Predictive Modeling Exam

Gihani Dissanayake

July 26, 2017

Chapter 2, #10a

```
library(MASS)
library(ggplot2)
?Boston
```

```
## starting httpd help server ...
```

```
##  done
```

```
dim(Boston)
```

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

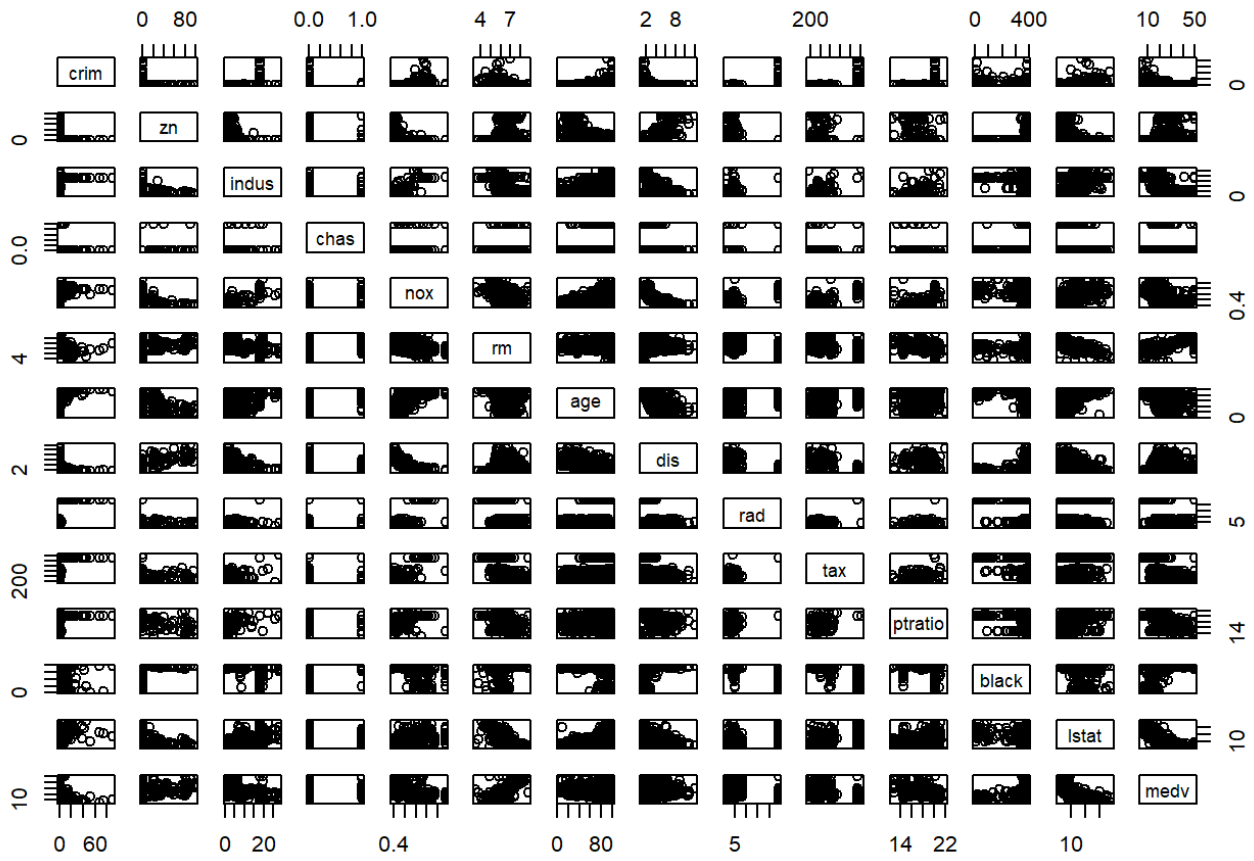
```
names(Boston)
```

```
##  [1] "crim"    "zn"      "indus"   "chas"    "nox"     "rm"      "age"
##  [8] "dis"     "rad"     "tax"     "ptratio" "black"   "lstat"   "medv"
```

There are 506 rows and 14 columns of data. These represent 14 features of each of the 506 neighborhoods. The variables are crim, zn, indus, chas, nox, rm, age, dis, rad, tax, ptratio, black, lstat, and medv.

Chapter 2, #10b

```
pairs(Boston)
```



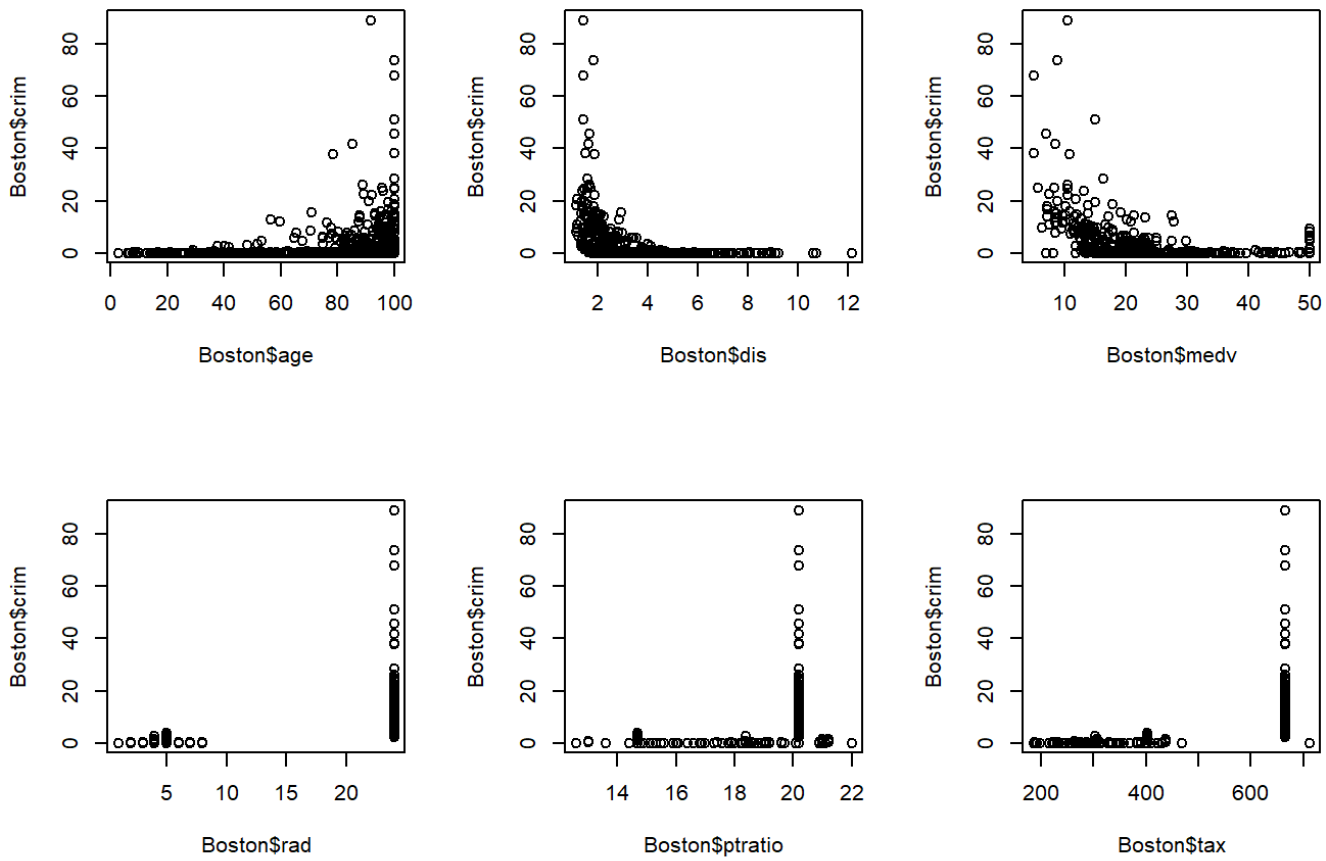

```
#pairs(Boston[,1:6])
#pairs(Boston[,7:14])
```

The scatter plots suggest several relationships between variables. - Crim seems to correlate to zn, indus, age, dis, rad, tax, ptratio, and medv. - Zn seems to correlate to indus, nox, age, rad, black, and lstat. - Indus seems to correlate to age, dis, and rad. - Nox seems to correlate to age and dis. - Rm seems to correlate to lstat and medv. - Dis seems to correlate to black and lstat. - Lstat seems to correlate to medv.

Though these are suggested, we will model out the specifics later.

Chapter 2, #10c

```
par(mfrow=c(2,3))
plot(Boston$age, Boston$crim)
plot(Boston$dis, Boston$crim)
plot(Boston$medv, Boston$crim)
plot(Boston$rad, Boston$crim)
plot(Boston$ptratio, Boston$crim)
plot(Boston$tax, Boston$crim)
```



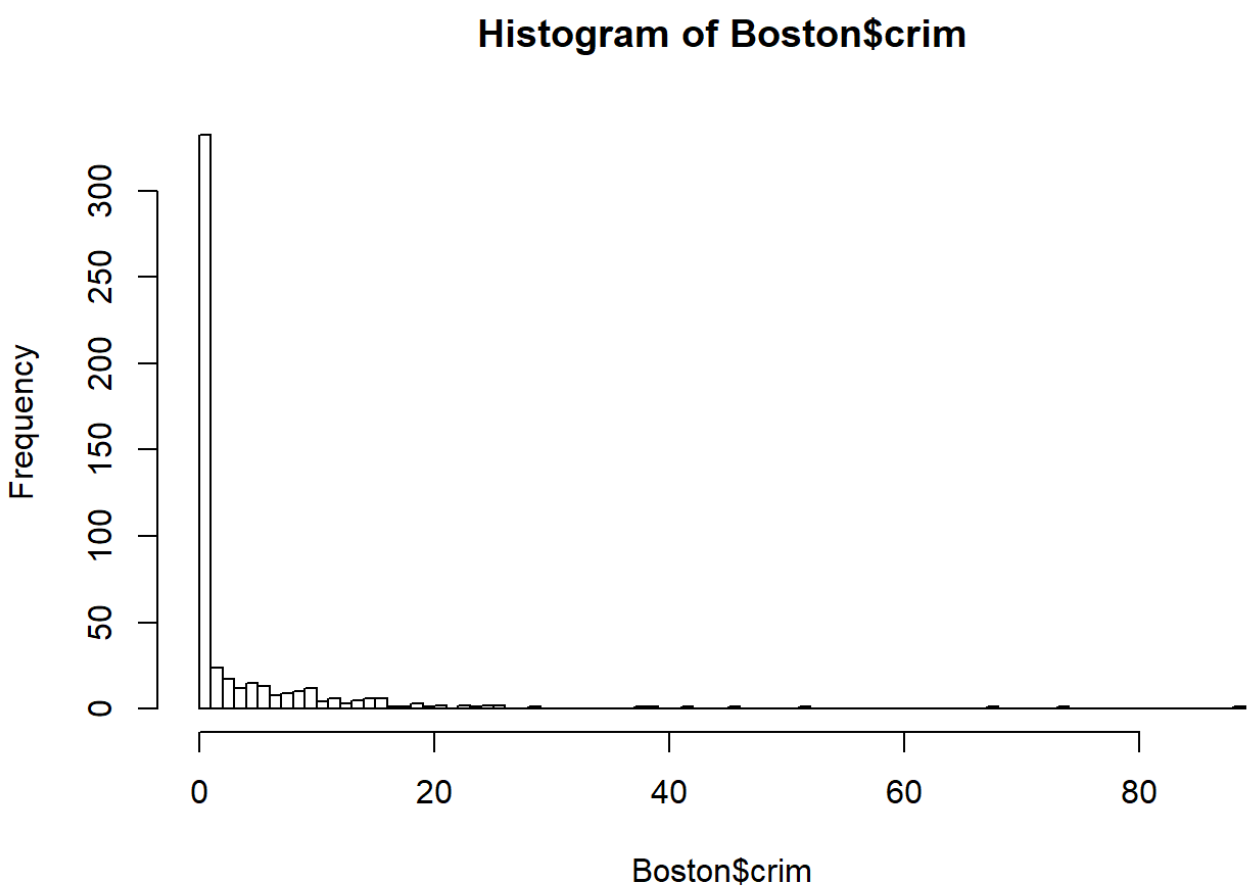
Predictors associated with per capita crime rate - The neighborhoods with high per capita crime are in older neighborhoods. - The neighborhoods with high per capita crime are closer to Boston employment centers. - The neighborhoods with high per capita crime are in poorer neighborhoods (places where the median value of owner-occupied homes is lower) - The neighborhoods with high per capita crime have higher indexes of accessibility to radial highways. - The neighborhoods with high per capita crime are in areas where the pupil:teacher ratio is higher (typically corresponding to worse public education systems) - The neighborhoods with high per capita crime have higher tax rates.

Chapter 2, #10d

```
summary(Boston$crim)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
##   0.00632   0.08204   0.25651   3.61352   3.67708 88.97620
```

```
hist(Boston$crim, 100)
```



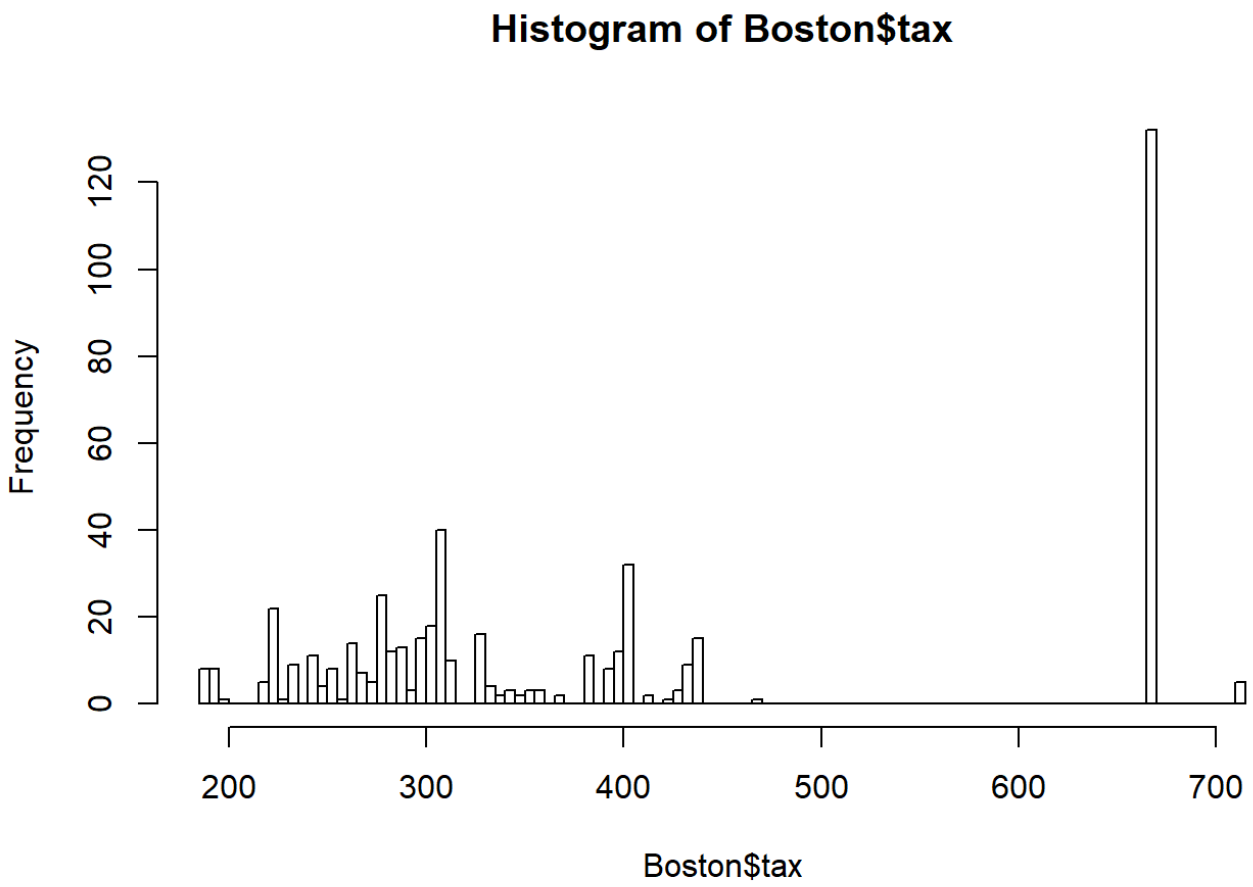

```
nrow(Boston[Boston$crim > 30, ])
```

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

```
range(Boston$crim)
```

```
## [1]  0.00632 88.97620
```

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



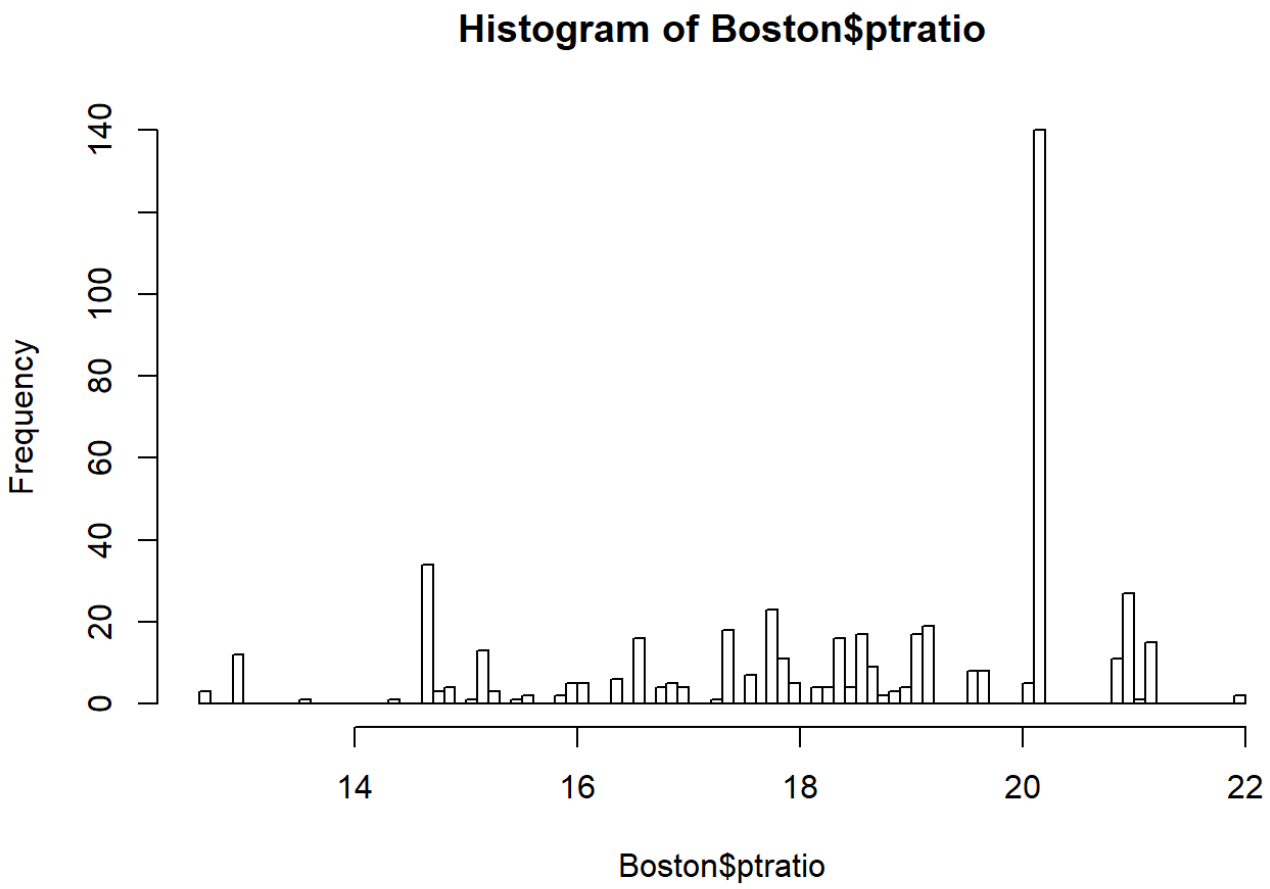

```
nrow(Boston[Boston$tax >600, ])
```

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

```
range(Boston$tax)
```

```
## [1] 187 711
```

```
hist(Boston$ptratio, 100)
```




```
nrow(Boston[Boston$ptratio <15,])
```

```
## [1] 58
```

```
nrow(Boston[Boston$ptratio >21,])
```

```
## [1] 18
```

```
range(Boston$ptratio)
```

```
## [1] 12.6 22.0
```

- There are a few neighborhoods where the per capita crime rate is very high, forming a long tail. The vast majority of neighborhoods have very low crime rates. There are only 8 neighborhoods where the per capita crime is greater than 30
- 137/506 or 27% of neighborhoods, have tax ratios over 600. All others have a tax rate below 500, indicating a large gap.
- Many neighborhoods have a ptratio around 20. However, with a range of 10 and a max of 22, indicating that the ptratio does not get much worse than the mean. There are more neighborhoods that have a ptratio less than 15 than neighborhoods with a ptratio greater than 21.

Chapter 2, #10e

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

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

There are 35 neighborhoods that bound the Charles River.

Chapter 2, #10f

```
summary(Boston$ptratio)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   12.60   17.40   19.05   18.46   20.20   22.00
```

The median pupil to teacher ratio for the 506 neighborhoods is 19.05.

Chapter 2, #10g

```
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
```

```
nrow(subset(Boston, medv == min(Boston$medv)))
```

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

```
t(subset(Boston, medv == min(Boston$medv)))
```

```
##              399      406
## crim     38.3518  67.9208
## zn        0.0000   0.0000
## indus    18.1000  18.1000
## chas      0.0000   0.0000
## nox       0.6930   0.6930
## rm        5.4530   5.6830
## age     100.0000 100.0000
## dis       1.4896   1.4254
## rad      24.0000  24.0000
## tax     666.0000 666.0000
## ptratio  20.2000  20.2000
## black   396.9000 384.9700
## lstat    30.5900  22.9800
## medv      5.0000   5.0000
```

The minimum medv value is 5, which occurs twice in this data set in row 399 and 406.

crim - Both have high crime, more than 3rd quartile zn - Both have low lot size indicating urban areas, both are at the min indus - Both have high industry, at 75th percentile chas - Both do not border Charles River nox - Both have more than the second quartile rm - Both are in the first quartile age - Both have very old homes, both are at the max rad - Both are at the max tax - Both are at the 75th percentile ptratio - Both are at the 75th percentile black - One is at the max, the other is above the first quartile lstat - Both are above the 3rd quartile

Overall, this information shows that the lowest medv values corresponds to poorer, urban, and more industrial areas that don't have the most desired features. The only big difference between these two neighborhoods is the proportion of black people in that town. This could be a an outlier, or it could show that there isn't that much of a relationship between those two variables.

Chapter 2, #10h

```
dim(subset(Boston, rm > 7))
```

```
## [1] 64 14
```

```
dim(subset(Boston, rm > 8))
```

```
## [1] 13 14
```

```
summary(subset(Boston, rm > 8))
```

```
##       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
```

There are 64 neighborhoods where rm is greater than 7, and there are 13 neighborhoods where rm is greater than 8.

When rm is greater than 8: - The range for all features, excluding the binary dummy variables, is much smaller, in part because it is a much smaller sample. - Dis is generally lower - Medv range is noticeably higher - Crim is significantly lower - Lstat is significantly lower

---

Chapter 3, #15a

```
crim.zn = lm(crim~zn, data=Boston)
crim.indus = lm(crim~indus, data=Boston)
crim.chas = lm(crim~chas, data=Boston)
crim.nox = lm(crim~nox, data=Boston)
crim.rm = lm(crim~rm, data=Boston)
crim.age = lm(crim~age, data=Boston)
crim.dis = lm(crim~dis, data=Boston)
crim.rad = lm(crim~rad, data=Boston)
crim.tax = lm(crim~tax, data=Boston)
crim.ptratio = lm(crim~ptratio, data=Boston)
crim.black = lm(crim~black, data=Boston)
crim.lstat = lm(crim~lstat, data=Boston)
crim.medv = lm(crim~medv, data=Boston)

summary(crim.zn)
```

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

```
summary(crim.indus)
```

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

```
summary(crim.chas)
```

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

```
summary(crim.nox)
```

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

```
summary(crim.rm)
```

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

```
summary(crim.age)
```

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

```
summary(crim.dis)
```

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

```
summary(crim.rad)
```

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

```
summary(crim.tax)
```

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

```
summary(crim.ptratio)
```

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

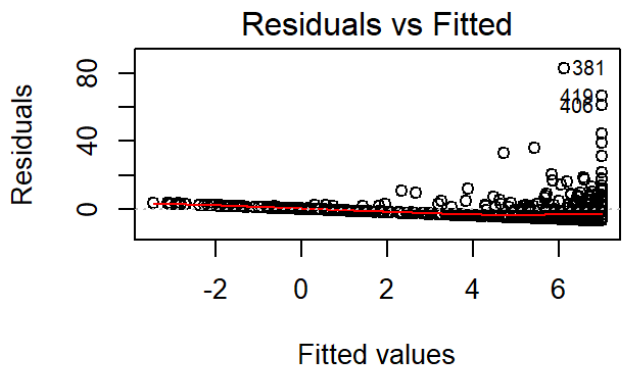
```
summary(crim.lstat)
```

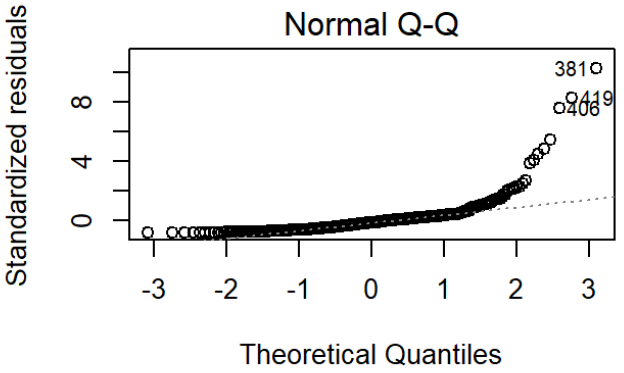
```
##
## Call:
## lm(formula = crim ~ lstat, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -13.925  -2.822  -0.664   1.079  82.862
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.33054    0.69376  -4.801 2.09e-06 ***
## lstat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

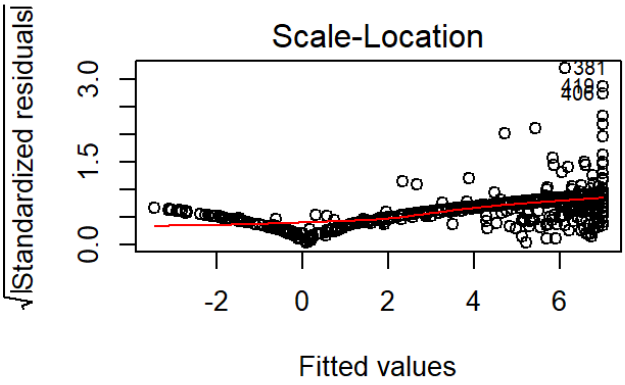
```
summary(crim.medv)
```

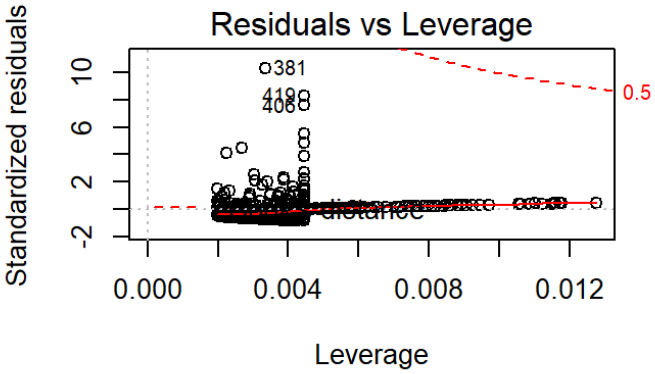
```
##
## Call:
## lm(formula = crim ~ medv, data = Boston)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -9.071 -4.022 -2.343  1.298 80.957
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 11.79654    0.93419   12.63   <2e-16 ***
## medv        -0.36316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

```
par(mfrow=c(2,2))
plot(crim.age) #significant
```

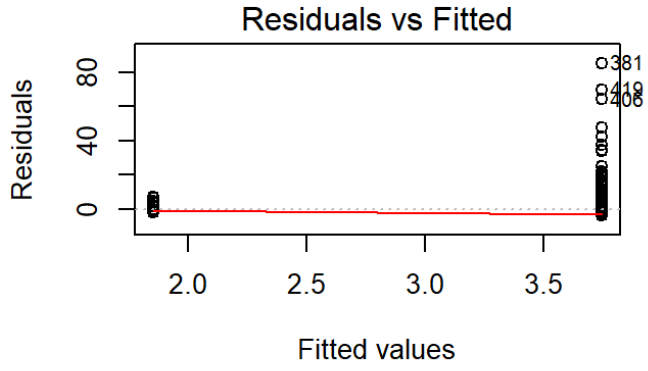


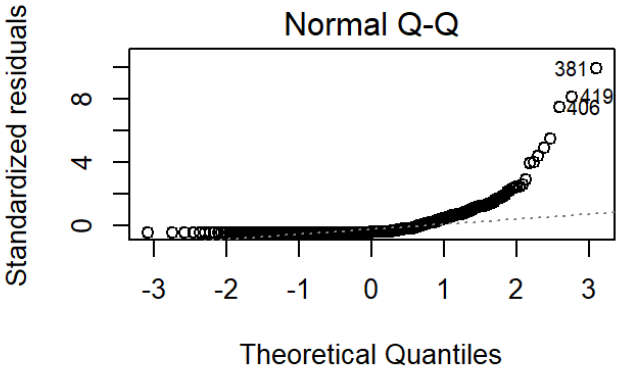


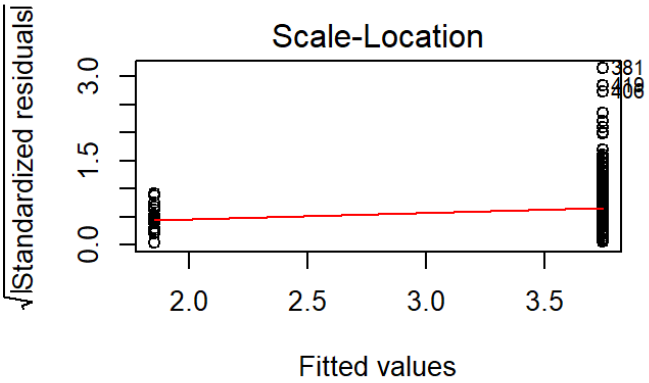


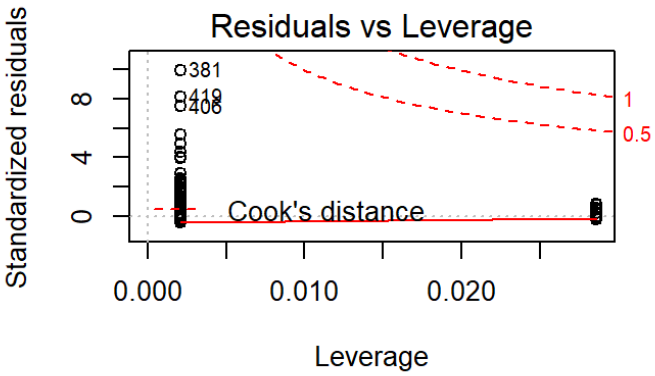



```
plot(crim.chas) #not significant
```









All of the linear models showed a statistically significant relationship between the predictor and response variable, per capita crime rate, except for chas. Bordering the Charles River and per capital crime rate do not seem to have a relationship, which is indicated by the high (.2094) p-value.

Chapter 3, #15b

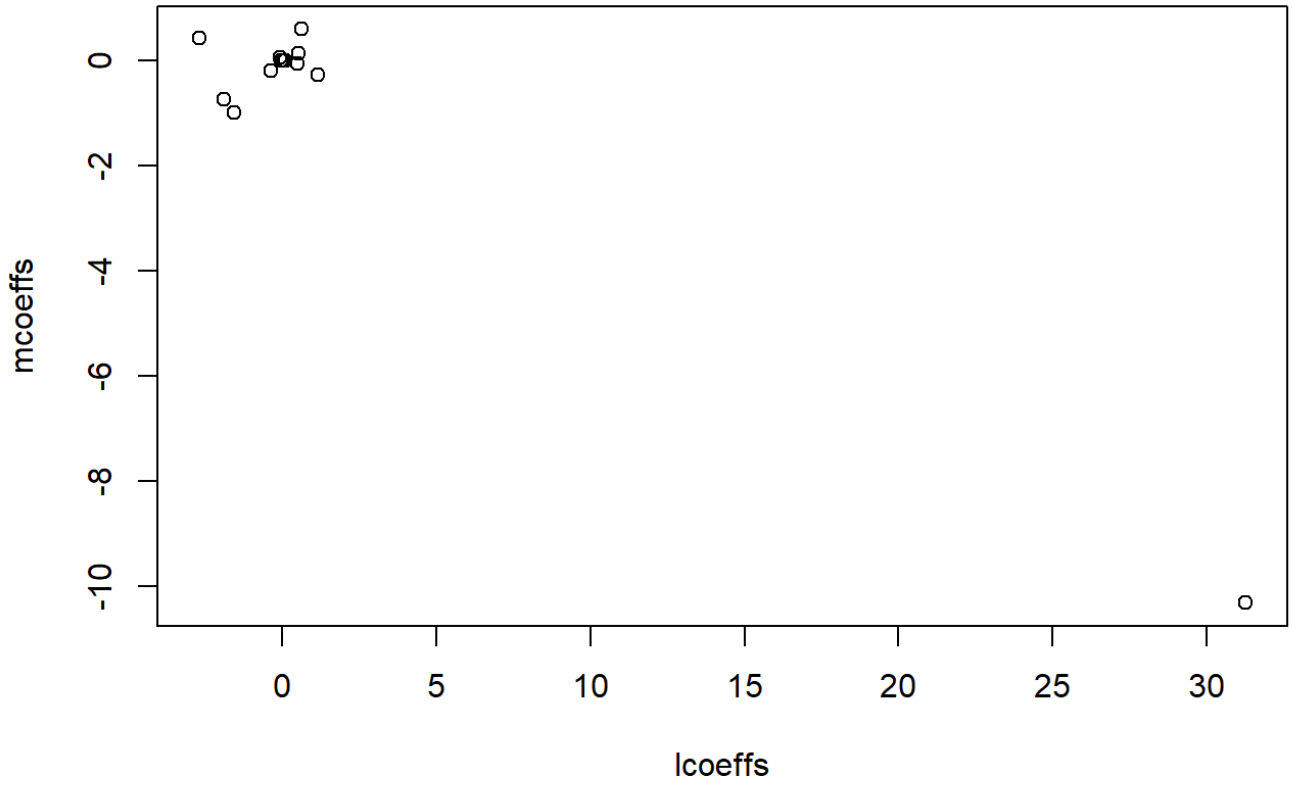
```
crim.all = lm(crim~., data=Boston)
summary(crim.all)
```

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

We can reject the null hypothesis for predictors where Pr(>|t|) is less than .05, indicated by one or more asterisks. We can reject the null hypothesis for zn, dis, rad, black, and medv.

Chapter 3, #15c

```
lcoeffs = c(coefficients(crim.zn)[2], coefficients(crim.indus)[2], coefficients(cr
im.chas)[2], coefficients(crim.nox)[2], coefficients(crim.rm)[2], coefficients(cri
m.age)[2], coefficients(crim.dis)[2], coefficients(crim.rad)[2], coefficients(crim
.tax)[2], coefficients(crim.ptratio)[2], coefficients(crim.black)[2], coefficients
(crim.lstat)[2], coefficients(crim.medv)[2])
mcoeffs = c(coefficients(crim.all)[2:14])
plot(lcoeffs, mcoeffs)
```




```
lcoeffs
```

```
##          zn       indus        chas         nox          rm         age
## -0.07393498  0.50977633 -1.89277655 31.24853120 -2.68405122  0.10778623
##         dis         rad         tax     ptratio       black       lstat
## -1.55090168  0.61791093  0.02974225  1.15198279 -0.03627964  0.54880478
##        medv
## -0.36315992
```

```
mcoeffs
```

```
##            zn         indus          chas           nox            rm
##   0.044855215  -0.063854824  -0.749133611 -10.313534912   0.430130506
##           age           dis           rad           tax       ptratio
##   0.001451643  -0.987175726   0.588208591  -0.003780016  -0.271080558
##         black         lstat          medv
##  -0.007537505   0.126211376  -0.198886821
```

It's a little hard to compare from the plot alone because one of the one data point (nox) is an extreme outlier. However, looking at the data itself, it is clear that there are significant differences between univariate regression and multiple regression. For example, univariate regression showed that rm had a -2.7 relationship with crim, but the multiple variable regression showed a positive 0.43 correlation. This difference is important! Univariate regression showed rm to be significant, but multivariate regression showed otherwise. Another example is nox, which shows positive 30 with univariate regression, but -10 with multiple regression.

Part of the reason for this is that a univariate regression model assumes Independence between features, which is often not the case and fundamentally different from a multiple variable regression.

Chapter 3, #15d

```
poly.zn= lm(Boston$crim ~ poly(Boston$zn,3))
poly.indus= lm(Boston$crim ~ poly(Boston$indus,3))
poly.nox= lm(Boston$crim ~ poly(Boston$nox,3))
poly.rm= lm(Boston$crim ~ poly(Boston$rm,3))
poly.age= lm(Boston$crim ~ poly(Boston$age,3))
poly.dis= lm(Boston$crim ~ poly(Boston$dis,3))
poly.rad= lm(Boston$crim ~ poly(Boston$rad,3))
poly.tax= lm(Boston$crim ~ poly(Boston$tax,3))
poly.ptratio= lm(Boston$crim ~ poly(Boston$ptratio,3))
poly.black= lm(Boston$crim ~ poly(Boston$black,3))
poly.lstat= lm(Boston$crim ~ poly(Boston$lstat,3))
poly.medv= lm(Boston$crim ~ poly(Boston$medv,3))

summary(poly.zn)
```

```
##
## Call:
## lm(formula = Boston$crim ~ poly(Boston$zn, 3))
##
## 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(Boston$zn, 3)1 -38.7498     8.3722  -4.628  4.7e-06 ***
## poly(Boston$zn, 3)2  23.9398     8.3722   2.859  0.00442 **
## poly(Boston$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
```

```
summary(poly.indus)
```

```
##
## Call:
## lm(formula = Boston$crim ~ poly(Boston$indus, 3))
##
## 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(Boston$indus, 3)1    78.591      7.423  10.587  < 2e-16 ***
## poly(Boston$indus, 3)2   -24.395      7.423  -3.286  0.00109 **
## poly(Boston$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
```

```
summary(poly.nox)
```

```
##
## Call:
## lm(formula = Boston$crim ~ poly(Boston$nox, 3))
##
## 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(Boston$nox, 3)1   81.3720     7.2336  11.249  < 2e-16 ***
## poly(Boston$nox, 3)2  -28.8286     7.2336  -3.985 7.74e-05 ***
## poly(Boston$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
```

```
summary(poly.rm)
```

```
##
## Call:
## lm(formula = Boston$crim ~ poly(Boston$rm, 3))
##
## 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(Boston$rm, 3)1 -42.3794     8.3297  -5.088 5.13e-07 ***
## poly(Boston$rm, 3)2  26.5768     8.3297   3.191  0.00151 **
## poly(Boston$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
```

```
summary(poly.age)
```

```
##
## Call:
## lm(formula = Boston$crim ~ poly(Boston$age, 3))
##
## 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(Boston$age, 3)1  68.1820     7.8397   8.697  < 2e-16 ***
## poly(Boston$age, 3)2  37.4845     7.8397   4.781 2.29e-06 ***
## poly(Boston$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
```

```
summary(poly.dis)
```

```
##
## Call:
## lm(formula = Boston$crim ~ poly(Boston$dis, 3))
##
## 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(Boston$dis, 3)1 -73.3886     7.3315 -10.010  < 2e-16 ***
## poly(Boston$dis, 3)2  56.3730     7.3315   7.689 7.87e-14 ***
## poly(Boston$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
```

```
summary(poly.rad)
```

```
##
## Call:
## lm(formula = Boston$crim ~ poly(Boston$rad, 3))
##
## 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(Boston$rad, 3)1 120.9074     6.6824  18.093  < 2e-16 ***
## poly(Boston$rad, 3)2  17.4923     6.6824   2.618  0.00912 **
## poly(Boston$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
```

```
summary(poly.tax)
```

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

```
summary(poly.ptratio)
```

```
##
## Call:
## lm(formula = Boston$crim ~ poly(Boston$ptratio, 3))
##
## 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(Boston$ptratio, 3)1   56.045      8.122   6.901 1.57e-11 ***
## poly(Boston$ptratio, 3)2   24.775      8.122   3.050  0.00241 **
## poly(Boston$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
```

```
summary(poly.black)
```

```
##
## Call:
## lm(formula = Boston$crim ~ poly(Boston$black, 3))
##
## 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(Boston$black, 3)1 -74.4312     7.9546  -9.357   <2e-16 ***
## poly(Boston$black, 3)2   5.9264     7.9546   0.745    0.457
## poly(Boston$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
```

```
summary(poly.lstat)
```

```
##
## Call:
## lm(formula = Boston$crim ~ poly(Boston$lstat, 3))
##
## 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(Boston$lstat, 3)1  88.0697     7.6294  11.543   <2e-16 ***
## poly(Boston$lstat, 3)2  15.8882     7.6294   2.082   0.0378 *
## poly(Boston$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
```

```
summary(poly.medv)
```

```
##
## Call:
## lm(formula = Boston$crim ~ poly(Boston$medv, 3))
##
## 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(Boston$medv, 3)1   -75.058      6.569 -11.426  < 2e-16 ***
## poly(Boston$medv, 3)2    88.086      6.569  13.409  < 2e-16 ***
## poly(Boston$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
```

Because the question asks about non linear, we know to use the poly function. And because the given formula goes to x^3, then we know to include additional polynomial terms up to the 3rd order.

- Chas is binary, so there's no way for there to be an observable nonlinear relationship.
- The cubic and quadratic coefficients for black are not significant, and thus it is unlikely that there is a nonlinear relationship between crim and black.
- The cubic coefficients for the features lstat, tax, rad, rm, and zn have p values greater than .05, indicating that they aren't statistically significant, but may be significant at the quadratic level.
- The cubic coefficients for the features medv, ptratio, dis, age, nox, and indus are statistically significant, indicating that a cubic relationship might exist.

Chapter 6, #9a

```
rm(list = ls())
library(ISLR)
set.seed(2)
train=sample(c(TRUE, FALSE), nrow(College), rep=TRUE)
Ctrain = College[train,]
Ctest=College[-train,]
```

Chapter 6, #9b

```
names(College)
```

```
##  [1] "Private"     "Apps"        "Accept"      "Enroll"      "Top10perc"
##  [6] "Top25perc"   "F.Undergrad" "P.Undergrad" "Outstate"    "Room.Board"
## [11] "Books"       "Personal"    "PhD"         "Terminal"    "S.F.Ratio"
## [16] "perc.alumni" "Expend"      "Grad.Rate"
```

```
lm.apps=lm(Apps~.,data=Ctrain)
lm.apps.predict=predict(lm.apps,Ctest)
sqrt(mean((lm.apps.predict - Ctest$Apps)^2))
```

```
## [1] 1095.979
```

For the linear model, the RMSE from the test data is 1096.

Chapter 6, #9c

```
library(glmnet)
```

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

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

```
## Loaded glmnet 2.0-10
```

```
xtrain= model.matrix(Apps~., Ctrain)[,-1]
xtest= model.matrix(Apps~., Ctest)[,-1]
ytrain= College$Apps[train]
ytest= College$Apps[-train]
grid=10^seq(10,-2,length=100)
ridge.mod = cv.glmnet(xtrain, ytrain, alpha=0)
ridge.best.lambda = ridge.mod$lambda.min
ridge.best.lambda
```

```
## [1] 372.4666
```

```
ridge.pred=predict(ridge.mod,s=ridge.best.lambda,newx=xtest)
sqrt(mean((ridge.pred-ytest)^2))
```

```
## [1] 1314.2
```

Used the train data to get the best lambda (372.5). Then used that to find the RMSE from the test data (1314.2).

Chapter 6, #9d

```
lasso.mod = cv.glmnet(xtrain, ytrain, alpha=1)
lasso.best.lambda = lasso.mod$lambda.min
lasso.best.lambda
```

```
## [1] 18.53775
```

```
lasso.pred=predict(lasso.mod,s=lasso.best.lambda,newx=xtest)
sqrt(mean((lasso.pred-ytest)^2))
```

```
## [1] 1108.553
```

```
lasso.mod2 = glmnet(model.matrix(Apps~., data = College), College$Apps, alpha = 1,
lambda = grid)
lasso.coef=predict(lasso.mod2, type="coefficients", s=lasso.best.lambda)[1:18,]
lasso.coef
```

```
##   (Intercept)   (Intercept)    PrivateYes        Accept        Enroll
## -5.772155e+02  0.000000e+00 -4.377795e+02  1.473693e+00 -2.591417e-01
##     Top10perc     Top25perc   F.Undergrad   P.Undergrad      Outstate
##  3.594421e+01 -3.989435e+00  0.000000e+00  2.622411e-02 -6.223558e-02
##    Room.Board         Books      Personal           PhD      Terminal
##  1.286666e-01  0.000000e+00  2.843357e-03 -6.034700e+00 -3.249639e+00
##     S.F.Ratio   perc.alumni        Expend
##  6.034228e+00 -8.678876e-01  7.080715e-02
```

Used the train data to get the best lambda (18.5). Then used that to find the RMSE from the test data (1108.5).

The 2 coefficient estimates that are zero are Books and F. Undergrad. The 14 nonzero coefficient estimates are PrivateYes, Accept, Enroll, Top10perc, Top25perc, P.Undergrad, Outstate, Room.Board, Personal, PhD, and Terminal, S.F.Ratio, perc.alumni, and Expend.

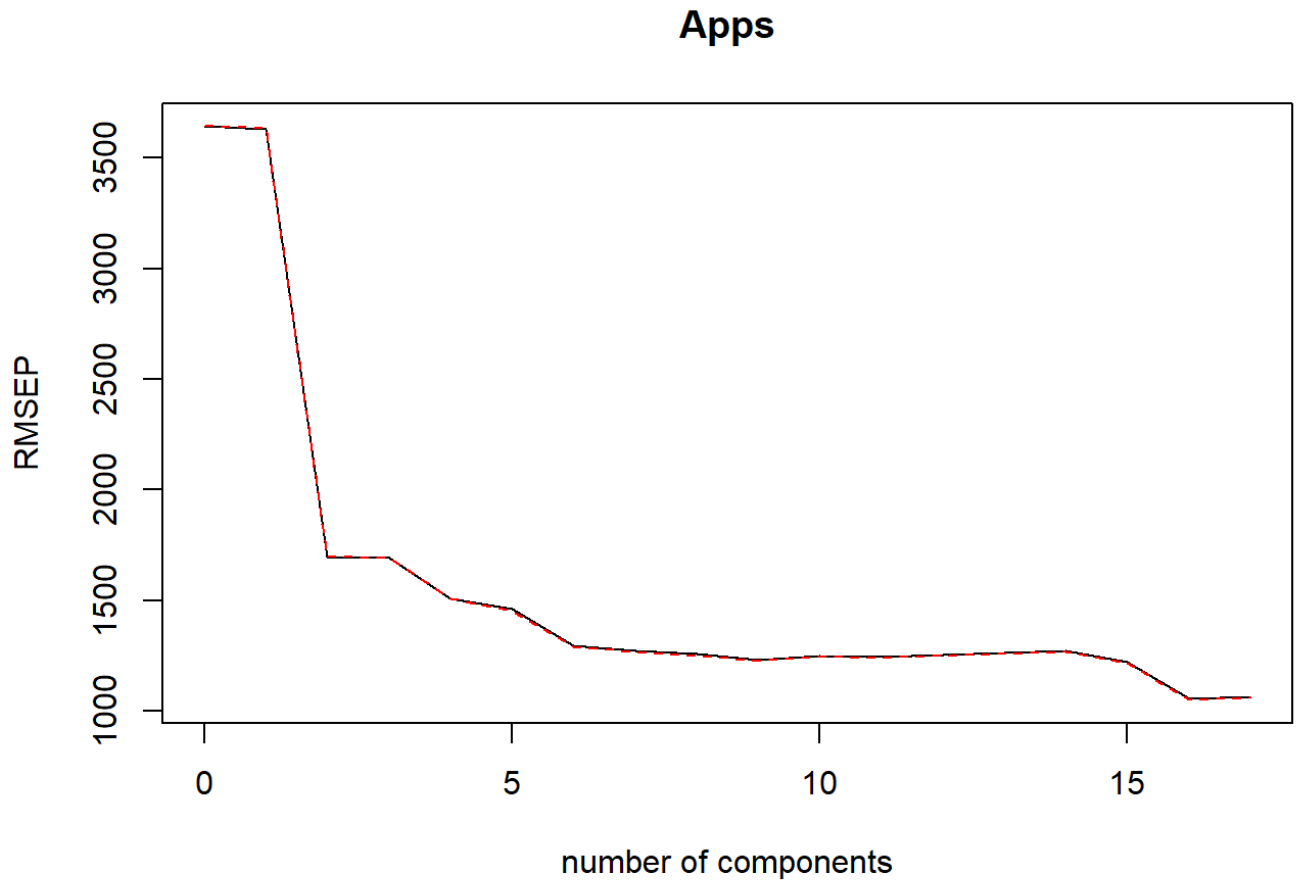
Chapter 6, #9e

```
library(pls)
```

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

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

```
set.seed(2)
pcr.apps = pcr(Apps~., data=Ctrain, scale=TRUE, validation="CV")
validationplot(pcr.apps,val.type="RMSEP")
```




```
summary(pcr.apps)
```

```
## Data:    X dimension: 391 17
##  Y dimension: 391 1
## Fit method: svdpc
## Number of components considered: 17
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV            3643     3631     1698     1692     1508     1462     1295
## adjCV         3643     3635     1695     1694     1506     1452     1290
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV        1272     1258     1233      1249      1245      1253      1265
## adjCV     1265     1250     1227      1246      1241      1249      1260
##        14 comps  15 comps  16 comps  17 comps
## CV         1271      1221      1053      1066
## adjCV      1266      1215      1049      1061
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X      31.071    56.82    64.00    70.05    75.30    80.20    84.04
## Apps    2.441    79.07    79.33    83.50    84.61    87.95    88.76
##       8 comps  9 comps  10 comps  11 comps  12 comps  13 comps  14 comps
## X       87.43    90.38     92.96     95.10     96.82     98.02     98.96
## Apps    89.08    89.24     89.46     89.65     89.66     89.66     89.66
##       15 comps  16 comps  17 comps
## X        99.45     99.88    100.00
## Apps     90.60     92.65     92.82
```

```
pcr.apps.predict = predict(pcr.apps, Ctest, ncomp = 16)
sqrt(mean((pcr.apps.predict-ytest)^2))
```

```
## [1] 1131.499
```

The minimum RMSE in this situation is M=16, which only reduced dimensionality by one. This produced an RMSE of 1131.5

Chapter 6, #9f

```
pls.apps = plsr(Apps~., data=Ctrain, scale=T, validation = "CV")
summary(pls.apps)
```

```
## Data:    X dimension: 391 17
##  Y dimension: 391 1
## Fit method: kernelpls
## Number of components considered: 17
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV            3643     1544     1280     1196     1185     1156     1082
## adjCV         3643     1540     1283     1193     1178     1146     1078
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV        1071     1065     1064      1067      1068      1073      1075
## adjCV     1066     1061     1059      1062      1063      1068      1069
##        14 comps  15 comps  16 comps  17 comps
## CV         1073      1073      1073      1073
## adjCV      1067      1067      1067      1067
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X       25.87    40.12    62.15    66.34    69.55    73.54    77.97
## Apps    82.91    88.32    90.11    91.05    92.12    92.61    92.73
##       8 comps  9 comps  10 comps  11 comps  12 comps  13 comps  14 comps
## X       81.42    84.30     86.93     89.67     91.84     93.49     95.22
## Apps    92.75    92.77     92.79     92.80     92.81     92.82     92.82
##       15 comps  16 comps  17 comps
## X        97.29     98.92    100.00
## Apps     92.82     92.82     92.82
```

```
pls.apps.predict = predict(pls.apps, Ctest, ncomp = 9)
sqrt(mean((pls.apps.predict-ytest)^2))
```

```
## [1] 1105.277
```

CV error is lowest at M=9, so we put ncom=9.

Test RMSE is 1105.3

Chapter 6, #9g

```
cat("Linear Regression produced an RMSE of", sqrt(mean((lm.apps.predict - Ctest$App
s)^2)), ". ")
```

```
## Linear Regression produced an RMSE of 1095.979 .
```

```
cat("Ridge produced an RMSE of", sqrt(mean((ridge.pred-ytest)^2)), ". ")
```

```
## Ridge produced an RMSE of 1314.2 .
```

```
cat("Lasso produced an RMSE of", sqrt(mean((lasso.pred-ytest)^2)), ". ")
```

```
## Lasso produced an RMSE of 1108.553 .
```

```
cat("PCR produced an RMSE of", sqrt(mean((pcr.apps.predict-ytest)^2)), ". ")
```

```
## PCR produced an RMSE of 1131.499 .
```

```
cat("PLS produced an RMSE of", sqrt(mean((pls.apps.predict-ytest)^2)), ". ")
```

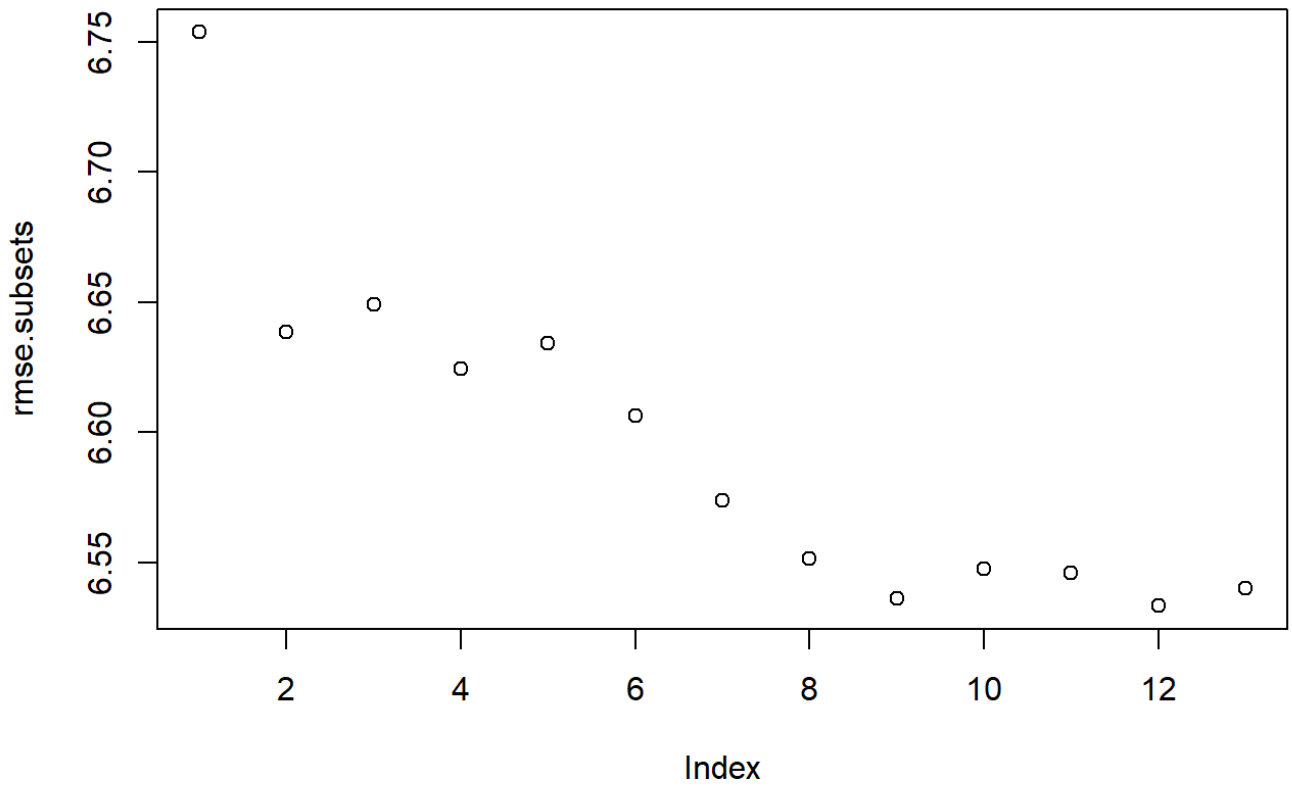
```
## PLS produced an RMSE of 1105.277 .
```

Aside from Ridge having a slightly larger RMSE, the others were fairly comparable, with linear coming out on top with the lowest RMSE.

Chapter 6, #11a

```
rm(list = ls())
library(ISLR)
library(glmnet)
library(pls)
library(leaps)
set.seed(2)

#best subsets
predict.regsubsets = function(object, newdata, id, ...) {
    form = as.formula(object$call[[2]])
    mat = model.matrix(form, newdata)
    coefi = coef(object, id = id)
    xvars=names(coefi)
    mat[, xvars] %*% coefi
}
k=10
folds=sample(1:k,nrow(Boston),replace =TRUE)
cv.errors=matrix(NA,k,13, dimnames=list(NULL, paste(1:13)))
for(j in 1:k){
    best.fit=regsubsets(crim~.,data=Boston [folds !=j,],nvmax =13)
    for(i in 1:13){
        pred=predict(best.fit,Boston[folds==j,],id=i)
        cv.errors[j,i] = mean( (Boston$crim[folds ==j]-pred)^2)
  }
}
rmse.subsets=sqrt(apply(cv.errors ,2, mean))
plot(rmse.subsets)
```



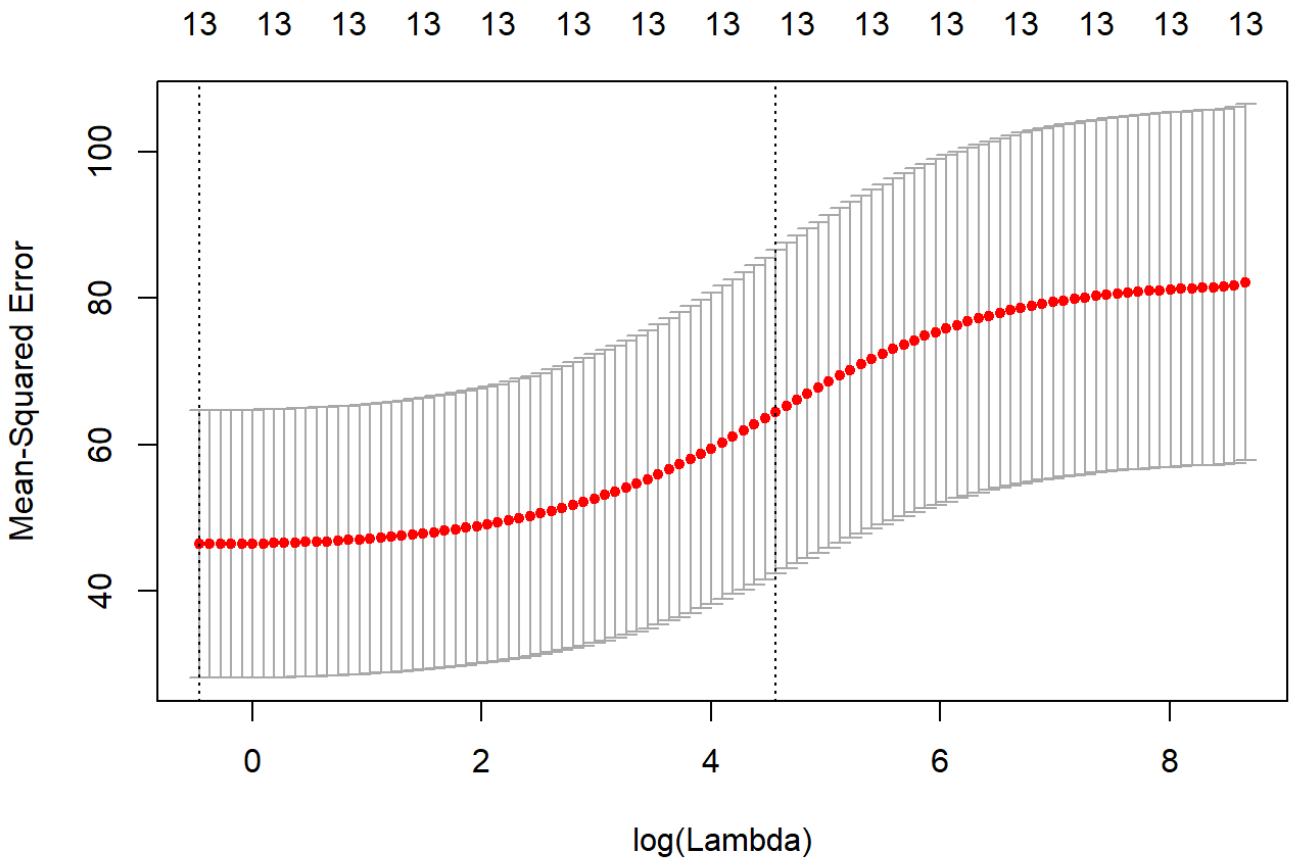

```
a = which.min(rmse.subsets)
a = rmse.subsets[a]
a
```

```
##      12
## 6.53343
```

Best subsets produced an RMSE of 6.53

```
train=sample(c(TRUE, FALSE), nrow(Boston), rep=TRUE)
B.train = Boston[train,]
B.test=Boston[-train,]

#Ridge
xtrain= model.matrix(crim~., B.train)[,-1]
xtest= model.matrix(crim~., B.test)[,-1]
ytrain= Boston$crim[train]
ytest= Boston$crim[-train]
grid=10^seq(10,-2,length=100)
ridge.mod = cv.glmnet(xtrain, ytrain, alpha=0)
plot(ridge.mod)
```




```
ridge.best.lambda = ridge.mod$lambda.min
ridge.best.lambda
```

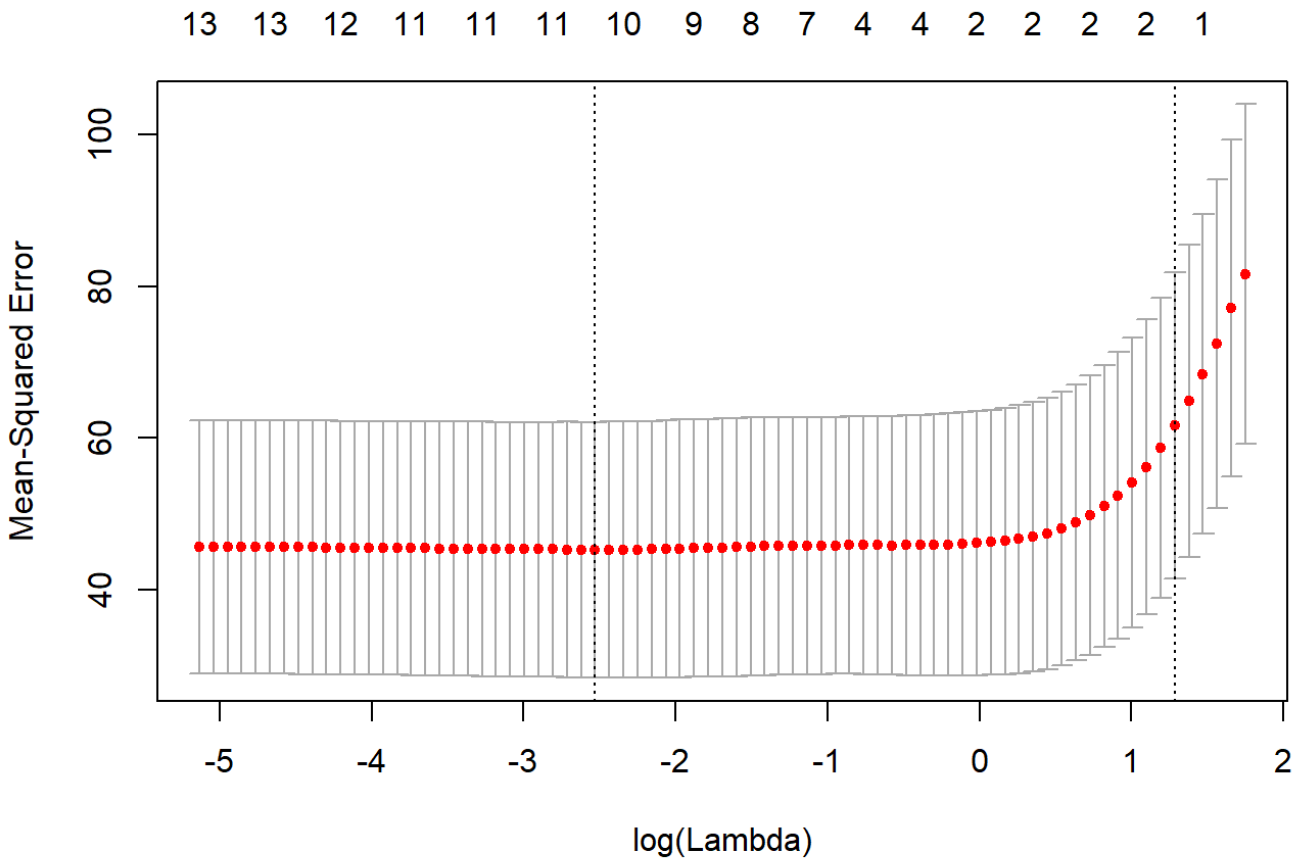
```
## [1] 0.6311884
```

```
ridge.pred=predict(ridge.mod,s=ridge.best.lambda,newx=xtest)
b = sqrt(mean((ridge.pred-ytest)^2))
b
```

```
## [1] 6.45468
```

Ridge produced an RMSE of 6.45

```
#Lasso
lasso.mod = cv.glmnet(xtrain, ytrain, alpha=1)
plot(lasso.mod)
```




```
lasso.best.lambda = lasso.mod$lambda.min
lasso.best.lambda
```

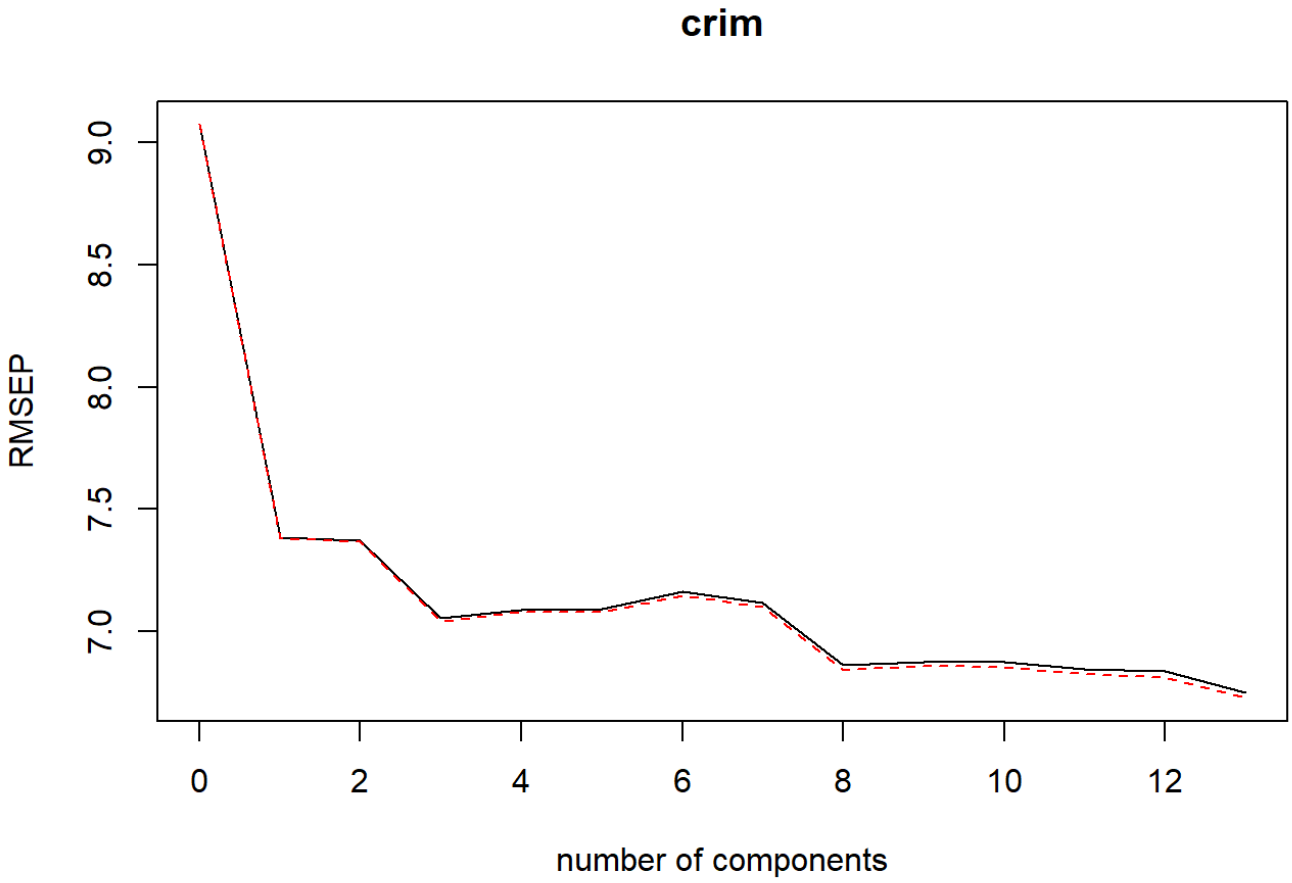
```
## [1] 0.07964694
```

```
lasso.pred=predict(lasso.mod,s=lasso.best.lambda,newx=xtest)
c = sqrt(mean((lasso.pred-ytest)^2))
c
```

```
## [1] 6.45811
```

Lasso produced an RMSE of 6.45

```
#PCR
pcr.crim = pcr(crim~., data=B.train, scale=TRUE, validation="CV")
validationplot(pcr.crim,val.type="RMSEP")
```




```
summary(pcr.crim)
```

```
## 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           9.073    7.384    7.371    7.053    7.089    7.091    7.165
## adjCV        9.073    7.380    7.368    7.043    7.078    7.081    7.148
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       7.118    6.861    6.877     6.874     6.845     6.836     6.751
## adjCV    7.102    6.843    6.860     6.855     6.826     6.814     6.728
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X       47.94    60.99    69.79    77.11    83.34    88.46    91.65
## crim    34.50    35.02    41.50    41.74    41.88    42.72    43.46
##       8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## X        94.0    95.73     97.29     98.57     99.52    100.00
## crim     47.5    47.51     48.17     48.73     49.35     50.97
```

```
pcr.crim.predict = predict(pcr.crim, B.test, ncomp = 13)
d = sqrt(mean((pcr.crim.predict-ytest)^2))
d
```

```
## [1] 6.48294
```

Best subsets produced an RMSE of 6.48

```
c(a,b,c,d)
```

```
##      12
## 6.53343 6.45468 6.45811 6.48294
```

Ridge produced the lowest RMSE of 6.454, but they were all very close.

Chapter 6, #11b

I would recommend using lasso and ridge on this data set, as they both present the lowest RMSEs when compared to other options for this data set. Though Ridge was slightly better, Lasso and Ridge are very similar, so it would not be difficult to run them both and see which works better. If new data points were presented, Lasso has a lower chance of over-fitting than Ridge, which is partly why both are worth considering.

Chapter 6, #11c

Yes, Ridge involves all of the features in the data set. However, running Lasso to compare against ridge would be a good idea because it zeros out unnecessary coefficients.

Chapter 8, #8a

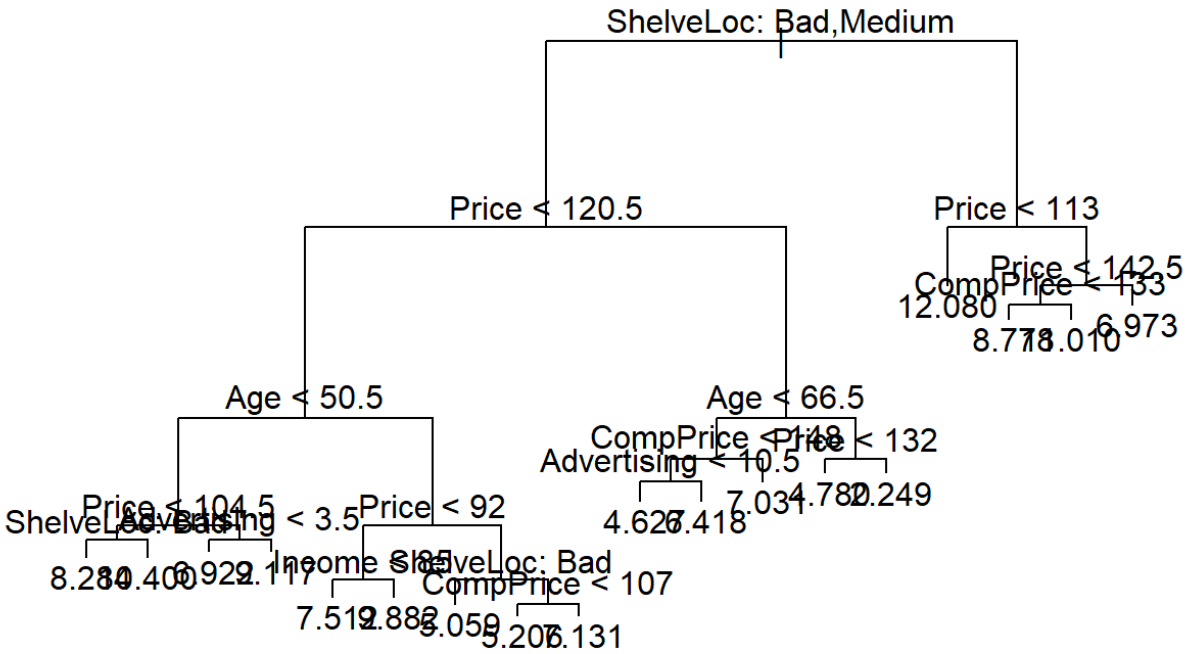
```
rm(list = ls())
library(tree)
library(ISLR)
set.seed(1)
train = sample(1:nrow(Carseats), nrow(Carseats)/2)
Strain = Carseats[train,]
Stest = Carseats[-train,]
```

Chapter 8, #8b

```
tree.sales=tree(Sales~.,Carseats ,subset =train)
summary(tree.sales)
```

```
##
## Regression tree:
## tree(formula = Sales ~ ., data = Carseats, subset = train)
## Variables actually used in tree construction:
## [1] "ShelveLoc"   "Price"       "Age"         "Advertising" "Income"
## [6] "CompPrice"
## Number of terminal nodes:  18
## Residual mean deviance:  2.36 = 429.5 / 182
## Distribution of residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -4.2570 -1.0360  0.1024  0.0000  0.9301  3.9130
```

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




```
predict.sales.tree = predict(tree.sales, Stest)
sqrt(mean((predict.sales.tree-Stest$Sales)^2))
```

```
## [1] 2.036884
```

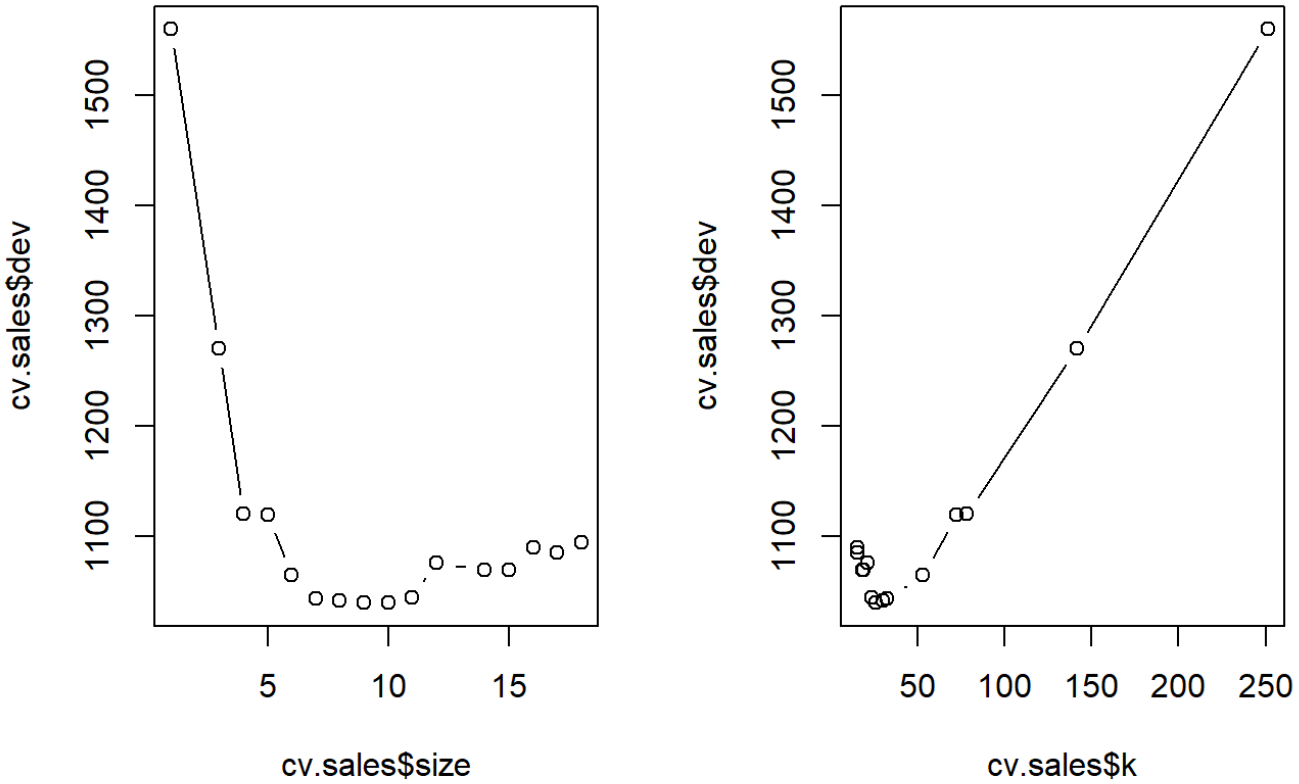
Fitting a regression tree produces a test RMSE of 2.04.

Chapter 8, #8c

```
cv.sales=cv.tree(tree.sales)
cv.sales
```

```
## $size
##  [1] 18 17 16 15 14 12 11 10  9  8  7  6  5  4  3  1
##
## $dev
##  [1] 1094.537 1085.178 1089.409 1069.294 1069.294 1075.806 1044.469
##  [8] 1039.212 1039.212 1041.308 1043.459 1064.162 1119.046 1120.344
## [15] 1270.012 1560.273
##
## $k
##  [1]      -Inf  15.48181  15.53599  18.69038  18.74886  21.05038  23.79480
##  [8]  25.78579  26.01210  30.10435  32.74801  53.28569  72.33061  78.19599
## [15] 141.73781 251.22901
##
## $method
## [1] "deviance"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

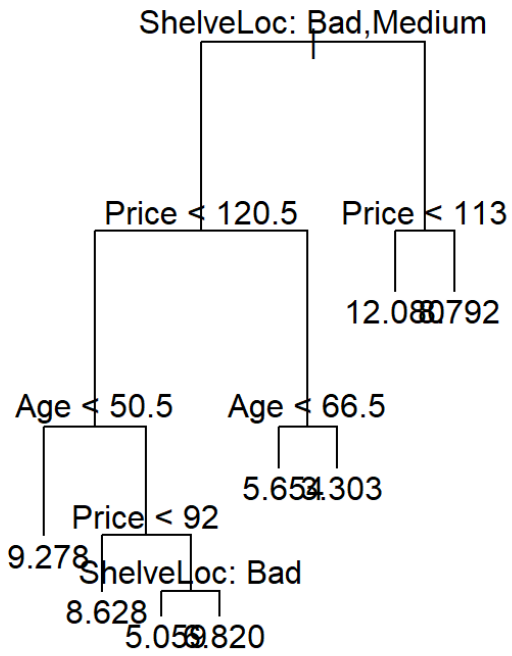
```
par(mfrow=c(1,2))
plot(cv.sales$size,cv.sales$dev,type="b")
plot(cv.sales$k,cv.sales$dev,type="b")
```



```
prune.sales=prune.tree(tree.sales,best=8)
plot(prune.sales)
text(prune.sales, pretty = 0)

predict.sales.prune = predict(prune.sales, Stest)
sqrt(mean((predict.sales.prune-Stest$Sales)^2))
```

```
## [1] 2.256291
```



The lowest dev is found with either 8 or 9 nodes, so that's what cross-validation selects. However, because simpler is always best, we're going to go with 8 nodes.

This produced a test RMSE of 2.256, which meas that pruning does not improve the test error.

Chapter 8, #8d

```
library(randomForest)
```

```
## randomForest 4.6-12
```

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

```
##
## Attaching package: 'randomForest'
```

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

```
bag.sales=randomForest(Sales~., Strain, mtry=10, ntree=500, importance =TRUE)

predict.sales.bag = predict(bag.sales, Stest)
sqrt(mean((predict.sales.bag - Stest$Sales)^2))
```

```
## [1] 1.613806
```

```
importance(bag.sales)
```

```
##                %IncMSE IncNodePurity
## CompPrice   14.4124562    133.731797
## Income       6.5147532     74.346961
## Advertising 15.7607104    117.822651
## Population   0.6031237     60.227867
## Price       57.8206926    514.802084
## ShelveLoc   43.0486065    319.117972
## Age         19.8789659    192.880596
## Education    2.9319161     39.490093
## Urban       -3.1300102      8.695529
## US           7.6298722     15.723975
```

The bagging test RMSE is 1.61, an improvement from previous models.

The most important indicators of Sales are Price, ShelveLoc, Age, Advertising and CopmPrice (in that order).

Chapter 8, #8e

```
rf.sales=randomForest(Sales~.,Strain, mtry=4, ntree=500, importance =TRUE)
predict.sales.rf = predict(rf.sales, newdata=Stest)
sqrt(mean((predict.sales.rf-Stest$Sales)^2))
```

```
## [1] 1.713418
```

```
importance(rf.sales)
```

```
##               %IncMSE IncNodePurity
## CompPrice    8.729129     129.37990
## Income       3.530901     115.77217
## Advertising 14.933751     136.46502
## Population   1.580225      87.31427
## Price       47.270421     417.31691
## ShelveLoc   34.562220     265.80282
## Age         18.650448     196.72434
## Education    2.352247      59.53539
## Urban       -2.652287      14.11129
## US           7.139449      24.16022
```

The test RMSE when M=4 was found to be 1.71. The most important indicators of Sales (when M=4) are Price, ShelveLoc, Age, and Advertising (in that order). As M increases, the test error also decreases, and the result using bagging ultimately produces the lowest test RMSE. The importance of some features also increases. This specifically occurs to features that are included in higher M models but excluded in lower M models, such as CompPrice.

Chapter 8, #11a

```
rm(list = ls())
library(ISLR)
train = 1:1000
Caravan$Purchase=ifelse(Caravan$Purchase=="Yes", 1, 0)
C.train = Caravan[train,]
C.test = Caravan[-train,]
```

Chapter 8, #11b

```
library(gbm)
```

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

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

```
## Loading required package: splines
```

```
## Loading required package: parallel
```

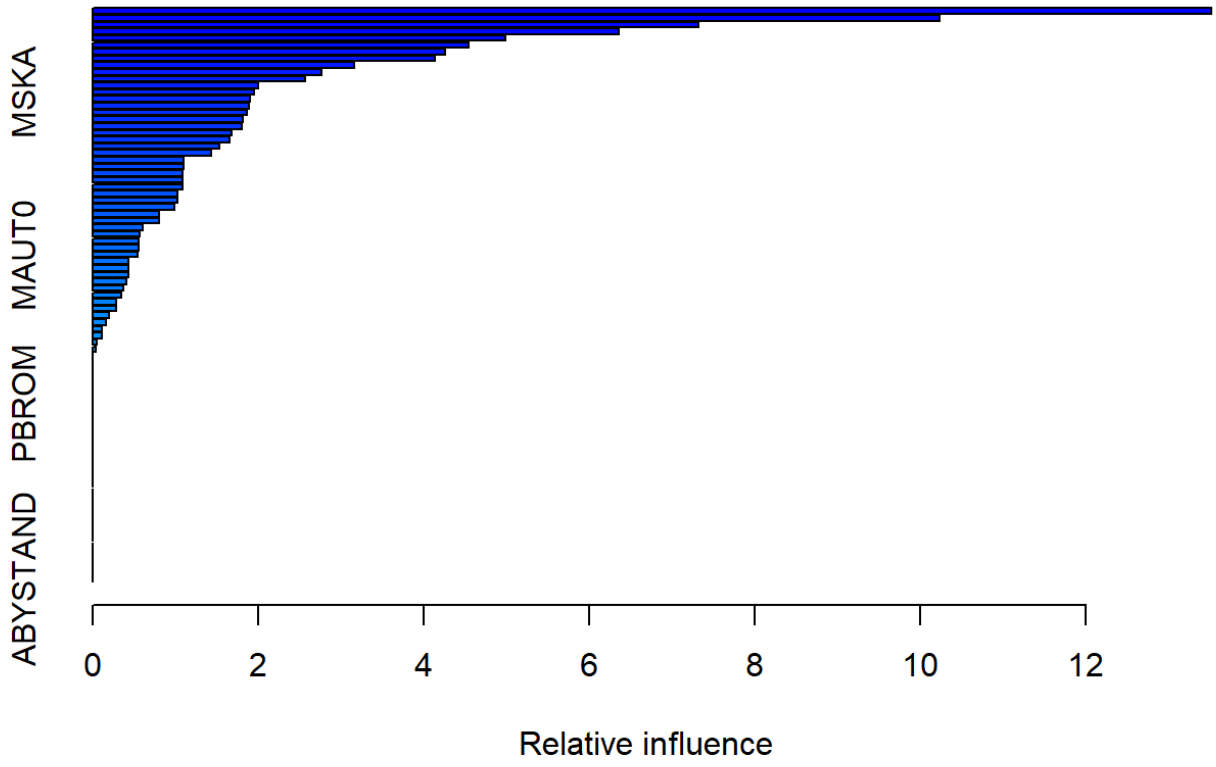
```
## Loaded gbm 2.1.3
```

```
set.seed(1)
boost.caravan=gbm(Purchase~.,data=C.train, distribution = "gaussian", n.trees=1000,
shrinkage=0.01)
```

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

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

```
summary(boost.caravan)
```




```
##                var     rel.inf
## PPERSAUT PPERSAUT 13.51824557
## MKOOPKLA MKOOPKLA 10.24062778
## MOPLHOOG MOPLHOOG  7.32689780
## MBERMIDD MBERMIDD  6.35820558
## PBRAND     PBRAND  4.98826360
## ABRAND     ABRAND  4.54504653
## MGODGE     MGODGE  4.26496875
## MINK3045 MINK3045  4.13253907
## PWAPART   PWAPART  3.15612877
## MAUT1       MAUT1  2.76929763
## MOSTYPE   MOSTYPE  2.56937935
## MAUT2       MAUT2  1.99879666
## MSKA         MSKA  1.94618539
## MBERARBG MBERARBG  1.89917331
## PBYSTAND PBYSTAND  1.88591514
## MINKGEM   MINKGEM  1.87131472
## MGODOV     MGODOV  1.81673309
## MGODPR     MGODPR  1.80814745
```

```
## MFWEKIND MFWEKIND  1.67884570
## MSKC         MSKC  1.65075962
## MBERHOOG MBERHOOG  1.53559951
## MSKB1       MSKB1  1.43339514
## MOPLMIDD MOPLMIDD  1.10617074
## MHHUUR     MHHUUR  1.09608784
## MRELGE     MRELGE  1.09039794
## MINK7512 MINK7512  1.08772012
## MZFONDS   MZFONDS  1.08427551
## MGODRK     MGODRK  1.03126657
## MINK4575 MINK4575  1.02492795
## MZPART     MZPART  0.98536712
## MRELOV     MRELOV  0.80356854
## MFGEKIND MFGEKIND  0.80335689
## MBERARBO MBERARBO  0.60909852
## APERSAUT APERSAUT  0.56707821
## MGEMOMV   MGEMOMV  0.55589456
## MOSHOOFD MOSHOOFD  0.55498375
## MAUT0       MAUT0  0.54748481
## PMOTSCO   PMOTSCO  0.43362597
## MSKB2       MSKB2  0.43075446
## MSKD         MSKD  0.42751490
## MINK123M MINK123M  0.40920707
## MINKM30   MINKM30  0.36996576
## MHKOOP     MHKOOP  0.34941518
## MBERBOER MBERBOER  0.28967068
## MFALLEEN MFALLEEN  0.28877552
## MGEMLEEF MGEMLEEF  0.20084195
## MOPLLAAG MOPLLAAG  0.15750616
## MBERZELF MBERZELF  0.11203381
## PLEVEN     PLEVEN  0.11030994
## MRELSA     MRELSA  0.04500507
## MAANTHUI MAANTHUI  0.03322830
## PWABEDR   PWABEDR  0.00000000
## PWALAND   PWALAND  0.00000000
## PBESAUT   PBESAUT  0.00000000
## PVRAAUT   PVRAAUT  0.00000000
## PAANHANG PAANHANG  0.00000000
## PTRACTOR PTRACTOR  0.00000000
## PWERKT     PWERKT  0.00000000
## PBROM       PBROM  0.00000000
## PPERSONG PPERSONG  0.00000000
## PGEZONG   PGEZONG  0.00000000
## PWAOREG   PWAOREG  0.00000000
## PZEILPL   PZEILPL  0.00000000
## PPLEZIER PPLEZIER  0.00000000
## PFIETS     PFIETS  0.00000000
## PINBOED   PINBOED  0.00000000
## AWAPART   AWAPART  0.00000000
## AWABEDR   AWABEDR  0.00000000
## AWALAND   AWALAND  0.00000000
## ABESAUT   ABESAUT  0.00000000
## AMOTSCO   AMOTSCO  0.00000000
## AVRAAUT   AVRAAUT  0.00000000
## AAANHANG AAANHANG  0.00000000
## ATRACTOR ATRACTOR  0.00000000
## AWERKT     AWERKT  0.00000000
```

```
## ABROM       ABROM  0.00000000
## ALEVEN     ALEVEN  0.00000000
## APERSONG APERSONG  0.00000000
## AGEZONG   AGEZONG  0.00000000
## AWAOREG   AWAOREG  0.00000000
## AZEILPL   AZEILPL  0.00000000
## APLEZIER APLEZIER  0.00000000
## AFIETS     AFIETS  0.00000000
## AINBOED   AINBOED  0.00000000
## ABYSTAND ABYSTAND  0.00000000
```

The predictors that appear to be the most important are: PPERSAUT, MKOOPKLA, MOPLHOOG, MBERMIDD (in that order).

Chapter 8, #11c

```
predict.caravan = predict(boost.caravan, newdata=C.test, type="response", n.trees=
1000)
predict.caravan.20 = ifelse(predict.caravan>0.2, 1, 0)
table(C.test$Purchase,predict.caravan.20)
```

```
##    predict.caravan.20
##        0    1
##   0 4493   40
##   1  278   11
```

```
11/(11+40)
```

```
## [1] 0.2156863
```

If you estimate that a person will make a purchase if the estimated probability of purchase is greater than 20%, then you will be correct 21.6% of the time, according to Boosting.

```
loglm.purchase = glm(Purchase~., data=C.train, family=binomial)
```

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

```
loglm.purchase.predict = predict(loglm.purchase, C.test, type= "response")
```

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

```
logpredict = ifelse(loglm.purchase.predict > 0.2, 1, 0)
table(C.test$Purchase, logpredict)
```

```
##     logpredict
##         0    1
##    0 4183  350
##    1  231   58
```

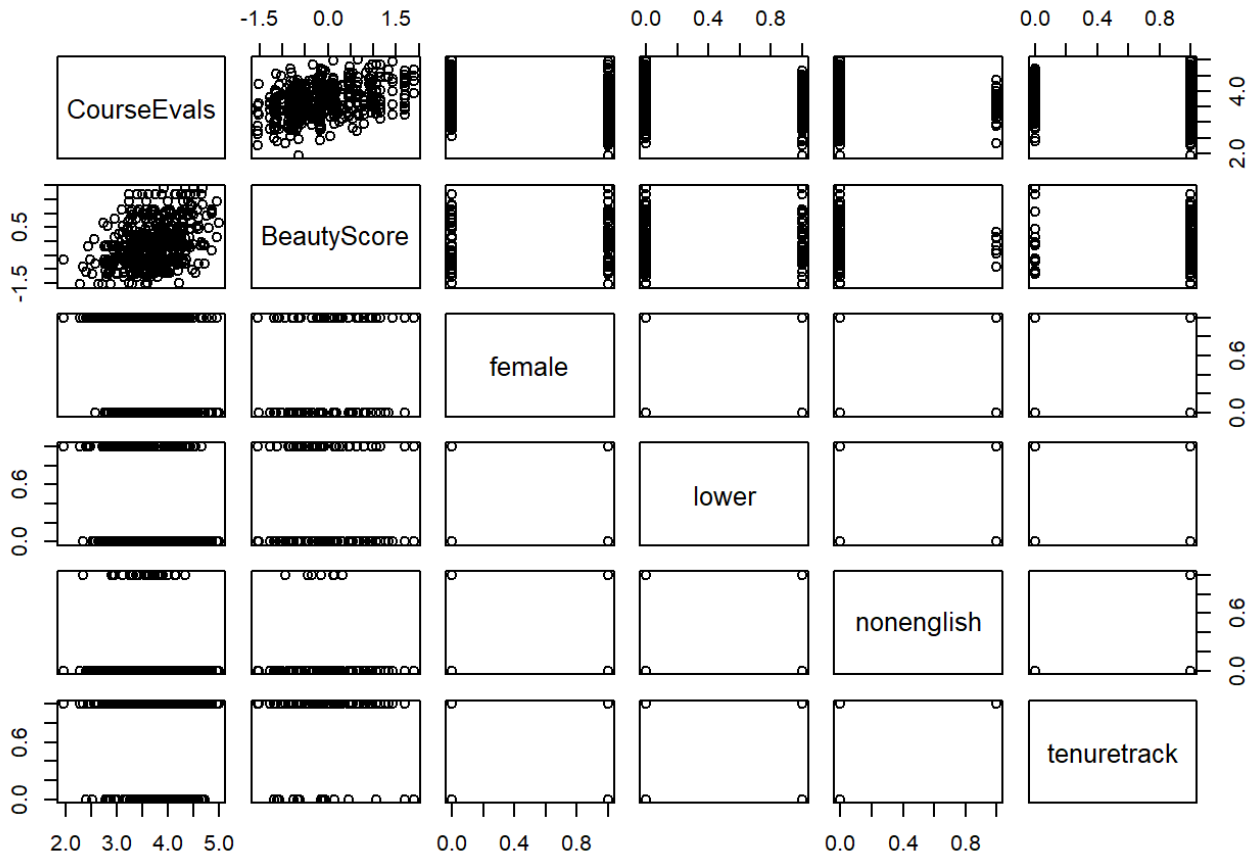
```
58/(58+350)
```

```
## [1] 0.1421569
```

If you estimate that a person will make a purchase if the estimated probability of purchase is greater than 20%, then you will be correct only 14.2% of the time, according to a logistic regression. This is lower than Boosting, and suggests that Boosting is more accurate.

Exam Problem 1.1

```
rm(list = ls())
setwd("~/R")
beauty <- read.csv("BeautyData.csv")
pairs(beauty)
```



```r
library(tree)
set.seed(1)
train = sample(1:nrow(beauty), nrow(beauty)/2)
Btrain = beauty[train,]
Btest = beauty[-train,]

#linear
lm.evals=lm(CourseEvals~.,data=Btrain)
lm.evals.predict=predict(lm.evals,Btest)
sqrt(mean((lm.evals.predict - Btest$CourseEvals)^2))
```

```
## [1] 0.42795
```

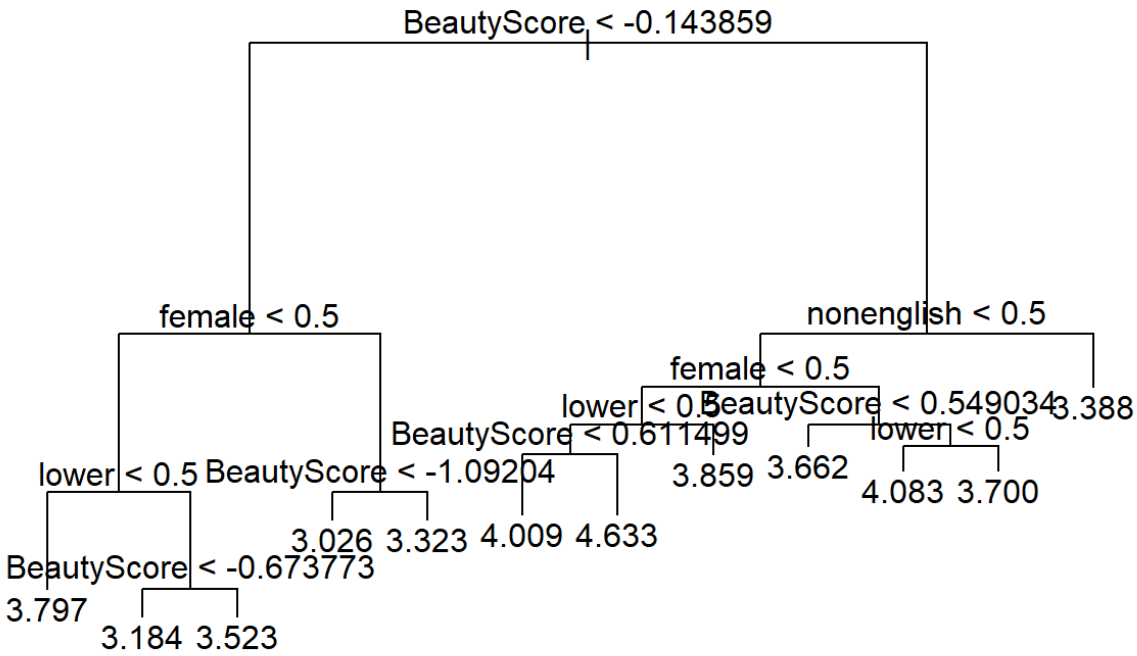
```r
summary(lm.evals)
```

```
##
## Call:
## lm(formula = CourseEvals ~ ., data = Btrain)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.32282 -0.31776  0.00488  0.29324  0.98308
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.00052    0.07325  54.617  < 2e-16 ***
## BeautyScore  0.30625    0.03718   8.236 1.45e-14 ***
## female      -0.32672    0.05823  -5.611 5.88e-08 ***
## lower       -0.34659    0.05925  -5.850 1.72e-08 ***
## nonenglish  -0.31638    0.12116  -2.611  0.00963 **
## tenuretrack -0.02676    0.06903  -0.388  0.69866
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4293 on 225 degrees of freedom
## Multiple R-squared:  0.3555, Adjusted R-squared:  0.3412
## F-statistic: 24.82 on 5 and 225 DF,  p-value: < 2.2e-16
```

```r
#reg tree
tree.evals=tree(CourseEvals~., beauty, subset =train)
summary(tree.evals)
```

```
##
## Regression tree:
## tree(formula = CourseEvals ~ ., data = beauty, subset = train)
## Variables actually used in tree construction:
## [1] "BeautyScore" "female"      "lower"       "nonenglish"
## Number of terminal nodes:  12
## Residual mean deviance:  0.1787 = 39.13 / 219
## Distribution of residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -1.1880 -0.3007 -0.0284  0.0000  0.2807  1.0600
```

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




```
predict.evals.tree = predict(tree.evals, Btest)
sqrt(mean((predict.evals.tree-Btest$CourseEvals)^2))
```

```
## [1] 0.4677175
```

```
#randomforest is worth trying too since the majority of variables are categorical a
nd for comparitive purposes
rf.evals=randomForest(CourseEvals~.,Btrain, mtry=2, ntree=500, importance =TRUE)
predict.evals.rf = predict(rf.evals, newdata=Btest)
sqrt(mean((predict.evals.rf-Btest$CourseEvals)^2))
```

```
## [1] 0.4446943
```

```
importance(rf.evals)
```

```
##                %IncMSE IncNodePurity
## BeautyScore 36.251375     17.012994
## female      24.610566      4.689427
## lower       28.464686      4.558349
## nonenglish  12.763441      1.461635
## tenuretrack  5.466474      1.345679
```

Using this data, we can conclude that the more beautiful an instructor is, the more likely it is that they will have higher course evaluations.

Multivariate linear regression produced a lower test RMSE than random forests did.

The coefficients p values in linear regression show that tenure track is the least significant variable, a conclusion shared by the importance function in random forests. Both also show that BeautyScore is the most significant contributor in the model, followed by lower, and female.

Other Variables that could influence CourseEvals: - Grades received in class. Typically people like the professors who teach the classes they did well in, and thus evaluate the class better. - Years of Experience. We would expect to see that more experienced instructors are better at teaching, and thus would score better in evaluations. Although tenure track is a related variable, a tenure track Assistant Professor who has been teaching for one year might not have as much teaching as a non-tenure track professor/lecturer who has been teaching for over a decade. - On a higher level, the way to truly determine the relationship between beauty and course evaluations, is to hold all the other factors constant. However, we do not even know what all the factors are, let alone have comprehensive data on them.

Exam Problem 1.2

Prof H is saying that's it is not possible to determine meaning/cause behind the correlation between beauty and higher course evaluations. Maybe pretty teachers are typically better instructors. Or maybe pretty teachers are no better at teaching than less pretty teachers, but they got higher evaluation scores because they are pretty. Or maybe it's a mix of two. The challenging hurdle of differentiating between these would require somehow controlling for all the other error related to this situation but that's impossible. Multiple people don't feel the same way and the same people can't take all the classes being considered; therefore, exactly replicating conditions between multiple people and multiple courses is impossible in reality. Additionally, all this extra error is hard to attribute, so it is in turn very hard to reduce.

Exam Probem 2.1

```
rm(list = ls())
setwd("~/R")
set.seed(1)
library(readr)
city <- read_csv("~/R/MidCity.csv")
```

```
## Parsed with column specification:
## cols(
##   Home = col_integer(),
##   Nbhd = col_integer(),
##   Offers = col_integer(),
##   SqFt = col_integer(),
##   Brick = col_character(),
##   Bedrooms = col_integer(),
##   Bathrooms = col_integer(),
##   Price = col_integer()
## )
```

```r
set.seed(1)
#settingt train and test
city$Brick=ifelse(city$Brick=="Yes", 1, 0)
city$Nbhd=as.factor(city$Nbhd)

train = sample(1:nrow(city), nrow(city)/2)
C.train = city[train,]
C.test = city[-train,]

#linear model
lm.price=lm(Price~.-Home,data=C.train)
lm.price.predict=predict(lm.price,C.test)
sqrt(mean((lm.price.predict - C.test$Price)^2))
```

```
## [1] 10511.67
```

```r
summary(lm.price)
```

```
##
## Call:
## lm(formula = Price ~ . - Home, data = C.train)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -28150  -3716  -1214   5635  26188
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2461.110  12368.038   0.199  0.84299
## Nbhd2       -1790.782   3605.136  -0.497  0.62132
## Nbhd3       19404.887   4339.514   4.472 3.84e-05 ***
## Offers      -7192.172   1474.338  -4.878 9.25e-06 ***
## SqFt           49.042      7.843   6.253 5.88e-08 ***
## Brick       17832.637   2931.653   6.083 1.11e-07 ***
## Bedrooms     6108.791   2242.421   2.724  0.00858 **
## Bathrooms    7379.366   2937.911   2.512  0.01492 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9801 on 56 degrees of freedom
## Multiple R-squared:  0.8615, Adjusted R-squared:  0.8441
## F-statistic: 49.74 on 7 and 56 DF,  p-value: < 2.2e-16
```

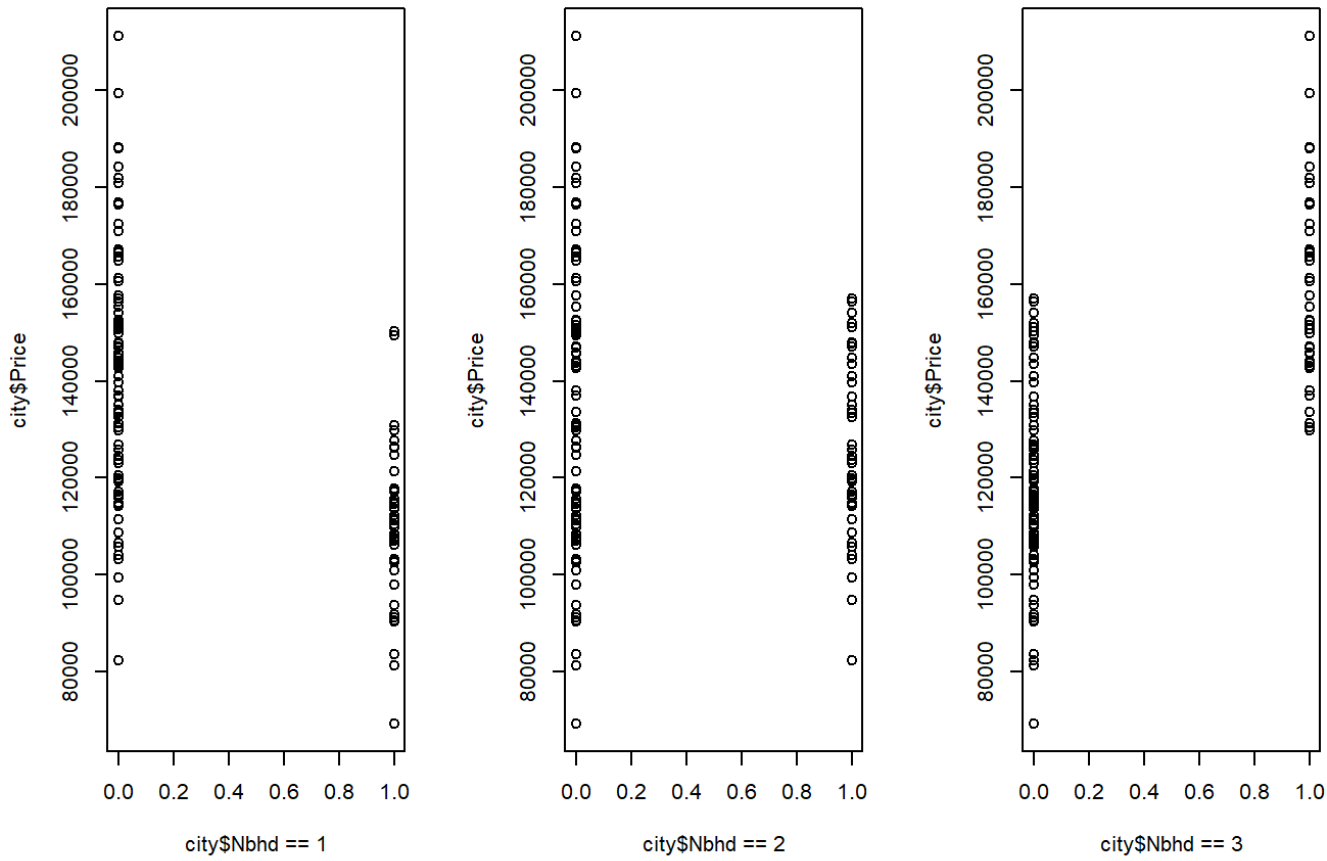
```r
confint(lm.price)
```

```
##                    2.5 %      97.5 %
## (Intercept) -22315.04684 27237.26605
## Nbhd2        -9012.73791  5431.17404
## Nbhd3        10711.79493 28097.97859
## Offers      -10145.62592 -4238.71765
## SqFt            33.33013    64.75339
## Brick        11959.83048 23705.44378
## Bedrooms      1616.68065 10600.90051
## Bathrooms     1494.02417 13264.70845
```

Running a model demonstrates that brick has a positive correlation with price, and brick is a good predictor of home price. This proves that there is a premium for brick houses, as they are more expensive. Specifically, if all else is held constant, changing a house to brick should increase the price of the house by $17,832.The confidence interval confirmed the premium.

Exam Probem 2.2

```
par(mfrow=c(1,3))
plot(city$Nbhd==1, city$Price)
plot(city$Nbhd==2, city$Price)
plot(city$Nbhd==3, city$Price)
```



There is a premium for houses in neighborhood 3, they are noticeably more expensive than houses in neighborhoods 1 and 2, as shown in the graphs and specified in the summary and confidence interval from part 1. If all else stays constant, a house being in the 3rd neighborhood should increase the price by $19,404.

Exam Probem 2.3

```
lm.price2 = lm(Price ~as.factor(Nbhd):Brick + Home + Nbhd + Offers + SqFt + Brick
+ Bedrooms + Bathrooms, data=C.train)

lm.price2.predict=predict(lm.price2,C.test)
sqrt(mean((lm.price2.predict - C.test$Price)^2))
```

```
## [1] 10723.88
```

```
summary(lm.price2)
```

```
##
## Call:
## lm(formula = Price ~ as.factor(Nbhd):Brick + Home + Nbhd + Offers +
##     SqFt + Brick + Bedrooms + Bathrooms, data = C.train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -25692.0  -4758.7   -428.7   4582.8  22848.6
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)                207.852  12333.185   0.017 0.986617
## Home                        22.248     37.016   0.601 0.550383
## Nbhd2                     1580.000   4145.602   0.381 0.704633
## Nbhd3                    16456.406   4852.062   3.392 0.001320 **
## Offers                   -7655.648   1524.772  -5.021 6.17e-06 ***
## SqFt                        50.954      8.525   5.977 1.97e-07 ***
## Brick                    18804.961   5240.457   3.588 0.000726 ***
## Bedrooms                  6863.723   2370.619   2.895 0.005491 **
## Bathrooms                 5772.427   3023.068   1.909 0.061622 .
## as.factor(Nbhd)2:Brick -8211.053   7168.406  -1.145 0.257169
## as.factor(Nbhd)3:Brick  6012.147   7407.958   0.812 0.420663
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9690 on 53 degrees of freedom
## Multiple R-squared:  0.8718, Adjusted R-squared:  0.8476
## F-statistic: 36.05 on 10 and 53 DF,  p-value: < 2.2e-16
```

```
confint(lm.price2)
```

```
##                                 2.5 %      97.5 %
## (Intercept)              -24529.38454 24945.08953
## Home                        -51.99706    96.49237
## Nbhd2                     -6735.02578  9895.02484
## Nbhd3                      6724.40119 26188.41039
## Offers                   -10713.95256 -4597.34266
## SqFt                        33.85559    68.05176
## Brick                      8293.93618 29315.98657
## Bedrooms                   2108.86366 11618.58183
## Bathrooms                  -291.08001 11835.93318
## as.factor(Nbhd)2:Brick -22589.05477  6166.94908
## as.factor(Nbhd)3:Brick  -8846.33528 20870.63017
```

By creating an interaction term between brick and neighborhood, we can show that there is an extra premium for brick houses in neighborhood 3. If all else is held constant, we a house in neighborhood 3 to increase in value by $6,012 if it goes from nonbrick to brick. However, the confidence interval includes 0, so a premium is not guaranteed.

Exam Probem 2.4

```
lmn1 = lm(Price~factor(Nbhd=="1")-Home, city)
summary(lmn1)
```

```
##
## Call:
## lm(formula = Price ~ factor(Nbhd == "1") - Home, data = city)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -58746 -16548   1195  11304  70154
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)                141046       2462  57.300  < 2e-16 ***
## factor(Nbhd == "1")TRUE    -30892       4198  -7.358 2.11e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 22560 on 126 degrees of freedom
## Multiple R-squared:  0.3005, Adjusted R-squared:  0.295
## F-statistic: 54.14 on 1 and 126 DF,  p-value: 2.109e-11
```

```
lmn2 = lm(Price~factor(Nbhd=="2")-Home, city)
summary(lmn2)
```

```
##
## Call:
## lm(formula = Price ~ factor(Nbhd == "2") - Home, data = city)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -64145 -19631  -2695  18216  77955
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)                133245       2930  45.468   <2e-16 ***
## factor(Nbhd == "2")TRUE     -8014       4942  -1.621    0.107
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 26700 on 126 degrees of freedom
## Multiple R-squared:  0.02044,    Adjusted R-squared:  0.01266
## F-statistic: 2.629 on 1 and 126 DF,  p-value: 0.1074
```

```
lmn3 = lm(Price~factor(Nbhd=="3")-Home, city)
summary(lmn3)
```

```
##
## Call:
## lm(formula = Price ~ factor(Nbhd == "3") - Home, data = city)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -48678 -12320  -1786  10368  51905
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)                117778       2002   58.83   <2e-16 ***
## factor(Nbhd == "3")TRUE     41517       3627   11.45   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 18890 on 126 degrees of freedom
## Multiple R-squared:  0.5098, Adjusted R-squared:  0.5059
## F-statistic:   131 on 1 and 126 DF,  p-value: < 2.2e-16
```

If a house is in neighborhood 1 or 3, we can make a prediction of price, because the p values are small enough to be significant. However, we cannot do this with neighborhood 2, which is why we consider merging neighborhoods 1 and 2.

```
levels(city$Nbhd) <- c("1", "1", "2")
lmnA = lm(Price ~ factor(Nbhd == "1")-Home, city)
summary(lmnA)
```

```
##
## Call:
## lm(formula = Price ~ factor(Nbhd == "1") - Home, data = city)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -48678 -12320  -1786  10368  51905
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)                159295       3024   52.67   <2e-16 ***
## factor(Nbhd == "1")TRUE    -41517       3627  -11.45   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 18890 on 126 degrees of freedom
## Multiple R-squared:  0.5098, Adjusted R-squared:  0.5059
## F-statistic:   131 on 1 and 126 DF,  p-value: < 2.2e-16
```

```
lmnB = lm(Price ~ factor(Nbhd == "2")-Home, city)
summary(lmnB)
```

```
##
## Call:
## lm(formula = Price ~ factor(Nbhd == "2") - Home, data = city)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -48678 -12320  -1786  10368  51905
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)                117778       2002   58.83   <2e-16 ***
## factor(Nbhd == "2")TRUE     41517       3627   11.45   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 18890 on 126 degrees of freedom
## Multiple R-squared:  0.5098, Adjusted R-squared:  0.5059
## F-statistic:   131 on 1 and 126 DF,  p-value: < 2.2e-16
```

Merging neighborhoods 1 and 2 makes prediction easier for the houses in neighborhood 2, and is therefore a good idea.

Exam Problem 3.1

There would be too many features to compare and it would be hard to reduce down to just the relationship between police and crime. Additionally, you wouldn't be able to tell what causes what in the scenario, whether increased crime causes the increase in police or if increased police causes a reduction of crime.

Exam Problem 3.2

The researchers were able to isolate crime and police presence by being strategic in their data collection and analysis. They found that the number of police on the street was not determined by the number of crimes that were occurring, but rather determined by the the threat level of DC on a given date. They also were able to control for the number of visitors to DC, represented in ridership of the Metro and found that ridership did not decrease when the terror threat level was orange. As a dummy variable, accounting for this was strategic and yielded better results.

Exam Problem 3.3

Checked hypothesis to see if high alert reduced the number of tourists. They were trying to control for the large number of people who visit DC, who likely use the metro to get around and are also more likely to be the targets of crime. High alert might sometimes correspond to when there are more tourists around, so there might be more police for events such as the inauguration, but that doesn't relate to this scenario.

If ridership has changed and caused crime to also go down, then it would have implied a causation relation between ridership and crime.

Exam Problem 3.4

The table shows the effect of a high alert day on different parts of DC by creating interaction terms. It shows that High Alert x District 1 is significant, but the others are less so. Ridership is significant as well, but less so than High Alert x District 1.

Exam Probem 4

For the R Cars project, my main contribution was that I wrote the write up for my group. I wrote it on a Google doc, so other people could give me feedback and edit if they wanted, especially with regards to the models that I did not run. I formatted it in such a way that presented both the high level and the deep dive into the models without copying the code itself and taking up too much space. As for the models themselves, I worked on Lasso and Ridge models at the same time as Steve so we could double check our answers. Our initial divide was that I would do Ridge and he would do Lasso, but it made more sense for us to both try both and compare results. We both used the same seed and train/test split, so we were able to check each others work.

---

output: ms_document