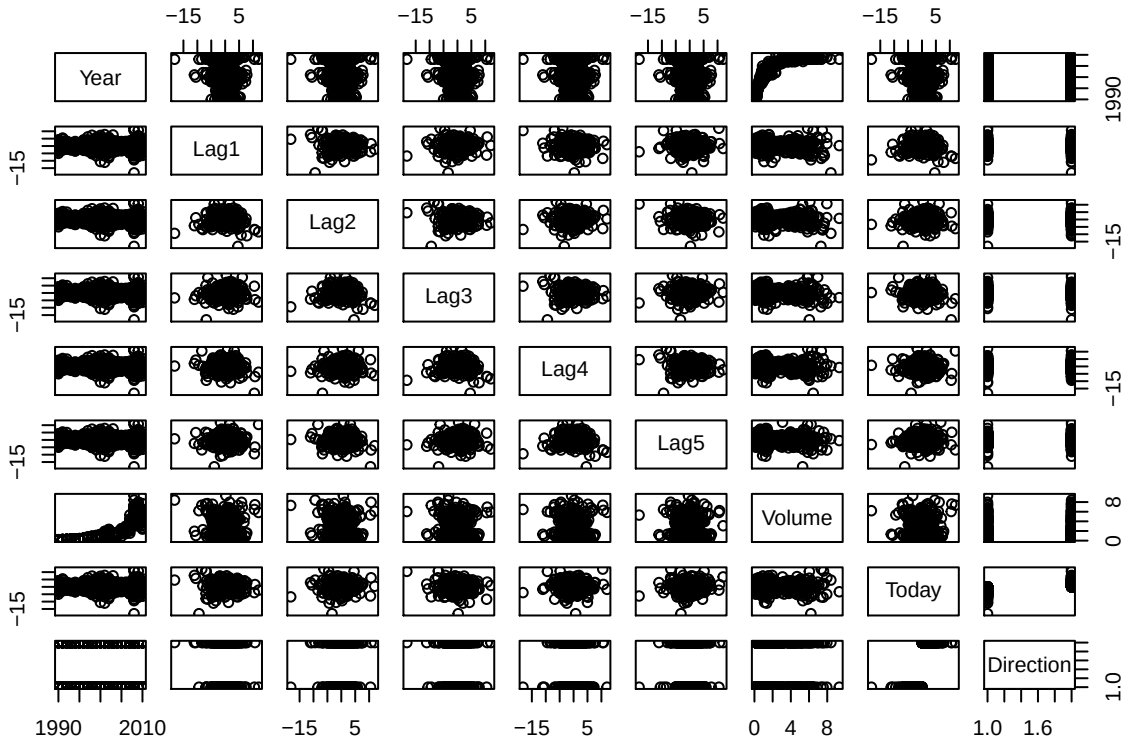
(c) My choosen model does not include all of the independent variables in the data set. one reason for this is becuase the best model based on test MSE excluded 2 of the variables. Another reason is that best subset selection also reduced the number of variables from 13 to somewhere between 3 and 9 depending on which method was used. Comparing lasso to best subset selection, the adjusted R^2 and AIC had the same predictors as lasso and lasso included additional variables. Overall, the main reason for reducing the number of independent variables was to increase iterpretability and eliminate irrelevant variables that could lead to unnecessary complexity in our model.

Chapter 4 Question 10

```
library(ISLR)
summary(Weekly)
```

```
##       Year           Lag1               Lag2               Lag3          
##  Min.   :1990   Min.   :-18.1950   Min.   :-18.1950   Min.   :-18.1950  
##  1st Qu.:1995   1st Qu.: -1.1540   1st Qu.: -1.1540   1st Qu.: -1.1580  
##  Median :2000   Median :  0.2410   Median :  0.2410   Median :  0.2410  
##  Mean   :2000   Mean   :  0.1506   Mean   :  0.1511   Mean   :  0.1472  
##  3rd Qu.:2005   3rd Qu.:  1.4050   3rd Qu.:  1.4090   3rd Qu.:  1.4090  
##  Max.   :2010   Max.   : 12.0260   Max.   : 12.0260   Max.   : 12.0260  
##       Lag4               Lag5              Volume       
##  Min.   :-18.1950   Min.   :-18.1950   Min.   :0.08747  
##  1st Qu.: -1.1580   1st Qu.: -1.1660   1st Qu.:0.33202  
##  Median :  0.2380   Median :  0.2340   Median :1.00268  
##  Mean   :  0.1458   Mean   :  0.1399   Mean   :1.57462  
##  3rd Qu.:  1.4090   3rd Qu.:  1.4050   3rd Qu.:2.05373  
##  Max.   : 12.0260   Max.   : 12.0260   Max.   :9.32821  
##      Today          Direction 
##  Min.   :-18.1950   Down:484  
##  1st Qu.: -1.1540   Up  :605  
##  Median :  0.2410             
##  Mean   :  0.1499             
##  3rd Qu.:  1.4050             
##  Max.   : 12.0260
```

```
attach(Weekly)
pairs(~., data = Weekly)
```



(a) most of the variables seem to be uncorrelated with eachother. The only trend that stands out is between Year and Volume with Volume increasing gradually over the years.

```
glm.fit = glm(Direction~Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume, data = Weekly, family = binomial)
summary(glm.fit)
```

```
##
## 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
```

```
table(Direction)
```

```
## Direction
## Down   Up
##  484  605
```

(b) Only the second lag is significant at the 5% level.

```
glm.probs = predict(glm.fit, type = "response")
glm.probs[1:10]
```

```
##         1         2         3         4         5         6         7
## 0.6086249 0.6010314 0.5875699 0.4816416 0.6169013 0.5684190 0.5786097
##         8         9        10
## 0.5151972 0.5715200 0.5554287
```

```
contrasts(Direction)
```

```
##      Up
## Down  0
## Up    1
```

```
glm.predi = rep("Down", 1089)
glm.predi[glm.probs > .5] = "Up"
table(glm.predi, Direction)
```

```
##          Direction
## glm.predi Down  Up
##      Down   54  48
##      Up    430 557
```

```
(557 + 54)/1089
```

```
## [1] 0.5610652
```

```
mean(glm.predi == Direction)
```

```
## [1] 0.5610652
```

(c) the diagonal elements of the confusion matrix indicate correct predictions, and the off-diagonals represent incorrect predictions. So the logistics regression correctly predicted the direction of the market 56.1% of the time.

```
set.seed(1)
train = (Year < 2009)
Weekly.2009.10 = Weekly[!train,]
dim(Weekly.2009.10)
```

```
## [1] 104   9
```

```
Direction.2009.10 = Direction[!train]
glm.fit1 = glm(Direction~ Lag2, data = Weekly, family = binomial, subset = train)
glm.probs = predict(glm.fit1, Weekly.2009.10, type = "response")
glm.pred = rep("Down", 104)
glm.pred[glm.probs > .5] = "Up"
table(glm.pred, Direction.2009.10)
```

```
##         Direction.2009.10
## glm.pred Down Up
##     Down    9  5
##     Up     34 56
```

```
print(paste0( "% of correct prediction = ",mean(glm.pred == Direction.2009.10)))
```

```
## [1] "% of correct prediction = 0.625"
```

(d) the above commands answer part (d)

```
library(class)
set.seed(1)
train.X = as.matrix(Lag2[train])
test.X = as.matrix(Lag2[!train])
train.Direction = Direction[train]
set.seed(1)
knn.pred = knn(train.X, test.X, train.Direction, k = 1)
table(knn.pred, Direction.2009.10)
```

```
##         Direction.2009.10
## knn.pred Down Up
##     Down   21 30
##     Up     22 31
```

```
mean(knn.pred == Direction.2009.10)
```

```
## [1] 0.5
```

(g) the above commands answer (g)

(h) Logistics regression appears to provide the best results compared to KNN since it correcly predicted the outcome 62.5% of the time compared to KNN which correctly predicted the outcome 50% of the time.

```
glm.fit2 = glm(Direction~ Lag2+Lag2^2, data = Weekly, family = binomial, subset = train)
glm.probs = predict(glm.fit2, Weekly.2009.10, type = "response")
glm.pred = rep("Down", 104)
glm.pred[glm.probs > .5] = "Up"
table(glm.pred, Direction.2009.10)
```

```
##         Direction.2009.10
## glm.pred Down Up
##     Down    9  5
##     Up     34 56
```

```
print(paste0( "% of correct prediction = ",mean(glm.pred == Direction.2009.10)))
```

```
## [1] "% of correct prediction = 0.625"
```

```
glm.fit2 = glm(Direction~ Lag1:Lag2, data = Weekly, family = binomial, subset = train)
glm.probs = predict(glm.fit2, Weekly.2009.10, type = "response")
glm.pred = rep("Down", 104)
glm.pred[glm.probs > .5] = "Up"
table(glm.pred, Direction.2009.10)
```

```
##         Direction.2009.10
## glm.pred Down Up
##     Down    1  1
##     Up     42 60
```

```
print(paste0( "% of correct prediction = ",mean(glm.pred == Direction.2009.10)))
```

```
## [1] "% of correct prediction = 0.586538461538462"
```

```
glm.fit2 = glm(Direction~. -Today, data = Weekly, family = binomial, subset = train)
glm.probs = predict(glm.fit2, Weekly.2009.10, type = "response")
glm.pred = rep("Down", 104)
glm.pred[glm.probs > .5] = "Up"
table(glm.pred, Direction.2009.10)
```

```
##         Direction.2009.10
## glm.pred Down Up
##     Down   30 44
##     Up     13 17
```

```
print(paste0( "% of correct prediction = ",mean(glm.pred == Direction.2009.10)))
```

```
## [1] "% of correct prediction = 0.451923076923077"
```

```
cat("k = 5\n")
```

```
## k = 5
```

```
knn.pred = knn(train.X, test.X, train.Direction, k = 5)
table(knn.pred, Direction.2009.10)
```

```
##         Direction.2009.10
## knn.pred Down Up
##     Down   15 20
##     Up     28 41
```

```
mean(knn.pred == Direction.2009.10)
```

```
## [1] 0.5384615
```

```
cat("k = 10\n")
```

```
## k = 10
```

```
knn.pred = knn(train.X, test.X, train.Direction, k = 10)
table(knn.pred, Direction.2009.10)
```

```
##         Direction.2009.10
## knn.pred Down Up
##     Down   17 19
##     Up     26 42
```

```
mean(knn.pred == Direction.2009.10)
```

```
## [1] 0.5673077
```

```
cat("k = 20\n")
```

```
## k = 20
```

```
knn.pred = knn(train.X, test.X, train.Direction, k = 20)
table(knn.pred, Direction.2009.10)
```

```
##         Direction.2009.10
## knn.pred Down Up
##     Down   21 20
##     Up     22 41
```

```
mean(knn.pred == Direction.2009.10)
```

```
## [1] 0.5961538
```

```
cat("k = 50\n")
```

```
## k = 50
```

```
knn.pred = knn(train.X, test.X, train.Direction, k = 50)
table(knn.pred, Direction.2009.10)
```

```
##         Direction.2009.10
## knn.pred Down Up
##     Down   20 21
##     Up     23 40
```

```
mean(knn.pred == Direction.2009.10)
```

```
## [1] 0.5769231
```

```
cat("k = 100\n")
```

```
## k = 100
```

```
knn.pred = knn(train.X, test.X, train.Direction, k = 100)
table(knn.pred, Direction.2009.10)
```

```
##         Direction.2009.10
## knn.pred Down Up
##     Down    9 11
##     Up     34 50
```

```
mean(knn.pred == Direction.2009.10)
```

```
## [1] 0.5673077
```

(i) experimenting with log and KNN models, the log model with just an interaction term between Lag1 and Lag2 correctly predicted the outcome 58.65% of the time. The log model just predicted UP 98% of the time. The best KNN model was k = 20 which correctly predicted the outcome 59.61% of the time. KNN predicted UP 59% of the time. So KNN with k = 20 would be the best model other than the log polynomial model. These two models had the highest correct predicted probabilities other than the log model with a sqaured term of Lag2. The polynomial log model predicted the correct outcome 62.5% of the time.

Chapter 8, Question 8

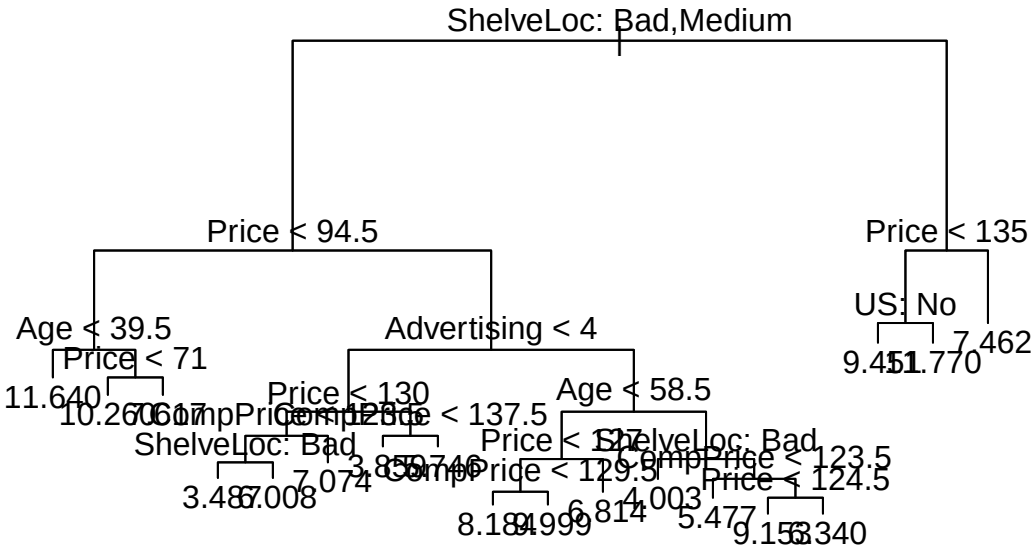
```
library(ISLR)
library(tree)
attach(Carseats)
#?Carseats
set.seed(1)
train = sample(1:nrow(Carseats), 200)
test = (-train)
Carseats.test = Carseats[-train,]
Sales.test = Sales[-train]
```

(a) the above commands answer part (a)

```
tree.carseats = tree(Sales~., Carseats, subset = train)
tree.pred = predict(tree.carseats, Carseats.test)
y = Carseats$Sales
y.test = y[test]
cat("reg tree test MSE =",mean((tree.pred - y.test)^2))
```

```
## reg tree test MSE = 4.922039
```

```
plot(tree.carseats)
text(tree.carseats, pretty = 0)
```



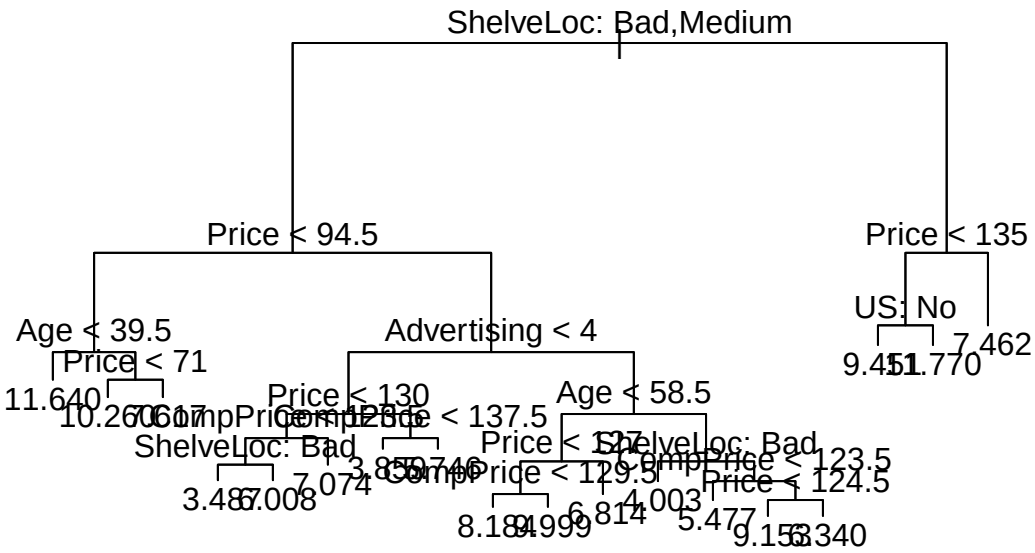
(b) the test MSE is 4.922. The tree indicates that shelve location is the most important variable with bad and medium shelve location reducing the sales of the carseats compared to a good shelve location. Price is the second most important variable.

```
set.seed(1)
cv.carseats = cv.tree(tree.carseats, FUN = prune.tree)
cv.carseats
```

```
## $size
##  [1] 18 17 16 15 14 13 12 11 10  8  7  6  5  4  3  2  1
##
## $dev
##  [1]  984.3936 1031.3372 1036.0021 1027.2166 1027.2166 1055.8168 1044.6955
##  [8] 1061.0899 1061.0899 1225.5973 1221.3487 1219.0219 1231.6886 1337.3952
```

```
## [15] 1300.0524 1338.3702 1605.0221
##
## $k
##  [1]      -Inf  16.99544  20.56322  25.01730  25.57104  28.01938  30.36962
##  [8]  31.56747  31.80816  40.75445  44.44673  52.57126  76.21881  99.59459
## [15] 116.69889 159.79501 337.60153
##
## $method
## [1] "deviance"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```
prune.carseats = prune.tree(tree.carseats, best = 18)
plot(prune.carseats)
text(prune.carseats, pretty = 0)
```




```
tree.cv.pred = predict(prune.carseats, Carseats.test)
mean((tree.cv.pred - y.test)^2)
```

```
## [1] 4.922039
```

(c) pruning the tree does not improve test MSE since the pruned MSE was 4.922 which is the same as a regression tree. This is becuase the best dev used all the predictors.

```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:dplyr':
##
##     combine
```

```r
set.seed(1)
dim(Carseats)
```

```
## [1] 400  11
```

```r
bag.Carseats = randomForest(Sales~., data = Carseats[train,], mtry = 10, ntree = 500, importance = TRUE)
bag.Carseats
```

```
##
## Call:
##  randomForest(formula = Sales ~ ., data = Carseats[train, ], mtry = 10,      ntree = 500, importance
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 10
##
##           Mean of squared residuals: 2.889221
##                     % Var explained: 63.26
```

```r
yhat.bag = predict(bag.Carseats, Carseats[-train,])
mean((Carseats.test$Sales - yhat.bag)^2)
```

```
## [1] 2.605253
```

```r
importance(bag.Carseats)
```

```
##                %IncMSE IncNodePurity
## CompPrice   24.8888481    170.182937
## Income       4.7121131     91.264880
## Advertising 12.7692401     97.164338
## Population  -1.8074075     58.244596
## Price       56.3326252    502.903407
## ShelveLoc   48.8886689    380.032715
## Age         17.7275460    157.846774
## Education    0.5962186     44.598731
## Urban        0.1728373      9.822082
## US           4.2172102     18.073863
```

(d) Bagging reduced the MSE to 2.605, the most important variables are price and shelve location.

```
rforest.Carseats = randomForest(Sales~., data = Carseats[train,], ntree = 500, importance = TRUE)
rforest.Carseats
```

```
##
## Call:
##  randomForest(formula = Sales ~ ., data = Carseats[train, ], ntree = 500,      importance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##           Mean of squared residuals: 3.42931
##                     % Var explained: 56.39
```

```
yhat.forest = predict(rforest.Carseats, Carseats[-train,])
mean((Carseats.test$Sales - yhat.forest)^2)
```

```
## [1] 3.054306
```

```
importance(rforest.Carseats)
```

```
##                %IncMSE IncNodePurity
## CompPrice   12.9540442     157.53376
## Income       2.1683293     129.18612
## Advertising  8.7289900     111.38250
## Population  -2.5290493     102.78681
## Price       33.9482500     393.61313
## ShelveLoc   34.1358807     289.28756
## Age         12.0804387     172.03776
## Education    0.2213600      72.02479
## Urban        0.9793293      14.73763
## US           4.1072742      33.91622
```

(e) the test MSE for random forest was 3.054, the most important variables were price, shelve location, and age. The error rate for RF was larger than bagging. the only difference between the two is that bagging allows all the variables to be considered at each split where RF only considers p/3 variables. In this case, allowing all variables to be considered reduced the test MSE cosiderably.

Chapter 8, Question 11

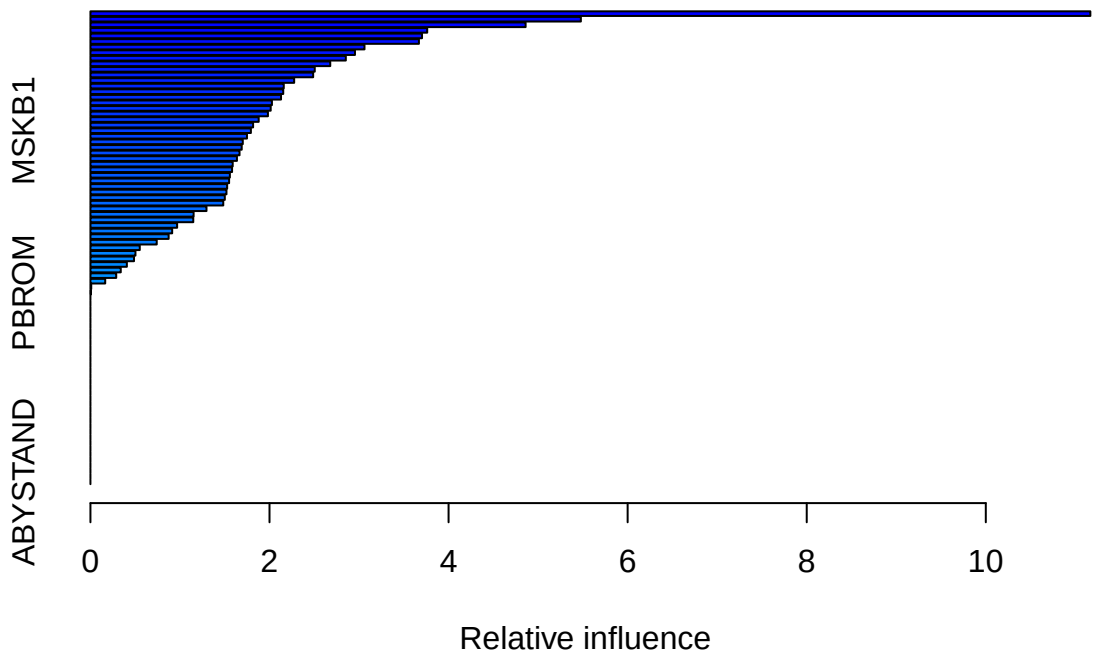
```
library(gbm)
```

```
## Loaded gbm 2.1.5
```

```
attach(Caravan)
#?Caravan
set.seed(1)
Caravan$Purchase = ifelse(Caravan$Purchase == "Yes", 1, 0)
train = sample(1:nrow(Caravan), 1000)
test = (-train)
Caravan.test = Caravan[-train,]
Caravan.train = Caravan[train,]
Purchase.test = Purchase[-train]
```

(a) the above commands answer (a)

```
boost.caravan = gbm(Purchase~., data = Caravan.train, distribution = "gaussian", n.trees = 1000, intera
summary(boost.caravan)
```




```
##                var      rel.inf
## PPERSAUT PPERSAUT 11.167900729
## PBRAND     PBRAND  5.477331783
## MOSTYPE   MOSTYPE  4.860383917
## MGODGE     MGODGE  3.761432939
## MBERMIDD MBERMIDD  3.703300923
## MAUT2       MAUT2  3.668784380
## MKOOPKLA MKOOPKLA  3.060735395
## MSKC         MSKC  2.954382294
## MSKB2       MSKB2  2.852363162
## MINK3045 MINK3045  2.678436222
## MHHUUR     MHHUUR  2.505501415
## MINKGEM   MINKGEM  2.487565610
## MINK4575 MINK4575  2.275483462
## MAUT1       MAUT1  2.159285297
## MHKOOP     MHKOOP  2.153463255
## MFALLEEN MFALLEEN  2.128650213
## MBERHOOG MBERHOOG  2.027033008
## MOPLLAAG MOPLLAAG  2.013462138
## ALEVEN     ALEVEN  1.982319117
```

```
## MBERARBO MBERARBO  1.879469568
## MFWEKIND MFWEKIND  1.816851211
## MSKB1       MSKB1  1.791964380
## MGODRK     MGODRK  1.748161271
## MINK7512 MINK7512  1.701593494
## MOPLHOOG MOPLHOOG  1.689636455
## MGODPR     MGODPR  1.665279444
## MBERARBG MBERARBG  1.636284388
## MOPLMIDD MOPLMIDD  1.589045261
## MGODOV     MGODOV  1.581865552
## MINKM30   MINKM30  1.557448412
## MSKD         MSKD  1.548692212
## MSKA         MSKA  1.527096284
## MFGEKIND MFGEKIND  1.519275458
## MAUT0       MAUT0  1.500708200
## MZFONDS   MZFONDS  1.483616113
## MRELGE     MRELGE  1.296140811
## PWAPART   PWAPART  1.152316697
## MRELOV     MRELOV  1.148592524
## MRELSA     MRELSA  0.968271481
## MGEMLEEF MGEMLEEF  0.912757753
## APERSAUT APERSAUT  0.873574767
## MZPART     MZPART  0.740540707
## MBERBOER MBERBOER  0.551081320
## MGEMOMV   MGEMOMV  0.502799695
## PLEVEN     PLEVEN  0.486554303
## MINK123M MINK123M  0.406729669
## MOSHOOFD MOSHOOFD  0.338762694
## MAANTHUI MAANTHUI  0.287925133
## MBERZELF MBERZELF  0.165071169
## PMOTSCO   PMOTSCO  0.008996733
## PBROM       PBROM  0.005081583
## PWABEDR   PWABEDR  0.000000000
## PWALAND   PWALAND  0.000000000
## PBESAUT   PBESAUT  0.000000000
## PVRAAUT   PVRAAUT  0.000000000
## PAANHANG PAANHANG  0.000000000
## PTRACTOR PTRACTOR  0.000000000
## PWERKT     PWERKT  0.000000000
## PPERSONG PPERSONG  0.000000000
## PGEZONG   PGEZONG  0.000000000
## PWAOREG   PWAOREG  0.000000000
## PZEILPL   PZEILPL  0.000000000
## PPLEZIER PPLEZIER  0.000000000
## PFIETS     PFIETS  0.000000000
## PINBOED   PINBOED  0.000000000
## PBYSTAND PBYSTAND  0.000000000
## AWAPART   AWAPART  0.000000000
## AWABEDR   AWABEDR  0.000000000
## AWALAND   AWALAND  0.000000000
## ABESAUT   ABESAUT  0.000000000
## AMOTSCO   AMOTSCO  0.000000000
## AVRAAUT   AVRAAUT  0.000000000
## AAANHANG AAANHANG  0.000000000
```

```
## ATRACTOR ATRACTOR  0.000000000
## AWERKT     AWERKT  0.000000000
## ABROM       ABROM  0.000000000
## APERSONG APERSONG  0.000000000
## AGEZONG   AGEZONG  0.000000000
## AWAOREG   AWAOREG  0.000000000
## ABRAND     ABRAND  0.000000000
## AZEILPL   AZEILPL  0.000000000
## APLEZIER APLEZIER  0.000000000
## AFIETS     AFIETS  0.000000000
## AINBOED   AINBOED  0.000000000
## ABYSTAND ABYSTAND  0.000000000
```

(b) The most important variables are PPERSAUT, PBRAND, and MOSTYPE.

```
yhat.boost = predict(boost.caravan, Caravan.test, n.trees = 1000, type = "response")
boost.pred = ifelse(yhat.boost > 0.2, 1, 0)
table(Caravan.test$Purchase, boost.pred)
```

```
##    boost.pred
##        0    1
##   0 4229  297
##   1  237   59
```

```
59/(59 + 297)
```

```
## [1] 0.1657303
```

```
log.caravan = glm(Purchase~., data = Caravan.train, family = "binomial")
log.prob = predict(log.caravan, Caravan.test, type = "response")
log.pred = ifelse(log.prob > 0.2, 1, 0)
table(Caravan.test$Purchase, log.pred)
```

```
##    log.pred
##        0    1
##   0 4207  319
##   1  244   52
```

```
52/(52+319)
```

```
## [1] 0.1401617
```

(c) 16.6% of the people predicted to make a purchase actually made a purchase using a boosted model. 14.0% of the people predicted to make a puchase actually made a puchase using a log model.

Problem 1, Beauty Pays!

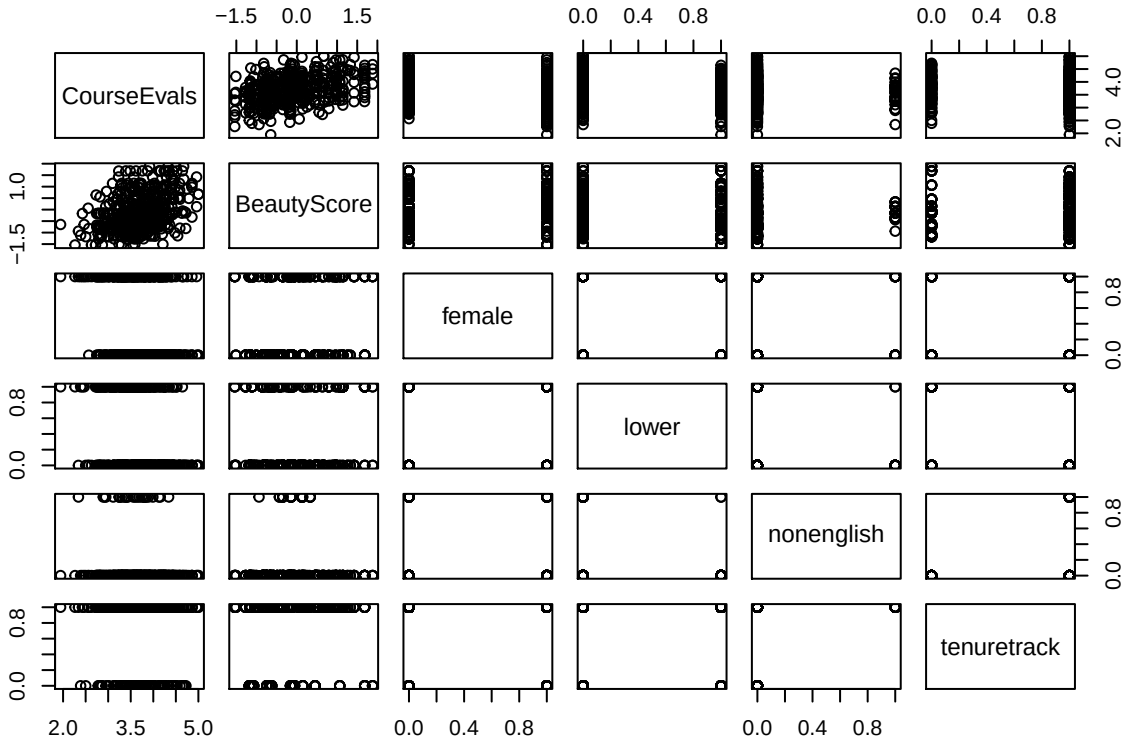
```
BeautyData <- read.csv("C:/Users/tsblo/Predictive modeling/Take_Home_Exam/BeautyData.csv")
summary(BeautyData)
```

```
##   CourseEvals     BeautyScore           female           lower
##  Min.   :1.944   Min.   :-1.53884   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:3.326   1st Qu.:-0.74462   1st Qu.:0.0000   1st Qu.:0.0000
##  Median :3.682   Median :-0.15636   Median :0.0000   Median :0.0000
##  Mean   :3.689   Mean   :-0.08835   Mean   :0.4212   Mean   :0.3391
##  3rd Qu.:4.067   3rd Qu.: 0.45725   3rd Qu.:1.0000   3rd Qu.:1.0000
##  Max.   :5.000   Max.   : 1.88167   Max.   :1.0000   Max.   :1.0000
##    nonenglish       tenuretrack
##  Min.   :0.00000   Min.   :0.0000
##  1st Qu.:0.00000   1st Qu.:1.0000
##  Median :0.00000   Median :1.0000
##  Mean   :0.06048   Mean   :0.7797
##  3rd Qu.:0.00000   3rd Qu.:1.0000
##  Max.   :1.00000   Max.   :1.0000
```

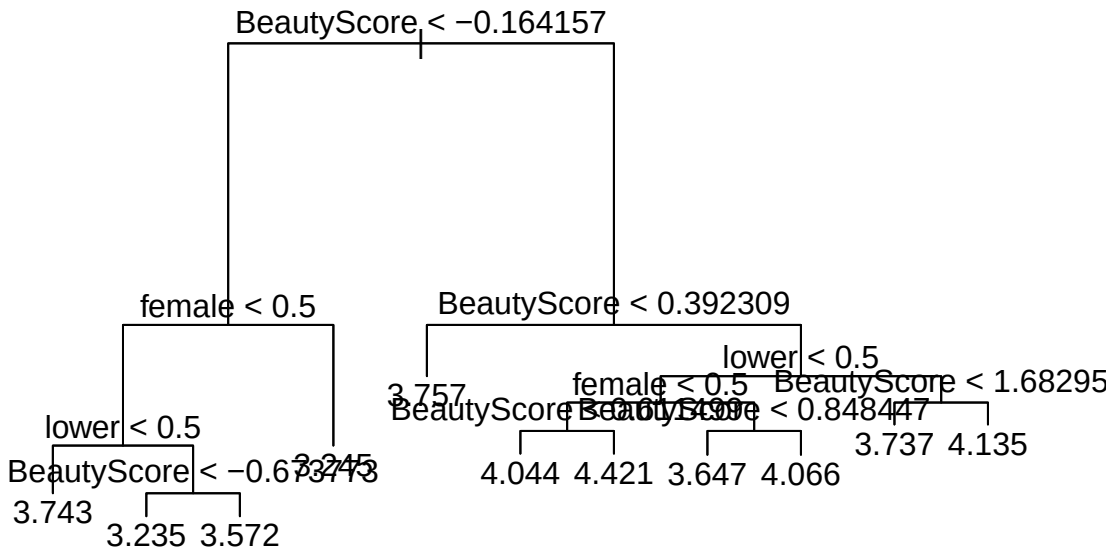
```
lm.fit = lm(CourseEvals~.*BeautyScore, data = BeautyData)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = CourseEvals ~ . * BeautyScore, data = BeautyData)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.28676 -0.29836  0.01589  0.28392  1.06096
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)                4.07729    0.05201  78.391  < 2e-16 ***
## BeautyScore                0.30943    0.07076   4.373 1.52e-05 ***
## female                    -0.32483    0.04145  -7.836 3.35e-14 ***
## lower                     -0.34792    0.04333  -8.030 8.51e-15 ***
## nonenglish                -0.22957    0.08620  -2.663  0.00802 **
## tenuretrack               -0.12195    0.05009  -2.435  0.01529 *
## BeautyScore:female         0.08216    0.05307   1.548  0.12228
## BeautyScore:lower         -0.04893    0.05427  -0.902  0.36775
## BeautyScore:nonenglish     0.36504    0.27747   1.316  0.18898
## BeautyScore:tenuretrack   -0.03497    0.06215  -0.563  0.57393
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4264 on 453 degrees of freedom
## Multiple R-squared:  0.3553, Adjusted R-squared:  0.3425
## F-statistic: 27.74 on 9 and 453 DF,  p-value: < 2.2e-16
```

```
pairs(BeautyData)
```




```
tree.beauty = tree(CourseEvals~., data = BeautyData)
plot(tree.beauty)
text(tree.beauty, pretty = 0)
```



1. from both linear regression and regression tree, the three most important variables are BeautyScore, gender, and lower division class. Increasing the attractiveness of the professor by 1 point, is predicted to increase the evaluation score by .31 holding the other variables constant. Being a female professor and teacher a lower division class is predicted to reduce the evaluation score. not being a native english speaking professor also reduced the predicted evaluation score. Non of the other predictors are significant at or below the 1% level. The regression tree only uses BeautyScore, female, and lower variables to predict evalutation score. None of the interaction terms with Beauty were significant. First, I was surpised by female professors getting lower scores than males. I was also suprised that none of the interaction terms were significant especially between female and Beauty.

2. What the professor means is that we cannot say with certainty that Beauty has a causal effect on evaluation score. This is becuase the BeautyScore could be associated with other variables that are not in the dataset or the regression. This is called ommitted variable bias. becuase there are variables that effect Beauty, we cannot accuratly predict evaluations score. More attractive people may be more productive and hence are better teachers. Or productivity could have no correlation with attractiveness. But without including some productivity variable in the regression, we cannot say that students are discriminating against professor becuase of their looks or their teaching methods.

Problem 2, Housing Price Structure

```
MidCity <- read.csv("C:/Users/tsblo/Predictive modeling/Take_Home_Exam/MidCity.csv")
summary(MidCity)
```

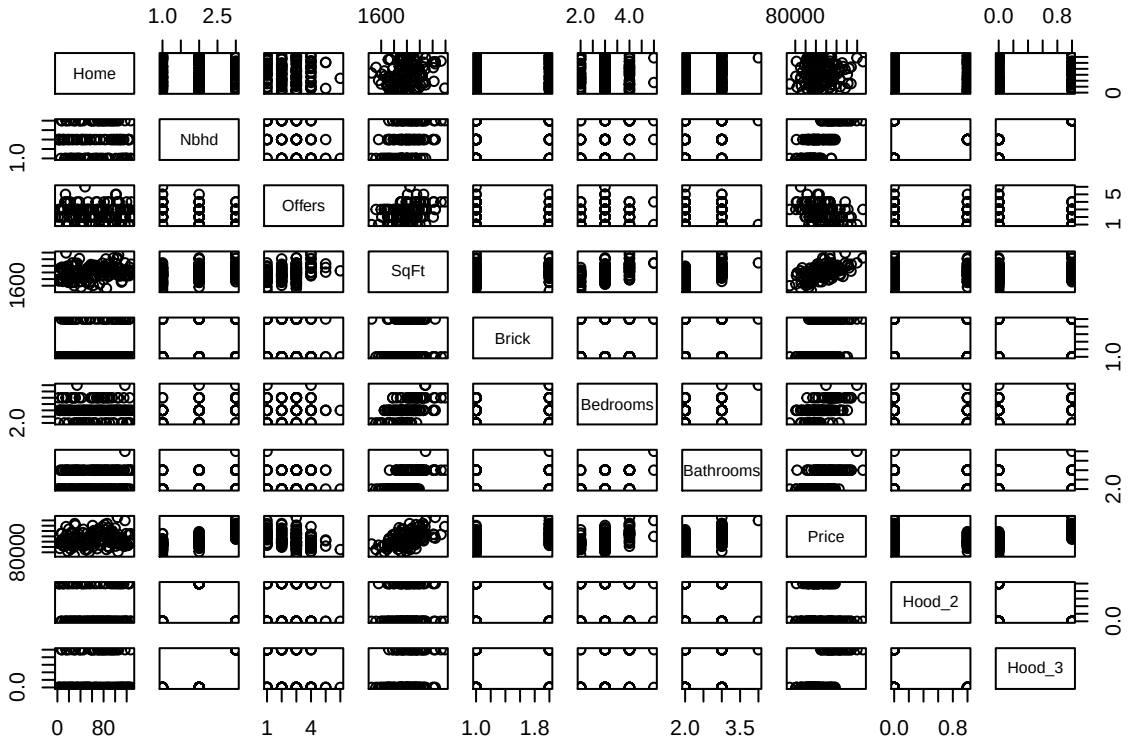
```
##       Home            Nbhd           Offers           SqFt      Brick
##  Min.   :  1.00   Min.   :1.000   Min.   :1.000   Min.   :1450   No :86
##  1st Qu.: 32.75   1st Qu.:1.000   1st Qu.:2.000   1st Qu.:1880   Yes:42
```

```
##  Median : 64.50   Median :2.000   Median :3.000   Median :2000
##  Mean   : 64.50   Mean   :1.961   Mean   :2.578   Mean   :2001
##  3rd Qu.: 96.25   3rd Qu.:3.000   3rd Qu.:3.000   3rd Qu.:2140
##  Max.   :128.00   Max.   :3.000   Max.   :6.000   Max.   :2590
##     Bedrooms       Bathrooms         Price
##  Min.   :2.000   Min.   :2.000   Min.   : 69100
##  1st Qu.:3.000   1st Qu.:2.000   1st Qu.:111325
##  Median :3.000   Median :2.000   Median :125950
##  Mean   :3.023   Mean   :2.445   Mean   :130427
##  3rd Qu.:3.000   3rd Qu.:3.000   3rd Qu.:148250
##  Max.   :5.000   Max.   :4.000   Max.   :211200
```

```
MidCity$Hood_2 = ifelse(MidCity$Nbhd == 2, 1, 0)
MidCity$Hood_3 = ifelse(MidCity$Nbhd == 3, 1, 0)
lm.fit = lm(Price~. +Hood_3:Brick-Nbhd, data = MidCity)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = Price ~ . + Hood_3:Brick - Nbhd, data = MidCity)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -27515.9  -5681.0   -459.6   4451.0  26695.6
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)      2885.512   8738.695   0.330  0.74183
## Home              -11.799     24.875  -0.474  0.63613
## Offers          -8486.348   1082.875  -7.837 2.26e-12 ***
## SqFt               54.726      5.823   9.397 5.36e-16 ***
## BrickYes        13839.320   2413.580   5.734 7.69e-08 ***
## Bedrooms         4605.046   1600.639   2.877  0.00477 **
## Bathrooms        6556.432   2170.200   3.021  0.00309 **
## Hood_2           -846.146   2412.025  -0.351  0.72636
## Hood_3          17086.915   3417.999   4.999 2.02e-06 ***
## BrickYes:Hood_3 10192.783   4178.971   2.439  0.01621 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9850 on 118 degrees of freedom
## Multiple R-squared:  0.8751, Adjusted R-squared:  0.8656
## F-statistic:  91.9 on 9 and 118 DF,  p-value: < 2.2e-16
```

```
pairs(MidCity)
```



1. There is a premium for brick houses holding all else equal. if a house is made of brick, the price is predicted to increase by $13,839 over a house not made of brick holding all the other variables equal.

2. yes, there is a premium on houses in neihborhood three. Looking at the dummy variable, Hood_3, a house in neighborhood 3 is predicted to have a price $17,086 higher than a house in neighborhood 1 holding all the other variables constant.

3. yes, there is a premium on brick houses in neighborhood 3. looking at th interaction term, a brick house in neighborhood 3 is predicted to have a price $10,192 higher than a non brick house in neighborhood 3 holding all else constant. One thing to note is that the interaction term is only significant at the 5% level so its signifigance is less compared to the past two questions.

```
MidCity$Hood_1and2 = ifelse(MidCity$Nbhd == 1 & MidCity$Nbhd == 2, 1, 0)
dim(MidCity)
```

```
## [1] 128  11
```

```
attach(MidCity)
set.seed(1)
train = sample(128, 60)
test = (-train)

lm.fit = lm(Price~.+Hood_3:Brick-Nbhd-Hood_2, data = MidCity)
summary(lm.fit)
```

```
##
```

```
## Call:
## lm(formula = Price ~ . + Hood_3:Brick - Nbhd - Hood_2, data = MidCity)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -27170.2  -5937.2   -458.1   4282.0  27145.2
##
## Coefficients: (1 not defined because of singularities)
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       3387.738   8588.803   0.394  0.69396
## Home               -10.479     24.498  -0.428  0.66961
## Offers           -8351.141   1008.227  -8.283 2.05e-13 ***
## SqFt                54.238      5.634   9.626  < 2e-16 ***
## BrickYes         13638.961   2336.373   5.838 4.68e-08 ***
## Bedrooms          4536.481   1582.797   2.866  0.00492 **
## Bathrooms         6477.586   2150.562   3.012  0.00317 **
## Hood_3           17679.562   2960.335   5.972 2.49e-08 ***
## Hood_1and2              NA         NA      NA       NA
## BrickYes:Hood_3  10412.933   4116.331   2.530  0.01272 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9813 on 119 degrees of freedom
## Multiple R-squared:  0.875,  Adjusted R-squared:  0.8666
## F-statistic: 104.1 on 8 and 119 DF,  p-value: < 2.2e-16
```

```
lm.fit = lm(Price~.+Hood_3:Brick-Nbhd-Hood_2, data = MidCity, subset = train)
print(paste0( "test MSE for OLS combining Nhbh 1 & 2 = ",mean((Price - predict(lm.fit, MidCity))[-train]
```

```
## [1] "test MSE for OLS combining Nhbh 1 & 2 = 99111000.2196552"
```

```
lm.fit = lm(Price~.+Hood_3:Brick-Nbhd-Hood_1and2, data = MidCity, subset = train)
print(paste0( "test MSE for OLS with seperate Nhbhs = ",mean((Price - predict(lm.fit, MidCity))[-train]
```

```
## [1] "test MSE for OLS with seperate Nhbhs = 99277868.5295582"
```

4. yes, you could combine neighborhoods 1 and 2 into a single binary variable, and as my validation set approach shows, the test MSE for combining the neighborhoods has a slightly lower test MSE than with the seperated neighborhoods but the test MSE's are quite similar. This is becuase both neighborhood 1 and 2 have very similar characteristics. So combining the variables does not decrease the predictive accuracy of OLS.

Problem 3, what causes what? 1. you cant just regress crime rate on number of cops becuase that will only show you the correlation between those two variables and will not answer the question: does number of cops have a causal effect on crime rate? This is becuase there is omitted variable bias, or variables that are not included in the regression that effect number of cops and crime rate. since these variables effect number of cops, they effect crime rate and the regression cannot dicipher which variables truly effect crime rate.

2. The UPENN researchers used an ingenius idea. they used an instrumental variable, high alert. they replaced number of cops in the regression with high alert. High alert is associated with more cops on the streets, so there is a high correlation between the two. But the researchers are assuming that high alert is not associated with, not correlated with, any other omitted variable that are also correlated

with crime levels. this gets around the problem of omitted variable bias, and if their assumptions are correct, the variable high alert can determine if there is a causal effect of number of cops on crime. Table 2 shows that in the simple regression of crime rate on high alert, being on high alert is predicted to decrease crime rate by 7.316 percent and is significant at the 5% level. Table 2 also shows the multi linear regression of crime rate on high alert and midday ridership. being on high alert is predicted to decrease crime by 6.046 percent holding midday ridership fixed and is significant at the 5% level.

3. Metro ridership could have gone down during a high risk day which could reduce crime since there are less people on the streets. if this were the case, the high risk variable would be correlated with metro ridership and would therefore not be an instrumental variable.

4. this is a multi linear regression of crime rate on high alert + district 1 interaction variable, high alert + other districts interaction variable, and midday ridership variable. the coefficent of high alert + district 1 can be interpreted as if washington has a high alert day, crime rate of district 1 is predicted to decrease by 2.621 percent at the 1% signifigance level compared to other districts. the coefficent for high alert + other districts is not significant. midday ridership variable is significant and positive which means ridership actually goes up during high risk days. If high alert is truly an instrumental variable for number of cops, this means that increasing the number of cops on the streets causes crime to decrease in district 1 but not in other districts. So without interaction terms like in table 2, it could be interpreted as increasing the number of cops decreases all crime in washington but in actuality, as table 4 shows, increasing the number of cops only decreases crime in district 1 and not any of the other districts.

Problem 4, BART (WANING! MOST OF THIS CODE IS CARLOS'S, I JUST USED IT TO GET THE RMSE OF RANDOM FOREST AND BOOSTING)

```
library(randomForest)
ca <- read.csv("CAhousing.csv",header=TRUE)
ca$AveBedrms <- ca$totalBedrooms/ca$households
ca$AveRooms <- ca$totalRooms/ca$households
ca$AveOccupancy <- ca$population/ca$households
logMedVal <- log(ca$medianHouseValue)
ca <- ca[,-c(4,5,9)] # lose lmedval and the room totals
ca$logMedVal = logMedVal

#-------------------------------------------------
#train, val, test
set.seed(99)
n=nrow(ca)
n1=floor(n/2)
n2=floor(n/4)
n3=n-n1-n2
ii = sample(1:n,n)
catrain=ca[ii[1:n1],]
caval = ca[ii[n1+1:n2],]
catest = ca[ii[n1+n2+1:n3],]
```

```
set.seed(1)
catrainval = rbind(catrain,caval)
finrf = randomForest(logMedVal~.,data=catrainval,mtry=3,ntree=500)
finrfpred=predict(finrf,newdata=catest)

finrfrmse = sqrt(sum((catest$logMedVal-finrfpred)^2)/nrow(catest))
cat('finrfrmse: ',finrfrmse,'\n')
```

```
## finrfrmse:  0.2379224
```

random forest regression, RMSE = .237

```
library(gbm)

set.seed(1)
catrainval = rbind(catrain,caval)
ntrees=5000
finb = gbm(logMedVal~.,data=catrainval,distribution='gaussian',
              interaction.depth=4,n.trees=ntrees,shrinkage=.2)
finbpred=predict(finb,newdata=catest,n.trees=ntrees)


finbrmse = sqrt(sum((catest$logMedVal-finbpred)^2)/nrow(catest))
cat('finbrmse: ',finbrmse,'\n')
```

```
## finbrmse:  0.2378038
```

boosting, RMSE = .237

```
library(MASS)
x = ca[,1:9]
y = ca$logMedVal
head(cbind(x,y))
```

```
##   longitude latitude housingMedianAge population households medianIncome
## 1   -122.23    37.88                41        322        126       8.3252
## 2   -122.22    37.86                21       2401       1138       8.3014
## 3   -122.24    37.85                52        496        177       7.2574
## 4   -122.25    37.85                52        558        219       5.6431
## 5   -122.25    37.85                52        565        259       3.8462
## 6   -122.25    37.85                52        413        193       4.0368
##   AveBedrms AveRooms AveOccupancy        y
## 1 1.0238095 6.984127     2.555556 13.02276
## 2 0.9718805 6.238137     2.109842 12.78968
## 3 1.0734463 8.288136     2.802260 12.77167
## 4 1.0730594 5.817352     2.547945 12.74052
## 5 1.0810811 6.281853     2.181467 12.74315
## 6 1.1036269 4.761658     2.139896 12.50507
```

```
library(BART) #BART package
```

```
## Warning: package 'BART' was built under R version 3.6.1
```

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

```
##
## Attaching package: 'nlme'
```

```
## The following object is masked from 'package:dplyr':
##
##     collapse

## Loading required package: nnet

## Loading required package: survival
```

```
set.seed(1) #MCMC, so set the seed
nd=200 # number of kept draws
burn=50 # number of burn in draws
bf = wbart(x,y,nskip=burn,ndpost=nd)
```

```
## *****Into main of wbart
## *****Data:
## data:n,p,np: 20640, 9, 0
## y1,yn: 0.937880, -0.684008
## x1,x[n*p]: -122.230000, 2.616981
## *****Number of Trees: 200
## *****Number of Cut Points: 100 ... 100
## *****burn and ndpost: 50, 200
## *****Prior:beta,alpha,tau,nu,lambda: 2.000000,0.950000,0.061989,3.000000,0.022423
## *****sigma: 0.339283
## *****w (weights): 1.000000 ... 1.000000
## *****Dirichlet:sparse,theta,omega,a,b,rho,augment: 0,0,1,0.5,1,9,0
## *****nkeeptrain,nkeeptest,nkeeptestme,nkeeptreedraws: 200,200,200,200
## *****printevery: 100
## *****skiptr,skipte,skipteme,skiptreedraws: 1,1,1,1
##
## MCMC
## done 0 (out of 250)
## done 100 (out of 250)
## done 200 (out of 250)
## time: 50s
## check counts
## trcnt,tecnt,temecnt,treedrawscnt: 200,0,0,200
```

```
lmf = lm(y~.,data.frame(x,y))
fitmat = cbind(y,bf$yhat.train.mean,lmf$fitted.values)
colnames(fitmat)=c("y","BART","Linear")
cor(fitmat)
```

```
##                y      BART    Linear
## y      1.0000000 0.9080333 0.8029780
## BART   0.9080333 1.0000000 0.8870965
## Linear 0.8029780 0.8870965 1.0000000
```

```
n=length(y) #total sample size
set.seed(1) #
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(1)
bf_train = wbart(xtrain,ytrain)
```

```
## *****Into main of wbart
## *****Data:
## data:n,p,np: 15480, 9, 0
## y1,yn: -0.706682, -0.494339
## x1,x[n*p]: -120.440000, 2.839898
## *****Number of Trees: 200
## *****Number of Cut Points: 100 ... 100
## *****burn and ndpost: 100, 1000
## *****Prior:beta,alpha,tau,nu,lambda: 2.000000,0.950000,0.061989,3.000000,0.022288
## *****sigma: 0.338257
## *****w (weights): 1.000000 ... 1.000000
## *****Dirichlet:sparse,theta,omega,a,b,rho,augment: 0,0,1,0.5,1,9,0
## *****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: 235s
## 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: 9
## from x,np,p: 9, 5160
## ***using serial code
```

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

finbrmse = sqrt(sum((ytest-yhat.mean)^2)/length(ytest))
cat('BART finbrmse: ',finbrmse,'\n')
```

```
## BART finbrmse:  0.2444159
```

Bart does not do as well as random forest or boosting looking at the validation RMSE, both RF and boosting have RMSE's of .237 where as Bart has RMSE of .244.

Problem 5, Neural Nets

```
attach(Boston)
library(nnet)
dim(Boston)
```

```
## [1] 506  14
```

```
set.seed(1)
maxs = as.numeric(apply(Boston, 2, max))
mins = as.numeric(apply(Boston, 2, min))
Boston = scale(Boston, center = mins, scale = maxs - mins)
n = nrow(Boston)
ii = sample(1:n, n)
n1=floor(n/2)
n2=floor(n/4)
n3=n-n1-n2
ii = sample(1:n,n)
Btrain= Boston[ii[1:n1],]
Bval = Boston[ii[n1+1:n2],]
Btest = Boston[ii[n1+n2+1:n3],]
```

```
set.seed(1)
Btrainval = rbind(Btrain, Bval)
decayList = c(.00001, .0001, .001, .01, .1, .3, .5, .8)
for (decay in decayList){
  sizeList = c(1, 3, 5, 10, 25, 50)
  for (size in sizeList){
    Bnn = nnet(crim~.,data = Btrainval,size=size,decay=decay,linout=T, trace = FALSE)
    pBnn = predict(Bnn, newdata = Btest)
    cv.rmse = sqrt(sum((Btest[, "crim"] -pBnn)^2)/nrow(Btest))
    cat("rmse:", cv.rmse, "decay =", decay, "size =", size, "\n")
}}
```

```
## rmse: 0.09331687 decay = 1e-05 size = 1
## rmse: 0.09616362 decay = 1e-05 size = 3
## rmse: 0.09725205 decay = 1e-05 size = 5
## rmse: 0.09444471 decay = 1e-05 size = 10
## rmse: 0.09197467 decay = 1e-05 size = 25
## rmse: 0.08793491 decay = 1e-05 size = 50
## rmse: 0.09581228 decay = 1e-04 size = 1
## rmse: 0.09456259 decay = 1e-04 size = 3
## rmse: 0.09413516 decay = 1e-04 size = 5
## rmse: 0.09420723 decay = 1e-04 size = 10
## rmse: 0.09193111 decay = 1e-04 size = 25
## rmse: 0.09074399 decay = 1e-04 size = 50
## rmse: 0.09251309 decay = 0.001 size = 1
## rmse: 0.09191416 decay = 0.001 size = 3
## rmse: 0.09195692 decay = 0.001 size = 5
## rmse: 0.09202006 decay = 0.001 size = 10
```

```
## rmse: 0.09015606 decay = 0.001 size = 25
## rmse: 0.09060464 decay = 0.001 size = 50
## rmse: 0.09004009 decay = 0.01 size = 1
## rmse: 0.08941482 decay = 0.01 size = 3
## rmse: 0.08918997 decay = 0.01 size = 5
## rmse: 0.08864141 decay = 0.01 size = 10
## rmse: 0.08869214 decay = 0.01 size = 25
## rmse: 0.08928459 decay = 0.01 size = 50
## rmse: 0.09363157 decay = 0.1 size = 1
## rmse: 0.09367814 decay = 0.1 size = 3
## rmse: 0.09362426 decay = 0.1 size = 5
## rmse: 0.09364999 decay = 0.1 size = 10
## rmse: 0.09353155 decay = 0.1 size = 25
## rmse: 0.09351047 decay = 0.1 size = 50
## rmse: 0.09503941 decay = 0.3 size = 1
## rmse: 0.09486434 decay = 0.3 size = 3
## rmse: 0.09484929 decay = 0.3 size = 5
## rmse: 0.09482125 decay = 0.3 size = 10
## rmse: 0.09479807 decay = 0.3 size = 25
## rmse: 0.09478361 decay = 0.3 size = 50
## rmse: 0.0962298 decay = 0.5 size = 1
## rmse: 0.09590017 decay = 0.5 size = 3
## rmse: 0.09585743 decay = 0.5 size = 5
## rmse: 0.09585097 decay = 0.5 size = 10
## rmse: 0.09579761 decay = 0.5 size = 25
## rmse: 0.09569887 decay = 0.5 size = 50
## rmse: 0.09842166 decay = 0.8 size = 1
## rmse: 0.09752126 decay = 0.8 size = 3
## rmse: 0.09745165 decay = 0.8 size = 5
## rmse: 0.09737774 decay = 0.8 size = 10
## rmse: 0.09734359 decay = 0.8 size = 25
## rmse: 0.09717638 decay = 0.8 size = 50
```

after cross validating for a few decay and size parameters, found that the best decay, size parameter pair was .00001, 50 with an RMSE of .087. decay = .01, size = 10 was the second best with an RMSE of .088. Just like in the slides, it is probably more important to CV for the decay parameter and just have a large size as default.

Problem 6: final project

My team was having a hard time finding a data set that we could easily create models for ie most of the data sets we liked had a lot of prepossesing work which we were not fimilair with. I found a data set that was clean and we chose to select it for our project. I created dummy variables for some of the categorical variables. I ran linear regression models with and without interaction terms and used a validation set approach to calculate test MSE. I ran cross validation lasso and ridge models and made interpretable graphs for each for our presentation. I ran basic regression tree, bagging, boosting, and randomforest models all using validation approach to calculate MSE. I ran cross validation boosting. I also helped structure the power point presentation and created some of the slides.