Exam_v2

Zack Bilderback

July 26, 2016

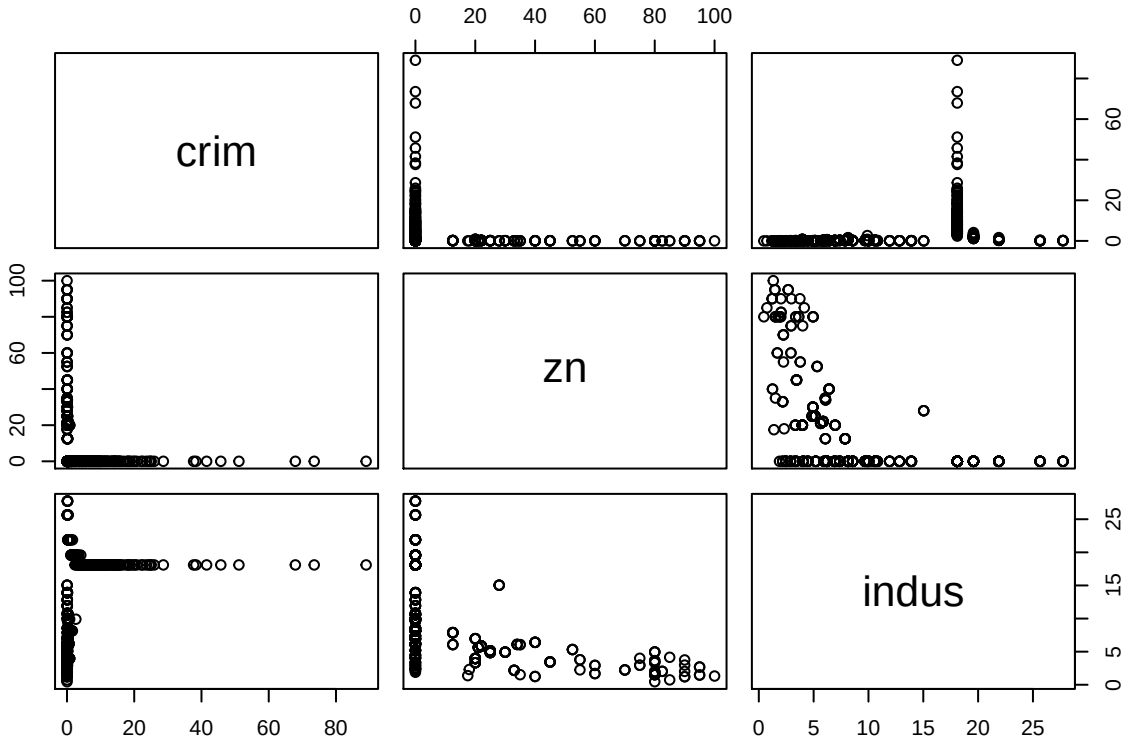
Book Problems

Chapter 2: #10

(a) The Boston data set has 506 rows and 14 columns. Each row represents a different town/suburb in Boston. The columns represent per capita crime rate by town, proportion of residential land zoned for lots over 25,000 sq.ft., proportion of non-retail business acres per town, Charles River dummy variable (= 1 if tract bounds river; 0 otherwise), nitrogen oxides concentration (parts per 10 million), average number of rooms per dwelling, proportion of owner-occupied units built prior to 1940, weighted mean of distances to five Boston employment centers, index of accessibility to radial highways, full-value property-tax rate per $10,000, pupil-teacher ratio by town, 1000(Bk - 0.63)ˆ2 where Bk is the proportion of blacks by town, lower status of the population (percent), and median value of owner-occupied homes in $1000s.

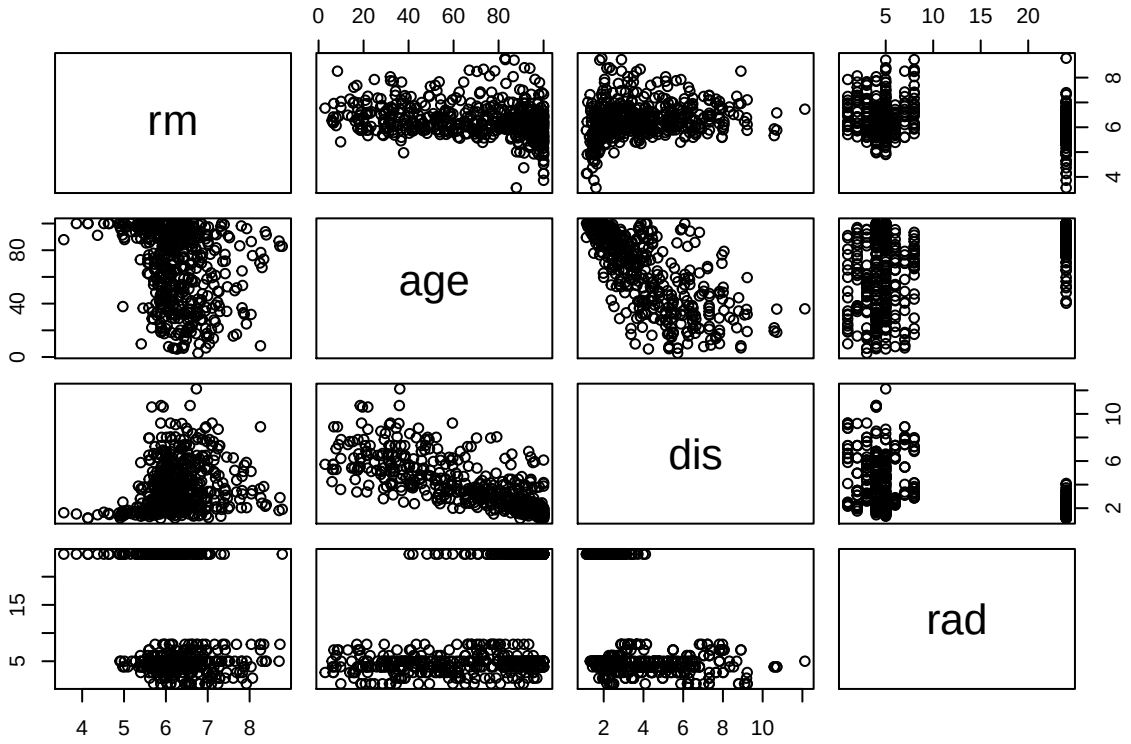
(b) This chart shows that the per captia crime rate by town is lower in proportions of residential land zoned for lots over 25,000 sq.ft We can see this in the chart by looking where zn is 0 and they crime rate is very high. This means many crimes happen in smaller residential zones. We can also observe that the crime rate jumps up when the proportion of non-retial business acres per town reaches about 15.

```
library(MASS)
attach(Boston)
plot(Boston[,1:3])
```



In this scatter plot, we can see that many of homes built prior to 1940 have about 6 or 7 rooms. Also, many of these homes are close to Boston employment centers with easy access to highways.

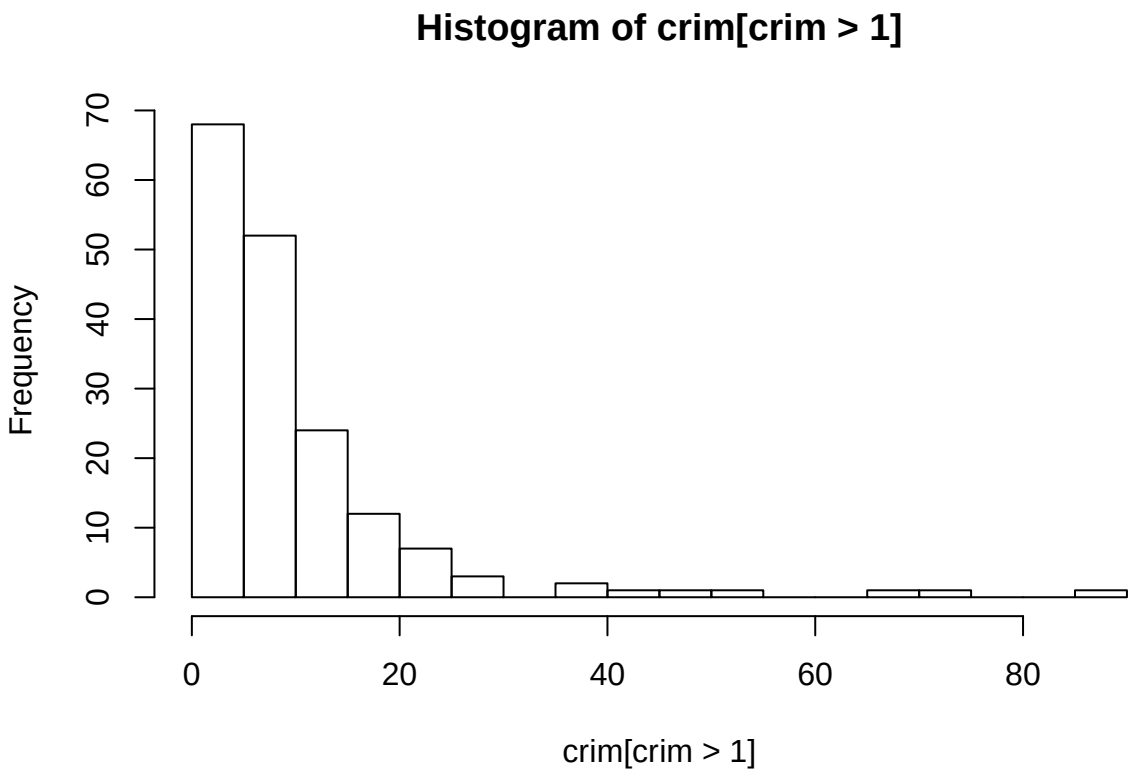
```
plot(Boston[,6:9])
```



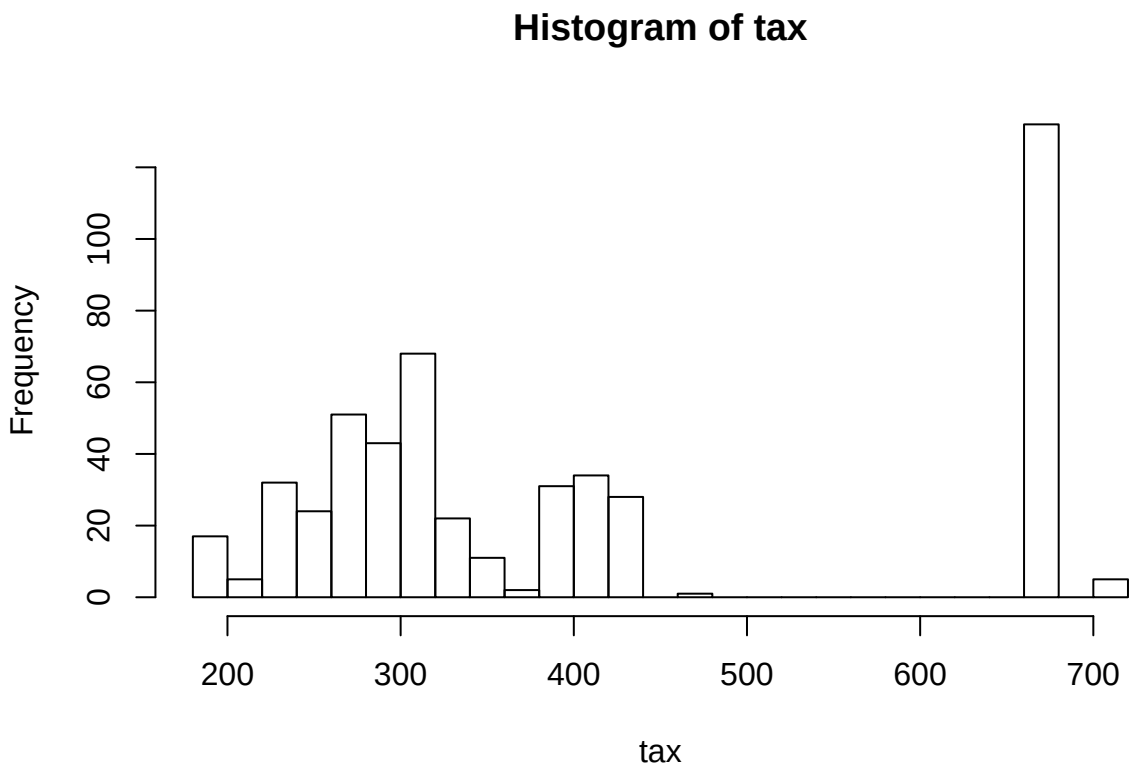
(c) There seems to be a slight relationship between the crime rate and median value of owner-occupied homes in $1000s. When there is a lower the median value of the home, the crime rate increases slightly. Also, the crime rate is higher in residential zones that have lots smaller than 25,000 sq.ft which makes sense because the lower median value of homes also means they are smaller homes. The lower median value of homes also relates to the home being older so crime rate is increased in older homes.

(d) The majority of suburbs have low crime rates but there are about 20 that have higher crime rates. There is a big gap between suburbs with a high tax rate, peaking at about 650, and the suburbs with lower tax rates. Many suburbs have a high ratio of pupil to teacher. A ratio of about 20 has a relatively large frequency compared to other suburbs.

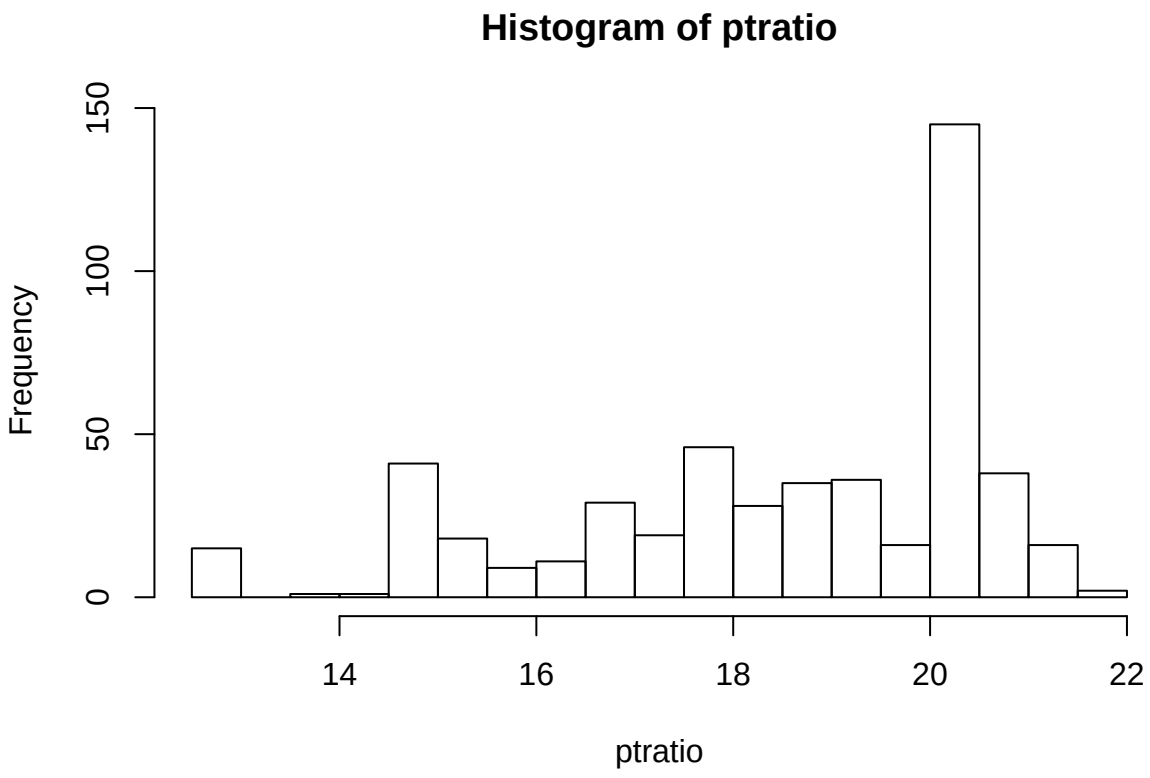
```
hist(crim[crim > 1], breaks = 25)
```




```
hist(tax, breaks=25)
```




```
hist(ptratio, breaks=25)
```



(e)

```
sum(chas == 1)
```

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

(f)

```
median(ptratio)
```

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

(g) These two suburbs are very similar and they are not the best suburbs to live in compared to the average but they are also not the worst.

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

(h)

```
sum(rm>7) ##Suburbs averaging more than 7 rooms
```

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

```
sum(rm > 8) ##Suburbs averaging more than 8 rooms
```

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

```
summary(subset(Boston, rm > 8)) ##The max crime rate and lstat is significantly lower in the suburbs wi
```

```
##       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
```

Chapter 3: #15

(a) All predictors have a p-value less than 0.05 except for the *chas* predictor, so every predictor is statistically significant to the response except for the *chas* predictor. Below are the results from fitting a simple linear regression with each predictor.

```
fit.zn <- lm(crim ~ zn)
summary(fit.zn)
```

```
##
## Call:
## lm(formula = crim ~ zn)
##
## 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
```

```
fit.indus <- lm(crim ~ indus)
summary(fit.indus)
```

```
##
## Call:
## lm(formula = crim ~ indus)
##
## 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
```

```
chas <- as.factor(chas)
fit.chas <- lm(crim ~ chas)
summary(fit.chas)
```

```
##
## Call:
## lm(formula = crim ~ chas)
##
## 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 ***
## chas1        -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
```

```
fit.nox <- lm(crim ~ nox)
summary(fit.nox)
```

```
##
## Call:
## lm(formula = crim ~ nox)
##
## 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
```

```
fit.rm <- lm(crim ~ rm)
summary(fit.rm)
```

```
##
## Call:
## lm(formula = crim ~ rm)
##
## 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
```

```
fit.age <- lm(crim ~ age)
summary(fit.age)
```

```
##
## Call:
## lm(formula = crim ~ age)
##
## 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
```

```
fit.dis <- lm(crim ~ dis)
summary(fit.dis)
```

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

```r
fit.rad <- lm(crim ~ rad)
summary(fit.rad)
```

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

```r
fit.tax <- lm(crim ~ tax)
summary(fit.tax)
```

```
##
## Call:
## lm(formula = crim ~ tax)
##
## 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
```

```r
fit.ptratio <- lm(crim ~ ptratio)
summary(fit.ptratio)
```

```
##
## Call:
## lm(formula = crim ~ ptratio)
##
## 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
```

```
fit.black <- lm(crim ~ black)
summary(fit.black)
```

```
##
## Call:
## lm(formula = crim ~ black)
##
## 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
```

```
fit.lstat <- lm(crim ~ lstat)
summary(fit.lstat)
```

```
##
## Call:
## lm(formula = crim ~ lstat)
##
## 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
```

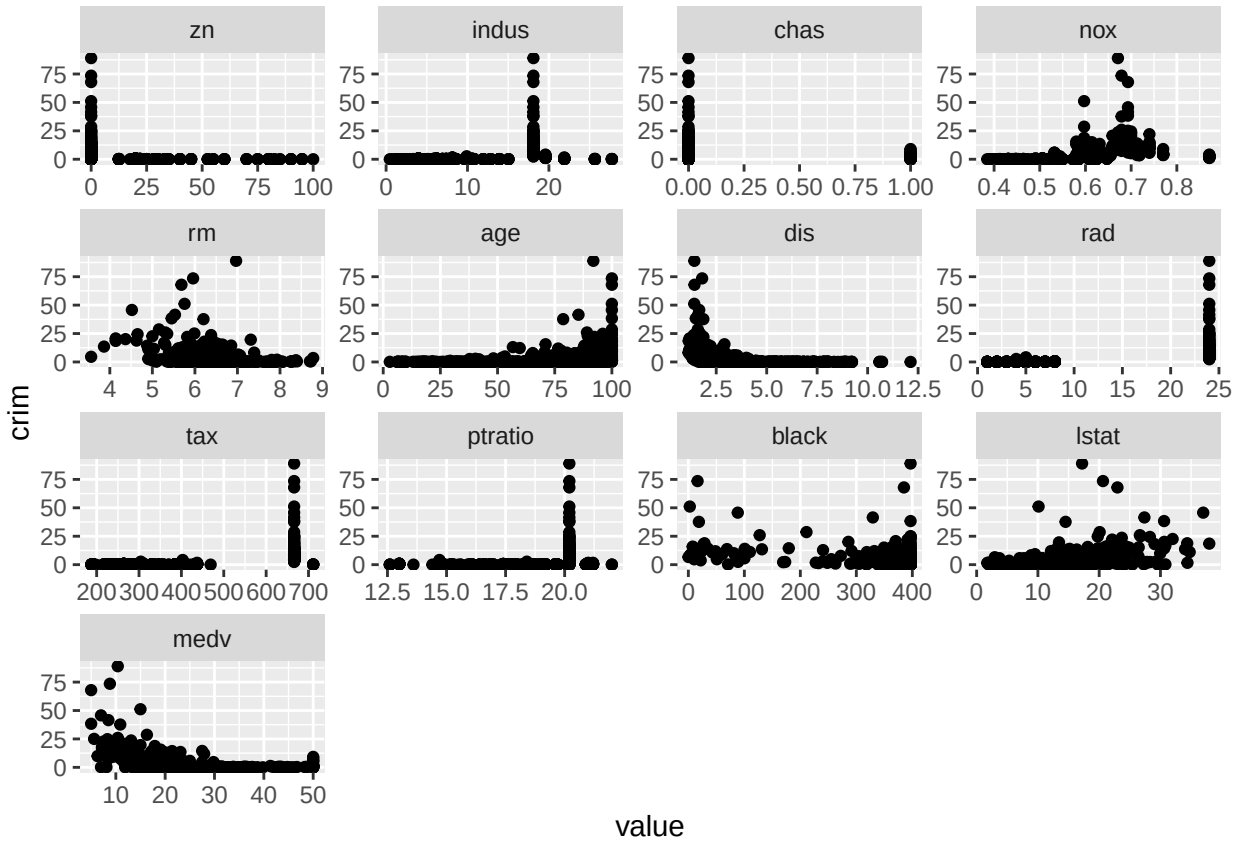
```
##
## Residual standard error: 7.664 on 504 degrees of freedom
## Multiple R-squared:  0.2076, Adjusted R-squared:  0.206
## F-statistic:   132 on 1 and 504 DF,  p-value: < 2.2e-16
```

```
fit.medv <- lm(crim ~ medv)
summary(fit.medv)
```

```
##
## Call:
## lm(formula = crim ~ medv)
##
## 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
```

The following plots have the per capita crime rate as the response variable compared to each predictor.

```
library(ggplot2)
library(reshape2)
mtmelt = melt(Boston, id = "crim")
ggplot(mtmelt, aes(x = value, y = crim)) +
facet_wrap(~variable, scales = "free") +
geom_point()
```



(b) We can reject the null hypothesis for *zn*, *dis*, *rad*, *black*, and *medv*

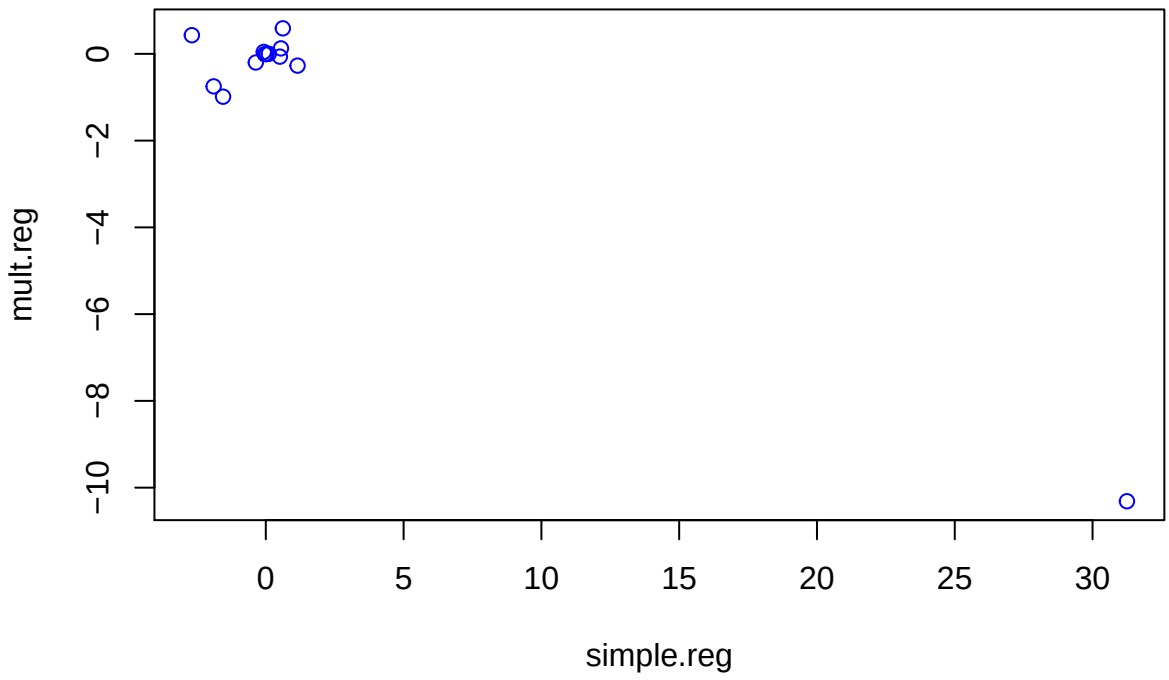
```
fit.all <- lm(crim ~ ., data = Boston)
summary(fit.all)
```

```
##
## 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
```

(c) When comparing results from parts a and b, we can see a difference in the coefficients. In the simple regression models, the slope is a result of the average effect of an increase in the predictor while ignoring all other predictors. However, the multiple regression model is the average effect of an increase in the predictor while holding all other predictors constant. It makes sense that the multiple regression model might suggest a relationship between predictors while the simple model suggests the opposite because the correlation between predictors shows some stronger relationships.

```
simple.reg <- vector("numeric",0)
simple.reg <- c(simple.reg, fit.zn$coefficient[2])
simple.reg <- c(simple.reg, fit.indus$coefficient[2])
simple.reg <- c(simple.reg, fit.chas$coefficient[2])
simple.reg <- c(simple.reg, fit.nox$coefficient[2])
simple.reg <- c(simple.reg, fit.rm$coefficient[2])
simple.reg <- c(simple.reg, fit.age$coefficient[2])
simple.reg <- c(simple.reg, fit.dis$coefficient[2])
simple.reg <- c(simple.reg, fit.rad$coefficient[2])
simple.reg <- c(simple.reg, fit.tax$coefficient[2])
simple.reg <- c(simple.reg, fit.ptratio$coefficient[2])
simple.reg <- c(simple.reg, fit.black$coefficient[2])
simple.reg <- c(simple.reg, fit.lstat$coefficient[2])
simple.reg <- c(simple.reg, fit.medv$coefficient[2])
mult.reg <- vector("numeric", 0)
mult.reg <- c(mult.reg, fit.all$coefficients)
mult.reg <- mult.reg[-1]
plot(simple.reg, mult.reg, col = "blue")
```



(d) When *zn*, *rm*, *rad*, *tax* and *lstat* are predictors, the p-values suggest that the cubic coefficient is not statistically significant. When *indus*, *nox*, *age*, *dis*, *ptratio* and *medv* are predictors, the p-values suggest the cubic fit is acceptable; When *black* is the predictor, the p-values suggest that the quadratic and cubic coefficients are not statistically significant, so no non-linear effect is visible.

```
fit.zn2 <- lm(crim ~ poly(zn, 3))
summary(fit.zn2)
```

```
##
## Call:
## lm(formula = crim ~ poly(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(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
```

```
fit.indus2 <- lm(crim ~ poly(indus, 3))
summary(fit.indus2)
```

```
##
## Call:
## lm(formula = crim ~ poly(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(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
```

```
fit.nox2 <- lm(crim ~ poly(nox, 3))
summary(fit.nox2)
```

```
##
## Call:
## lm(formula = crim ~ poly(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(nox, 3)1  81.3720     7.2336  11.249  < 2e-16 ***
## poly(nox, 3)2 -28.8286     7.2336  -3.985 7.74e-05 ***
## poly(nox, 3)3 -60.3619     7.2336  -8.345 6.96e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.234 on 502 degrees of freedom
## Multiple R-squared:  0.297,  Adjusted R-squared:  0.2928
## F-statistic: 70.69 on 3 and 502 DF,  p-value: < 2.2e-16
```

```r
fit.rm2 <- lm(crim ~ poly(rm, 3))
summary(fit.rm2)
```

```
##
## Call:
## lm(formula = crim ~ poly(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(rm, 3)1 -42.3794     8.3297  -5.088 5.13e-07 ***
## poly(rm, 3)2  26.5768     8.3297   3.191  0.00151 **
## poly(rm, 3)3  -5.5103     8.3297  -0.662  0.50858
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.33 on 502 degrees of freedom
## Multiple R-squared:  0.06779,    Adjusted R-squared:  0.06222
## F-statistic: 12.17 on 3 and 502 DF,  p-value: 1.067e-07
```

```r
fit.age2 <- lm(crim ~ poly(age, 3))
summary(fit.age2)
```

```
##
## Call:
## lm(formula = crim ~ poly(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(age, 3)1  68.1820     7.8397   8.697  < 2e-16 ***
## poly(age, 3)2  37.4845     7.8397   4.781 2.29e-06 ***
## poly(age, 3)3  21.3532     7.8397   2.724  0.00668 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.84 on 502 degrees of freedom
## Multiple R-squared:  0.1742, Adjusted R-squared:  0.1693
## F-statistic: 35.31 on 3 and 502 DF,  p-value: < 2.2e-16
```

```r
fit.dis2 <- lm(crim ~ poly(dis, 3))
summary(fit.dis2)
```

```
##
```

```
## Call:
## lm(formula = crim ~ poly(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(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
```

```
fit.rad2 <- lm(crim ~ poly(rad, 3))
summary(fit.rad2)
```

```
##
## Call:
## lm(formula = crim ~ poly(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(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
```

```
fit.tax2 <- lm(crim ~ poly(tax, 3))
summary(fit.tax2)
```

```
##
## Call:
## lm(formula = crim ~ poly(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(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
```

```
fit.ptratio2 <- lm(crim ~ poly(ptratio, 3))
summary(fit.ptratio2)
```

```
##
## Call:
## lm(formula = crim ~ poly(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(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
```

```
fit.black2 <- lm(crim ~ poly(black, 3))
summary(fit.black2)
```

```
##
## Call:
## lm(formula = crim ~ poly(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(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
```

```
fit.lstat2 <- lm(crim ~ poly(lstat, 3))
summary(fit.lstat2)
```

```
##
## Call:
## lm(formula = crim ~ poly(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(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
```

```
fit.medv2 <- lm(crim ~ poly(medv, 3))
summary(fit.medv2)
```

```
##
## Call:
## lm(formula = crim ~ poly(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(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
```

Chapter 6: #9

```
library(ISLR)
data(College)
```

(a)

```
set.seed(1)
train = sample(c(TRUE,FALSE),nrow(College),rep=TRUE)
test = (!train)

College.train = College[train,,drop=F]
College.test = College[test,,drop=F]
```

(b)

```
lm.fit = lm(Apps~.,data=College.train)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = Apps ~ ., data = College.train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2485.0  -392.4   -57.9   261.6  6637.3
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -756.88732  535.44217  -1.414 0.158301
## PrivateYes  -515.36459  190.44202  -2.706 0.007112 **
## Accept         1.20916    0.07046  17.161  < 2e-16 ***
## Enroll        -0.29026    0.23380  -1.241 0.215188
## Top10perc     52.53768    7.69358   6.829 3.38e-11 ***
## Top25perc    -16.38973    6.09706  -2.688 0.007500 **
## F.Undergrad    0.09625    0.04103   2.346 0.019495 *
## P.Undergrad    0.02831    0.05417   0.523 0.601540
## Outstate      -0.02161    0.02599  -0.831 0.406262
## Room.Board     0.13204    0.06968   1.895 0.058847 .
## Books          0.16642    0.34092   0.488 0.625726
## Personal       0.16409    0.09270   1.770 0.077519 .
## PhD           -8.51810    6.73921  -1.264 0.207015
## Terminal      -0.35379    7.43056  -0.048 0.962049
## S.F.Ratio      3.79436   16.20785   0.234 0.815028
## perc.alumni   -2.86286    5.56666  -0.514 0.607349
```

```
## Expend         0.05855    0.01528   3.832 0.000148 ***
## Grad.Rate      7.14929    3.91090   1.828 0.068323 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1001 on 382 degrees of freedom
## Multiple R-squared:  0.9314, Adjusted R-squared:  0.9283
## F-statistic: 304.9 on 17 and 382 DF,  p-value: < 2.2e-16
```

```
pred = predict(lm.fit, College.test)
ssr = sum((pred-College.test$Apps)^2)
sst = sum((College.test$Apps-mean(College.test$Apps))^2)
test.rsq = 1 - (ssr/sst)
test.rsq
```

```
## [1] 0.9051744
```

(c)

```
library(glmnet)
College.train.X=scale(model.matrix(Apps~.,data=College.train)[,-1],scale=T,center=T)
College.train.Y=College.train$Apps

College.test.X=scale(model.matrix(Apps~.,data=College.test)[,-1],
      attr(College.train.X,"scaled:center"),
      attr(College.train.X,"scaled:scale"))

College.test.Y=College.test$Apps

cv.out=cv.glmnet(College.train.X,College.train.Y,alpha=0)
bestlam=cv.out$lambda.min
bestlam
```

```
## [1] 384.4944
```

```
lasso.mod = glmnet(College.train.X, College.train.Y, alpha = 0, lambda = bestlam)
pred = predict(lasso.mod, College.test.X, s=bestlam)
ssr = sum((pred-College.test$Apps)^2)
sst = sum((College.test$Apps-mean(College.test$Apps))^2)
test.rsq = 1 - (ssr/sst)
test.rsq
```

```
## [1] 0.8386754
```

(d)

```
cv.out=cv.glmnet(College.train.X,College.train.Y,alpha=1)
bestlam=cv.out$lambda.min
bestlam
```

```
## [1] 14.47597
```

```
lasso.mod=glmnet(College.train.X,College.train.Y,alpha=1,lambda=bestlam)
pred=predict(lasso.mod,College.test.X,s=bestlam)
ssr=sum((pred-College.test$Apps)^2)
sst=sum((College.test$Apps-mean(College.test$Apps))^2)
test.rsq=1-(ssr/sst)
test.rsq
```

```
## [1] 0.8994791
```

```
#Number of coefficients not equal to 0
sum(coef(lasso.mod)[,1]!=0)
```

```
## [1] 15
```

(e)

```
library(pls)
pcr.fit=pcr(Apps~.,data=College.train, scale=TRUE, validation="CV")
summary(pcr.fit)
```

```
## Data:    X dimension: 400 17
##  Y dimension: 400 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            3744     3547     1681     1696     1508     1308     1259
## adjCV         3744     3550     1679     1700     1462     1272     1255
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV        1257     1257     1177      1182      1185      1188      1185
## adjCV     1254     1255     1172      1179      1182      1185      1182
##        14 comps  15 comps  16 comps  17 comps
## CV         1189      1194      1081      1059
## adjCV      1186      1191      1076      1055
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X       32.07    58.34    64.91    70.53    75.95    81.04    84.82
## Apps    11.14    80.28    80.29    86.40    89.31    89.41    89.47
##       8 comps  9 comps  10 comps  11 comps  12 comps  13 comps  14 comps
## X       88.19    91.14     93.51     95.61     97.27     98.31     99.01
## Apps    89.57    90.68     90.77     90.78     90.82     90.88     90.89
##       15 comps  16 comps  17 comps
## X        99.48     99.81    100.00
## Apps     90.93     92.63     93.14
```

```
pred=predict(pcr.fit,College.test,ncomp=17)
ssr=sum((pred-College.test$Apps)^2)
sst=sum((College.test$Apps-mean(College.test$Apps))^2)
test.rsq=1-(ssr/sst)
test.rsq
```

```
## [1] 0.9051744
```

(f)

```
pls.fit=plsr(Apps~.,data=College.train, scale=TRUE, validation="CV")
summary(pls.fit)
```

```
## Data:    X dimension: 400 17
##  Y dimension: 400 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            3744     1551     1360     1172     1142     1093     1082
## adjCV         3744     1549     1360     1168     1137     1077     1075
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV        1076     1073     1069      1069      1067      1068      1067
## adjCV     1070     1067     1064      1064      1063      1063      1062
##        14 comps  15 comps  16 comps  17 comps
## CV         1067      1067      1067      1067
## adjCV      1062      1062      1062      1062
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X       27.28    50.45    63.72    66.60    68.38    72.56    75.63
## Apps    83.57    87.63    90.98    91.71    92.72    92.99    93.06
##       8 comps  9 comps  10 comps  11 comps  12 comps  13 comps  14 comps
## X       79.99    83.33     85.84     88.70     91.51     93.71     96.30
## Apps    93.09    93.11     93.12     93.13     93.13     93.14     93.14
##       15 comps  16 comps  17 comps
## X        97.40     99.20    100.00
## Apps     93.14     93.14     93.14
```

```
pred=predict(pls.fit,College.test,ncomp=9)
ssr=sum((pred-College.test$Apps)^2)
sst=sum((College.test$Apps-mean(College.test$Apps))^2)
test.rsq=1-(ssr/sst)
test.rsq
```

```
## [1] 0.9034921
```

(g) They all performed simliarily and reported an R^2 of around .9 but ridge regression reported a value of about .83 which was slightly below. Most of the variables contributed information to the model which I believe made all the models equal in their R^2 values.

Chapter 6: #11

(a)

```
library(ISLR)
library(leaps)
library(MASS)

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
set.seed(1)
p=ncol(Boston)-1
folds=sample(1:k,nrow(Boston),replace=TRUE)

cv.errors=c()
for(j in 1:k){
  Boston.sub=Boston[folds!=j,]
  #now do CV on this CV subset to choose the best model, and apply
  # it to the whole thing.
  cv.err=matrix(NA,k,p,dimnames=list(NULL,paste(1:p)))
  folds.sub=sample(1:k,nrow(Boston.sub),replace=TRUE)

  for(q in 1:k){
    best.fit=regsubsets(crim~.,data=Boston.sub[folds.sub!=q,],nvmax=p)
    for(i in 1:p){
      pred=predict.regsubsets(best.fit,Boston.sub[folds.sub==q,],id=i)
      cv.err[q,i]=mean((Boston.sub$crim[folds.sub==q]-pred)^2)
    }
  }

  best.k = which.min(apply(cv.err,2,mean))

  best.fit.all=regsubsets(crim~.,data=Boston.sub,nvmax=p)
  pred=predict.regsubsets(best.fit.all,Boston[folds==j,],id=best.k)

  cv.errors=c(cv.errors,mean((Boston$crim[folds==j]-pred)^2))
}

mean(cv.errors)
```

```
## [1] 42.18284
```

```
## Ridge regression (alpha=0)
library(glmnet)

cv.errors = sapply(1:k, function(j){
  Boston.X=as.matrix(Boston[,-1])
  Boston.Y=Boston[,1]

  cv.out=cv.glmnet(Boston.X[folds!=j,],Boston.Y[folds!=j],alpha=0)
```

```
  bestlam=cv.out$lambda.min
  bestlam

  lasso.mod=glmnet(Boston.X[folds!=j,],Boston.Y[folds!=j],alpha=0,lambda=bestlam)
  pred=predict(lasso.mod,Boston.X[folds==j,],s=bestlam)
  return(mean((Boston.Y[folds==j]-pred)^2))
})

mean(cv.errors)
```

```
## [1] 41.1623
```

```
## Lasso (alpha=1)
cv.errors = sapply(1:k, function(j){
  Boston.X=as.matrix(Boston[,-1])
  Boston.Y=Boston[,1]

  cv.out=cv.glmnet(Boston.X[folds!=j,],Boston.Y[folds!=j],alpha=1)
  bestlam=cv.out$lambda.min
  bestlam

  lasso.mod=glmnet(Boston.X[folds!=j,],Boston.Y[folds!=j],alpha=1,lambda=bestlam)
  pred=predict(lasso.mod,Boston.X[folds==j,],s=bestlam)
  return(mean((Boston.Y[folds==j]-pred)^2))
})

mean(cv.errors)
```

```
## [1] 41.27
```

```
## PCR
library (pls)

cv.errors = sapply(1:k, function(j){

  pcr.fit=pcr(crim~.,data=Boston[folds!=j,],scale=TRUE,validation="CV")
  res=RMSEP(pcr.fit)
  pcr.best=which.min(res$val[1,,])-1

  pred=predict(pcr.fit,Boston[folds==j,],ncomp=pcr.best)
  return(mean((Boston[folds==j,1]-pred)^2))
})

mean(cv.errors)
```

```
## [1] 41.06013
```

You can see that PCR is the best when comparing the mean of cv.errors.

(b) Ridge regression and PCR were the best. Lasso was almost as good, and best subset selection was a little bit worse compared to the rest. RR uses some information from each feature, even though it might discredit the effect from some of the features.

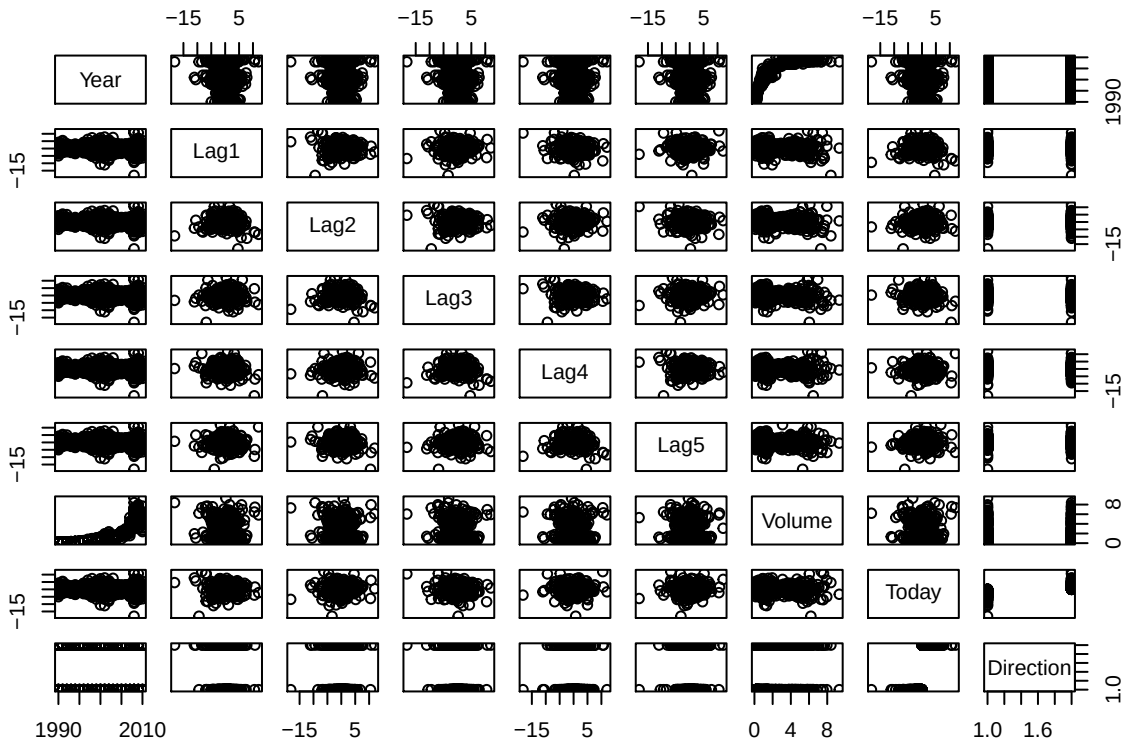
(c) PCR regression includes all features. The pcr function selects a subset of linear combinations of all the features. Some variance in a few of the features is not included, but some information from each feature will be in the final model regardless of the parameter that was selected in each CV iteration.

CH.4 #10

(a) There is high correlation between Volume and Year. It looks like an exponential relationship where volume increases exponentially as a function of year. Certain years seem to have more or less variation than other years. There is little to no correlation between any other variables.

```
library(MASS)
library(ISLR)

pairs(Weekly)
```




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

```
##              Year         Lag1        Lag2        Lag3         Lag4
## Year    1.00000000 -0.032289274 -0.03339001 -0.03000649 -0.031127923
## Lag1   -0.03228927  1.000000000 -0.07485305  0.05863568 -0.071273876
## Lag2   -0.03339001 -0.074853051  1.00000000 -0.07572091  0.058381535
## Lag3   -0.03000649  0.058635682 -0.07572091  1.00000000 -0.075395865
## Lag4   -0.03112792 -0.071273876  0.05838153 -0.07539587  1.000000000
## Lag5   -0.03051910 -0.008183096 -0.07249948  0.06065717 -0.075675027
```

```
## Volume  0.84194162 -0.064951313 -0.08551314 -0.06928771 -0.061074617
## Today  -0.03245989 -0.075031842  0.05916672 -0.07124364 -0.007825873
##                Lag5      Volume        Today
## Year   -0.030519101  0.84194162 -0.032459894
## Lag1   -0.008183096 -0.06495131 -0.075031842
## Lag2   -0.072499482 -0.08551314  0.059166717
## Lag3    0.060657175 -0.06928771 -0.071243639
## Lag4   -0.075675027 -0.06107462 -0.007825873
## Lag5    1.000000000 -0.05851741  0.011012698
## Volume -0.058517414  1.00000000 -0.033077783
## Today   0.011012698 -0.03307778  1.000000000
```

(b) The intercept and Lag2 seem to be important.

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

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

```
summary(logit.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
```

(c) From the confusion matrix, we can conclude that the up direction is guessed a significant amount more than down. Many mistakes are occurring from guessing that the market will go up when it really goes down instead. We can see that the model and data do not fit well.

```
glm.probs=predict(logit.fit,Weekly,type="response")
glm.pred=rep("Down",nrow(Weekly))
glm.pred[glm.probs > 0.50]="Up"
table(glm.pred,Weekly$Direction)
```

```
##
## glm.pred Down  Up
##     Down   54  48
##     Up    430 557
```

```
mean(glm.pred==Weekly$Direction)
```

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

(d)

```
train=Weekly$Year <= 2008
Weekly.test=Weekly[!train,]
logit.fit = glm(Direction ~ Lag2, family=binomial, data=Weekly, subset=train)
contrasts(Weekly$Direction)
```

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

```
summary(logit.fit)
```

```
##
## Call:
## glm(formula = Direction ~ Lag2, family = binomial, data = Weekly,
##     subset = train)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.536  -1.264   1.021   1.091   1.368
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.20326    0.06428   3.162  0.00157 **
## Lag2         0.05810    0.02870   2.024  0.04298 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1354.7  on 984  degrees of freedom
## Residual deviance: 1350.5  on 983  degrees of freedom
```

```
## AIC: 1354.5
##
## Number of Fisher Scoring iterations: 4
```

```
glm.probs=predict(logit.fit,Weekly.test,type="response")
glm.pred=rep("Down",nrow(Weekly.test))
glm.pred[glm.probs > 0.50]="Up"
table(glm.pred,Weekly.test$Direction)
```

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

```
mean(glm.pred==Weekly.test$Direction)
```

```
## [1] 0.625
```

(g)

```
library(class)
train.X=Weekly[train,"Lag2",drop=F]
test.X=Weekly[!train,"Lag2",drop=F]
train.Direction=Weekly[train,"Direction",drop=T]
test.Direction=Weekly[!train,"Direction",drop=T]
set.seed(1)
knn.pred=knn(train.X,test.X,train.Direction,k=1)
table(knn.pred,test.Direction)
```

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

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

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

(h) KKN appears to fit better than logistic regression since LR kept picking up most of the time. KKN appears to slightly more random in its choices.

(i) It appears that logistic regression does worse with more variables so I reduced it down to 3 variables instead. Lag2 had a good correlation from earlier which is why I included it here.

```
train=Weekly$Year <= 2008
Weekly.test=Weekly[!train,]
logit.fit = glm(Direction ~ Lag1+Lag2+Volume, family=binomial, data=Weekly, subset=train)
contrasts(Weekly$Direction)
```

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

```
summary(logit.fit)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2 + Volume, family = binomial,
##     data = Weekly, subset = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4681  -1.2581   0.9929   1.0840   1.5339
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.29792    0.09136   3.261  0.00111 **
## Lag1        -0.05975    0.02917  -2.048  0.04054 *
## Lag2         0.04774    0.02941   1.624  0.10446
## Volume      -0.07093    0.05263  -1.348  0.17777
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1354.7  on 984  degrees of freedom
## Residual deviance: 1345.1  on 981  degrees of freedom
## AIC: 1353.1
##
## Number of Fisher Scoring iterations: 4
```

```
glm.probs=predict(logit.fit,Weekly.test,type="response")
glm.pred=rep("Down",nrow(Weekly.test))
glm.pred[glm.probs > 0.50]="Up"
table(glm.pred,Weekly.test$Direction)
```

```
##
## glm.pred Down Up
##     Down   27 33
##     Up     16 28
```

```
mean(glm.pred==Weekly.test$Direction)
```

```
## [1] 0.5288462
```

KKN with k=3 has a mean of .54 which is pretty good.

```
set.seed(1)
knn.pred=knn(train.X,test.X,train.Direction,k=3)
table(knn.pred,test.Direction)
```

```
##         test.Direction
## knn.pred Down Up
##     Down   16 20
##     Up     27 41
```

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

```
## [1] 0.5480769
```

CH.8 #8

(a)

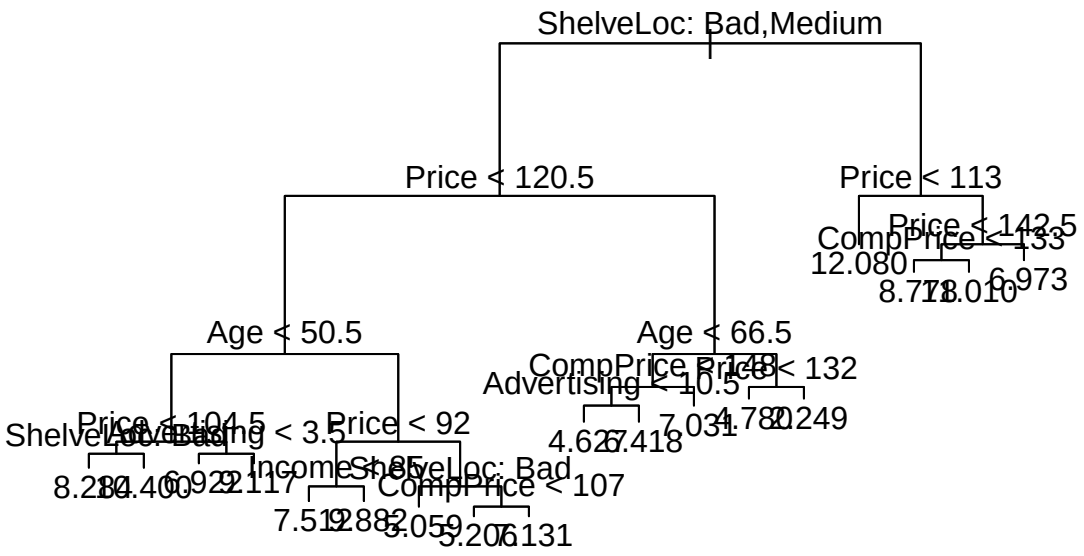
```
library(ISLR)
set.seed(1)
train <- sample(1:nrow(Carseats), nrow(Carseats) / 2)
Carseats.train <- Carseats[train, ]
Carseats.test <- Carseats[-train, ]
```

(b)

```
library(tree)
tree.carseats <- tree(Sales ~ ., data = Carseats.train)
summary(tree.carseats)
```

```
##
## Regression tree:
## tree(formula = Sales ~ ., data = Carseats.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.carseats)
text(tree.carseats, pretty = 0)
```

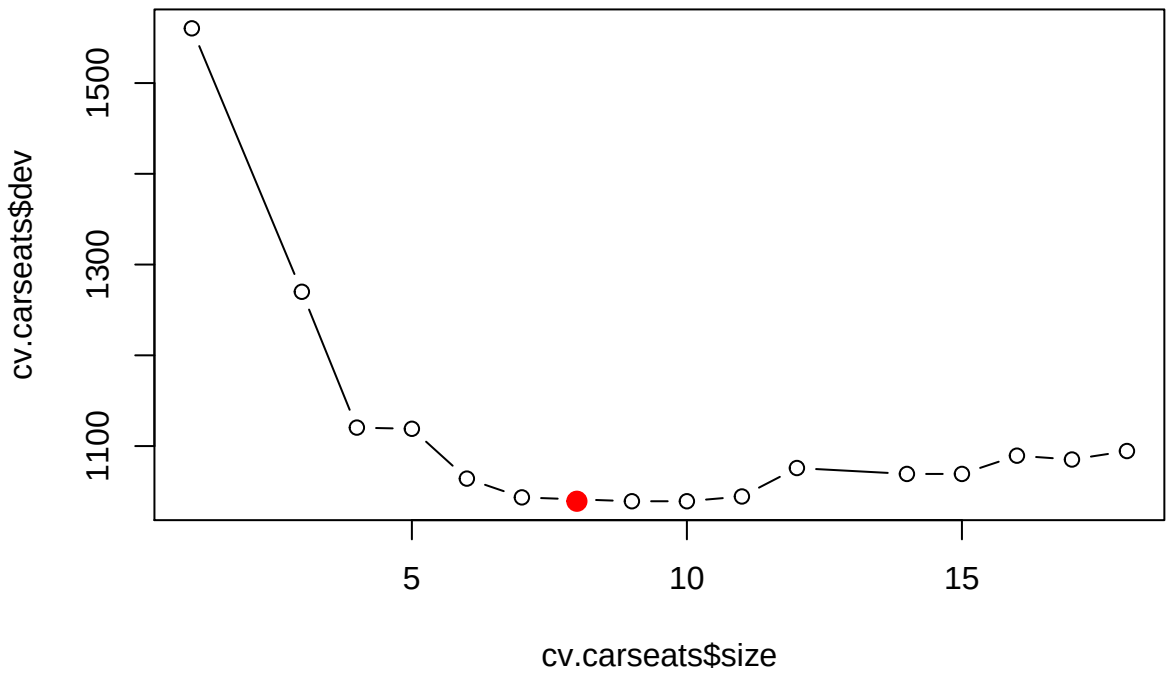



```
yhat <- predict(tree.carseats, newdata = Carseats.test)
mean((yhat - Carseats.test$Sales)^2) #Test MSE
```

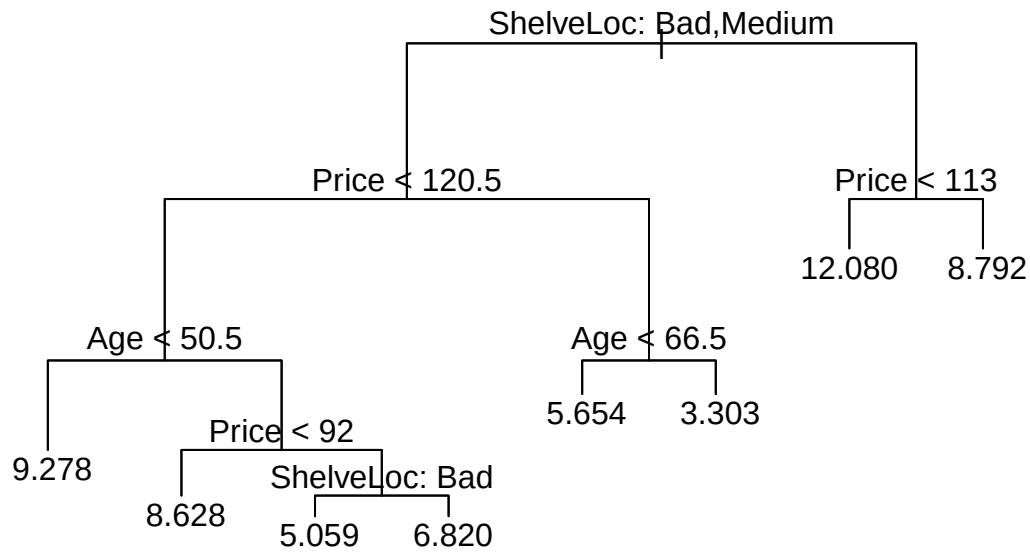
```
## [1] 4.148897
```

(c) The test MSE rose from 4.14 to 5.09.

```
cv.carseats <- cv.tree(tree.carseats)
plot(cv.carseats$size, cv.carseats$dev, type = "b")
tree.min <- which.min(cv.carseats$dev)
points(tree.min, cv.carseats$dev[tree.min], col = "red", cex = 2, pch = 20)
```




```
prune.carseats <- prune.tree(tree.carseats, best = 8)
plot(prune.carseats)
text(prune.carseats, pretty = 0)
```




```
yhat <- predict(prune.carseats, newdata = Carseats.test)
mean((yhat - Carseats.test$Sales)^2)
```

```
## [1] 5.09085
```

(d) The test MSE decreased down to 2.6 with bagging. The two most important variables are "Price" and "ShelveLoc"

```
library(randomForest)
bag.carseats <- randomForest(Sales ~ ., data = Carseats.train, mtry = 10, ntree = 500, importance = TRU
yhat.bag <- predict(bag.carseats, newdata = Carseats.test)
mean((yhat.bag - Carseats.test$Sales)^2)
```

```
## [1] 2.604369
```

```
importance(bag.carseats)
```

```
##                %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
```

(e) The test MSE is about 3.29 and $m = \sqrt{P}$. "Price" and "ShelveLoc" are still the most important variables

```
rf.carseats <- randomForest(Sales ~ ., data = Carseats.train, mtry = 3, ntree = 500, importance = TRUE)
yhat.rf <- predict(rf.carseats, newdata = Carseats.test)
mean((yhat.rf - Carseats.test$Sales)^2)
```

```
## [1] 3.296078
```

```
importance(rf.carseats)
```

```
##                %IncMSE IncNodePurity
## CompPrice    7.5233429     127.36625
## Income       4.3612064     119.19152
## Advertising 12.5799388     138.13567
## Population  -0.2974474     100.28836
## Price       37.1612032     383.12126
## ShelveLoc   30.3751253     246.19930
## Age         16.0261885     197.44865
## Education    1.7855151      63.87939
## Urban       -1.3928225      16.01173
## US           5.6393475      32.85850
```

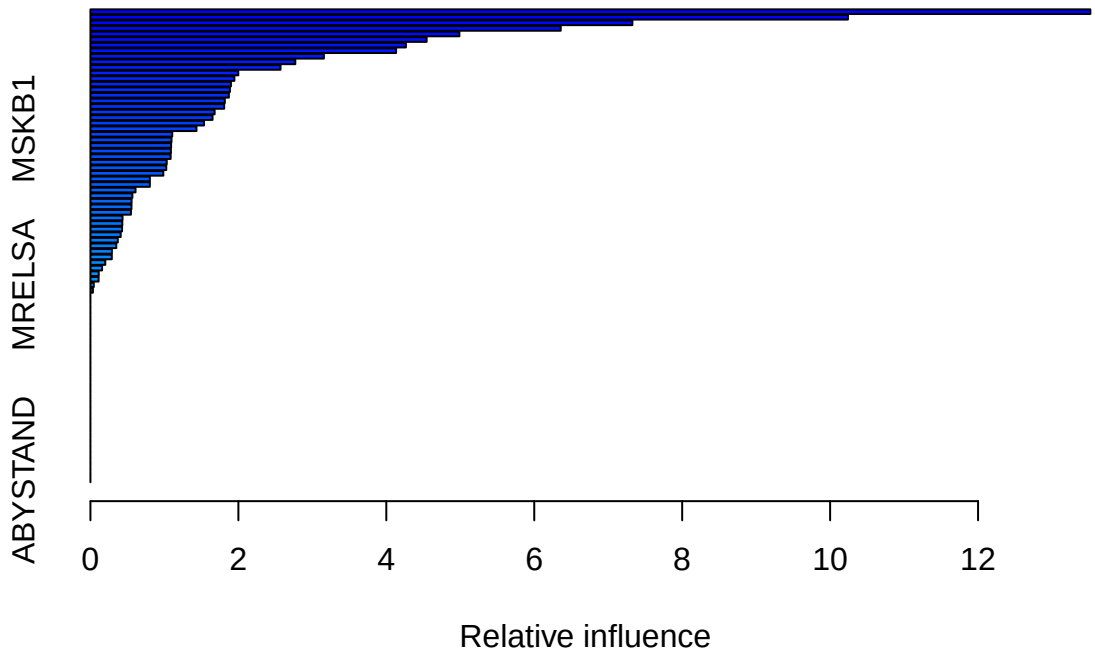
Ch.8 #11

(a)

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

(b) The two most important variables are "PPERSAUT" and "MKOOPKLA"

```
library(gbm)
set.seed(1)
boost.caravan <- gbm(Purchase ~ ., data = Caravan.train, distribution = "gaussian", n.trees = 1000, shr
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
```

(c) The fraction of people predicted by boosting to make purchase that actually make one is .215 and logistic regression gave the same result.

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

```
##    pred.test
##        0    1
##   0 4493   40
##   1  278   11
```

```
logit.caravan <- glm(Purchase ~ ., data = Caravan.train, family = "binomial")
probs.test2 <- predict(logit.caravan, Caravan.test, type = "response")
pred.test2 <- ifelse(probs.test > 0.2, 1, 0)
table(Caravan.test$Purchase, pred.test2)
```

```
##    pred.test2
##        0    1
##   0 4493   40
##   1  278   11
```

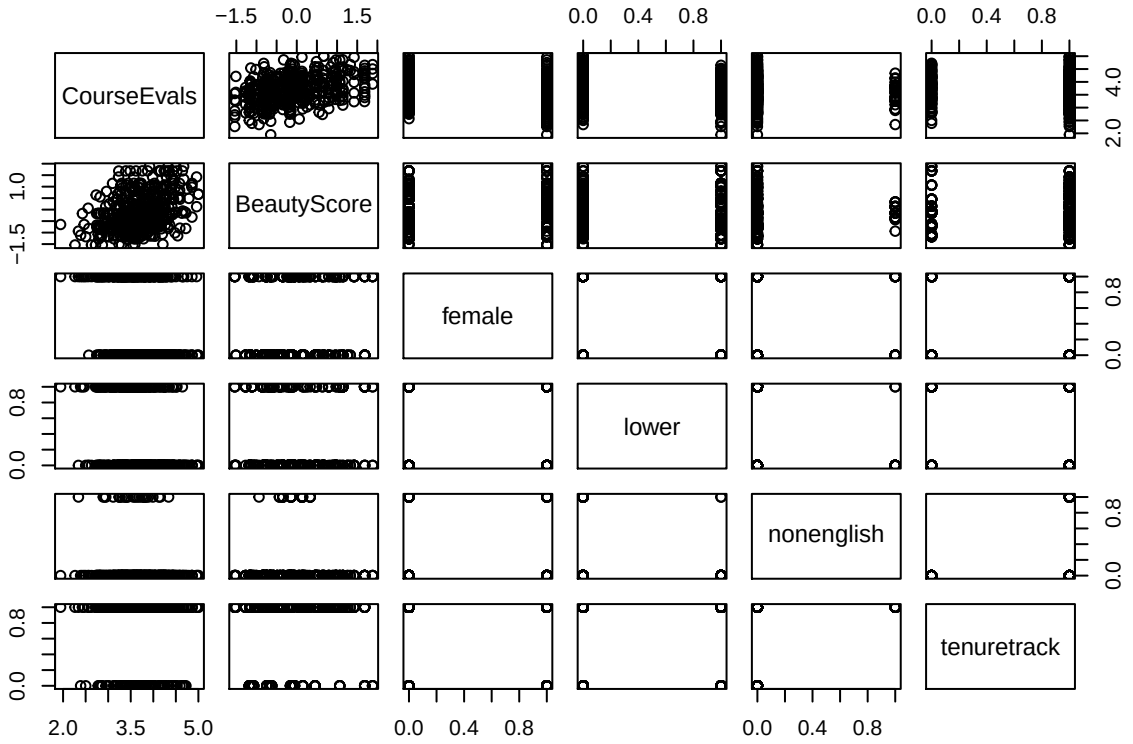
Worksheet Problems

Question 1

```
Beauty = read.csv("BeautyData.csv", header = TRUE)
attach(Beauty)
```

(1) When looking at the pairwise relations between all variables in the dataset, we see there are not any strong linear relationships at this point. Also the correlation between CourseEvals and BeautyScore is not strong enough to suggest a linear dependence.

```
pairs(Beauty)
```




```
cor(CourseEvals, BeautyScore)
```

```
## [1] 0.4070912
```

When running multiple linear regression models with different variable interactions, we do not get a great fitting model. Adding more interactions barely improves the r squared value which leads me to believe there is not a good fitting linear regression model. Also the confidence intervals show that the interaction between BeautyScore and female results in that interaction being thrown out because the interval includes 0. The interval for BeautyScore alone shows that it is significant because it is postive and does not include zero but we saw in the regression fits that the R^2 value was around .24 which indicates that the model is not a great fit.

```
lm.fit = lm(CourseEvals~BeautyScore)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = CourseEvals ~ BeautyScore)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.5936 -0.3346  0.0097  0.3702  1.2321
##
## Coefficients:
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.71340    0.02249 165.119   <2e-16 ***
## BeautyScore  0.27148    0.02837   9.569   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4809 on 461 degrees of freedom
## Multiple R-squared:  0.1657, Adjusted R-squared:  0.1639
## F-statistic: 91.57 on 1 and 461 DF,  p-value: < 2.2e-16
```

```
lm.fit = lm(CourseEvals~BeautyScore+female)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = CourseEvals ~ BeautyScore + female)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.40303 -0.29780  0.00792  0.31807  1.14350
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.84439    0.02833 135.706  < 2e-16 ***
## BeautyScore  0.29559    0.02720  10.869  < 2e-16 ***
## female      -0.30597    0.04339  -7.051 6.53e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4573 on 460 degrees of freedom
## Multiple R-squared:  0.2471, Adjusted R-squared:  0.2438
## F-statistic: 75.49 on 2 and 460 DF,  p-value: < 2.2e-16
```

```
lm.fit = lm(CourseEvals~BeautyScore+female+BeautyScore:female)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = CourseEvals ~ BeautyScore + female + BeautyScore:female)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.37133 -0.30235 -0.00191  0.31627  1.15215
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)         3.83751    0.02861 134.133  < 2e-16 ***
## BeautyScore         0.25574    0.03690   6.930 1.43e-11 ***
## female             -0.30038    0.04346  -6.912 1.61e-11 ***
## BeautyScore:female  0.08685    0.05448   1.594    0.112
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 0.4566 on 459 degrees of freedom
## Multiple R-squared:  0.2512, Adjusted R-squared:  0.2464
## F-statistic: 51.34 on 3 and 459 DF,  p-value: < 2.2e-16
```

```
confint(lm.fit)
```

```
##                           2.5 %     97.5 %
## (Intercept)          3.78128285  3.8937272
## BeautyScore          0.18321737  0.3282643
## female              -0.38578854 -0.2149781
## BeautyScore:female -0.02022196  0.1939191
```

When I ran forward stepwise selection to determine variable importance and interactions, I got the following results.

```
library(leaps)
regfit.fwd=regsubsets(CourseEvals~.,data=Beauty, method="forward")
summary(regfit.fwd)
```

```
## Subset selection object
## Call: regsubsets.formula(CourseEvals ~ ., data = Beauty, method = "forward")
## 5 Variables  (and intercept)
##             Forced in Forced out
## BeautyScore     FALSE      FALSE
## female          FALSE      FALSE
## lower           FALSE      FALSE
## nonenglish      FALSE      FALSE
## tenuretrack     FALSE      FALSE
## 1 subsets of each size up to 5
## Selection Algorithm: forward
##          BeautyScore female lower nonenglish tenuretrack
## 1  ( 1 ) "*"         " "    " "   " "         " "
## 2  ( 1 ) "*"         "*"    " "   " "         " "
## 3  ( 1 ) "*"         "*"    "*"   " "         " "
## 4  ( 1 ) "*"         "*"    "*"   "*"         " "
## 5  ( 1 ) "*"         "*"    "*"   "*"         "*"
```

```
coef(regfit.fwd, 5)
```

```
## (Intercept) BeautyScore      female       lower  nonenglish tenuretrack
##  4.06541587  0.30414581 -0.33199440 -0.34255058 -0.25808207 -0.09945058
```

In conclusion, I did not find any strong relationships between variables involving beauty and course evaluation because beauty and productivity are tough variables to measure. Perhaps a hidden driver of CourseEval would be how productive the teacher was and this variable is not measured directly so we can only conclude that beauty has a small effect on CourseEval.

(2) When Dr. Hamermesh said "Disentangling wheter this outcome represents productivity or discrimination is, as with the issue generally, probably impossible" he was reffering to the fact that it is impossible to distinguish whether better looking people receive higher wages for their looks or for working harder. It is very hard to determine what variable leads to a higher income because their are interactions between many variables which make it difficult to analyze the effect of just one variable.

Question 2

```
MidCity = read.csv("MidCity.csv",header=T)
attach(MidCity)

n = dim(MidCity)[1]

dn1 = rep(0,n)
dn1[Nbhd==1]=1

dn2 = rep(0,n)
dn2[Nbhd==2]=1

dn3 = rep(0,n)
dn3[Nbhd==3]=1

BR = rep(0,n)
BR[Brick=="Yes"]=1

Price = Price/1000
SqFt = SqFt/1000

Nbhd = factor(Nbhd)
MidCityModel = lm(Price~BR+dn2+dn3+Offers+SqFt+Bedrooms+Bathrooms)
confint(MidCityModel)
```

```
##                   2.5 %    97.5 %
## (Intercept) -15.417947 19.736943
## BR           13.373887 21.220812
## dn2          -6.306008  3.184850
## dn3          14.446328 26.915747
## Offers      -10.415271 -6.119706
## SqFt         41.640344 64.347137
## Bedrooms      1.083042  7.410546
## Bathrooms     3.691696 12.074861
```

(1) There seems to be a premium for brick houses because the confidence interval does not include zero and the entire interval is positive for the brick coefficient.

(2) Again the confidence interval is postive and does not include zero so we can conclude that there is a premium to live in neighborhood 3.

(3)

```
MidCityModel = lm(Price~BR+dn2+dn3+Offers+SqFt+Bedrooms+Bathrooms+dn3:BR)
confint(MidCityModel)
```

```
##                   2.5 %    97.5 %
## (Intercept) -14.229279 20.249266
## BR            9.063223 18.589707
## dn2          -5.378691  4.032634
## dn3          10.526207 23.956619
```

```
## Offers      -10.508647 -6.293529
## SqFt         42.904932 65.224631
## Bedrooms      1.594333  7.841994
## Bathrooms     2.197708 10.729022
## BR:dn3        1.933918 18.429237
```

Based off the confidence intervals, we can see that the interaction between neighborhood 3 and brick houses results in a postive interval that does not include zero so that variable is statstically significant or in other words there is a premium for brick houses in neighborhood 3.

(4) Since the confidence interval for the neighborhood 2 variable includes zero, we can conclude that setting that coefficient to zero i.e. combining it into neighborhood 1 is reasonable.

Question 3

(1) While there might be a strong correlation between those two variables, there could actually be a different causation for the results. Statistics is not enough to get the correct causation of crime in cities. You have to isolate the effect of adding more police to a city not in response to increased crime in order to isolate the effect of more police on the streets, i.e. when Washington D.C adds more police for high terriosts threats and not for higher crime rates.

(2) Researchers from UPENN isolated the effect of addiing more police by observing crime rates on high terriost alert days. This allowed them to see how crime rates were affected when more police were on the streets even though the police were not specifically there to stop normal crime. From the table, we see that on high alert days the crime rates went down by 6 or 7 crimes. Even though the police were looking for terriosts, the normal every day criminals did not commit has many crimes in response to increased police on the streets.

(3) They controlled for METRO ridership in order to see if the number of victims changed on high alert days. They oberserved that the victims of normal every day crimes stayed constant on high alert days which meant that the victims were not scared on high alert days and went on with their normal routines while the criminals stayed off the streets. This captured the notion that crimes went down because there were less victims but that was disproven because ridership stayed constant.

(4) I believe the model being estimated is a logistic multiple regression that is trying to predict the total daily number of crimes in D.C. and the important variables are district 1 and the high alert variable along with the interaction between the two. The conclusion is that the number of crimes per day is dependent on the district and if is a high alert day or not.