ExamVincentChou

Vincent Chou

August 3, 2016

2.10

A

To begin, load in the Boston data set. The Boston data set is part of the MASS library in R. Now the data set is contained in the object Boston. Read about the data set.
How many rows are in this data set?
How many columns?
What do the rows and columns represent?

```
rm(list=ls())
library(MASS)
```

Loaded the library "MASS"

```
?Boston
```

Examined the Boston dataframe.

```
bos = Boston
```

Stored Boston as bos for ease of use.

```
dim(bos)
```

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

Finds the rows and columns.

```
names(bos) #?Boston goes into detail
```

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

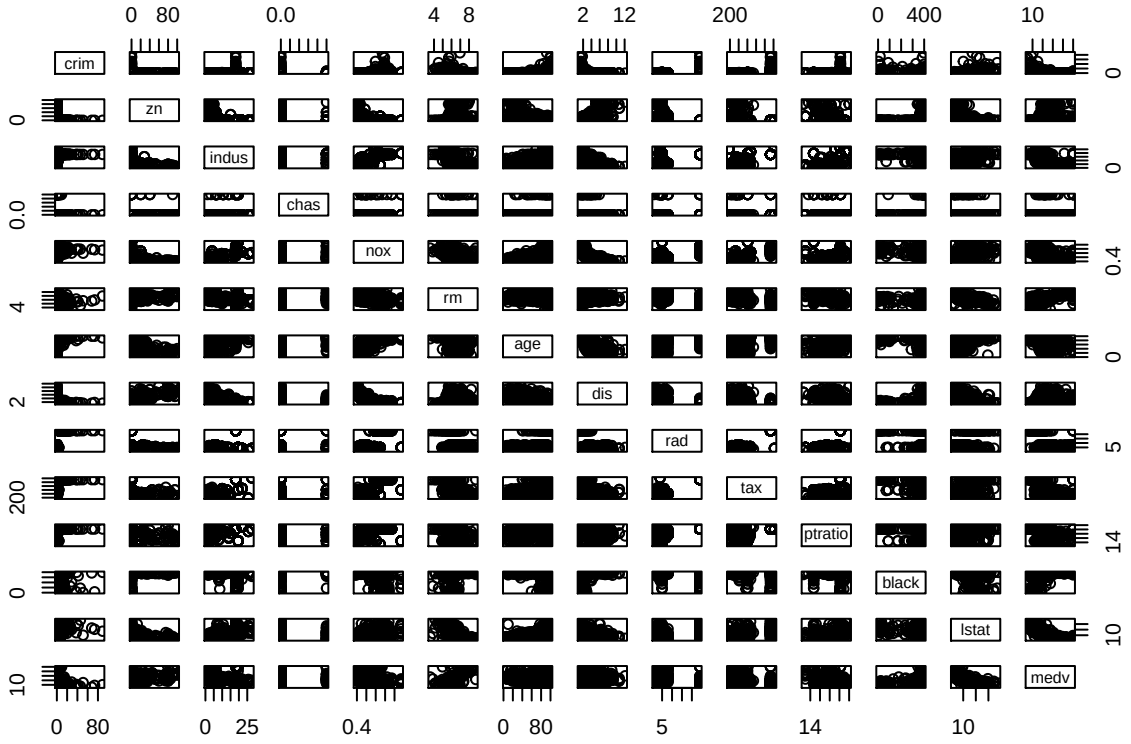
These are what the columns represent. The Boston help file details these variables:
crim - per capita crime rate by town.
zn - proportion of residential land zoned for lots over 25,000 sq.ft.
indus - proportion of non-retail business acres per town. chas - Charles River dummy variable (= 1 if tract bounds river; 0 otherwise).
nox - nitrogen oxides concentration (parts per 10 million).
rm - average number of rooms per dwelling.

age - proportion of owner-occupied units built prior to 1940.
dis - weighted mean of distances to five Boston employment centres.
rad - index of accessibility to radial highways.
tax - full-value property-tax rate per $10,000.
ptratio - pupil-teacher ratio by town.
black - 1000(Bk - 0.63)ˆ2 where Bk is the proportion of blacks by town.
lstat - lower status of the population (percent).
medv - median value of owner-occupied homes in $1000s.

B

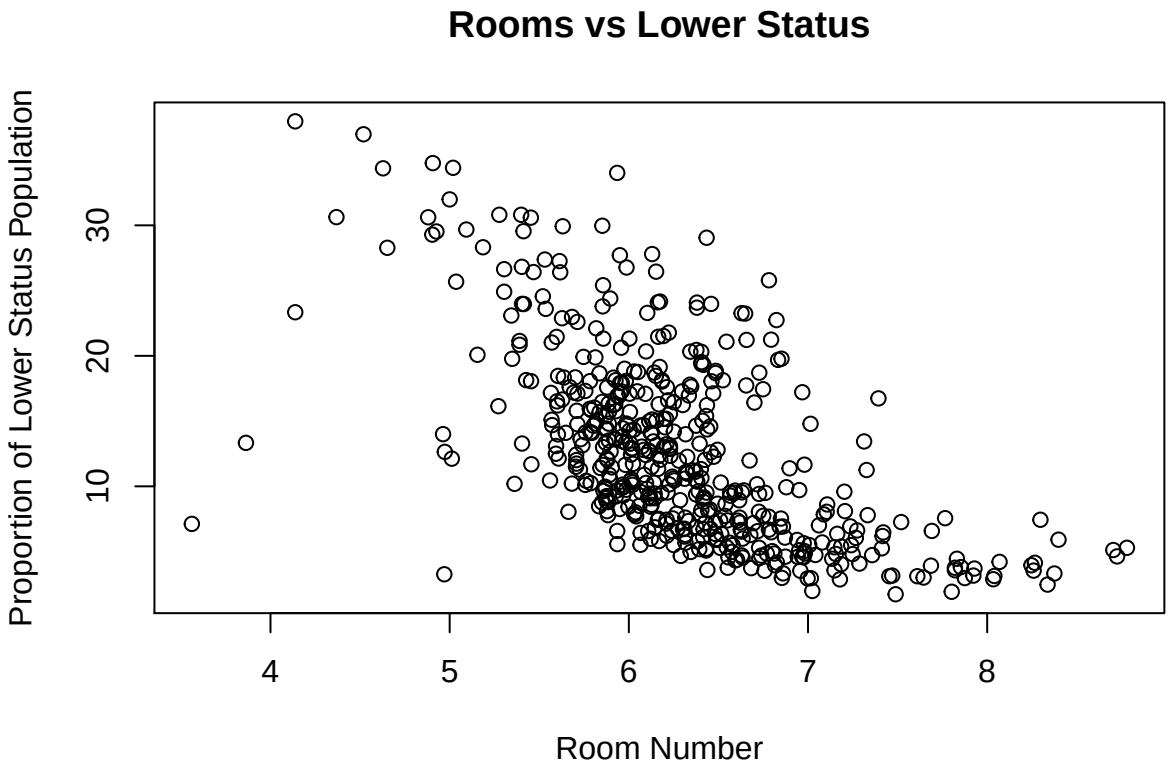
Make some pairwise scatterplots of the predictors (columns) in this data set. Describe your findings. ### C
Are any of the predictors associated with per capita crime rate? If so, explain the relationship.

```
pairs(bos)
```

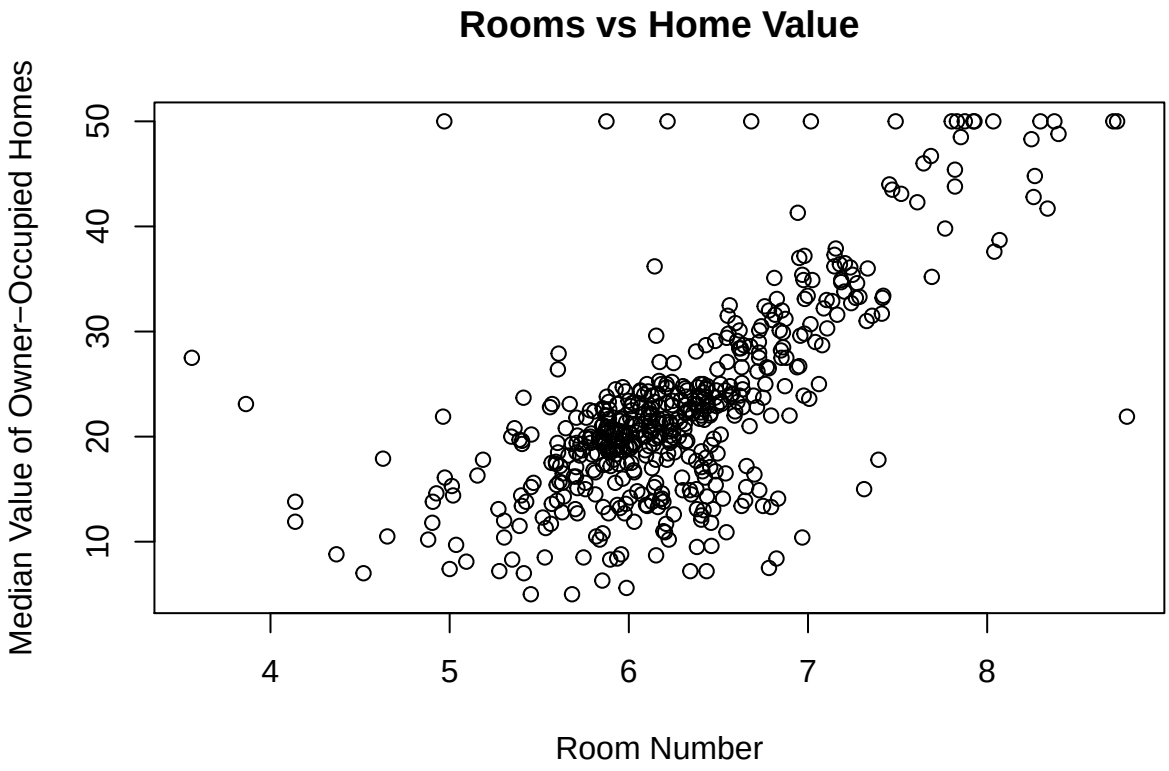


Examined all pairs of scatterplots to look for a correlation.

```
plot(bos$rm,bos$lstat, xlab = "Room Number", ylab = "Proportion of Lower Status Population", main = "Ro
```

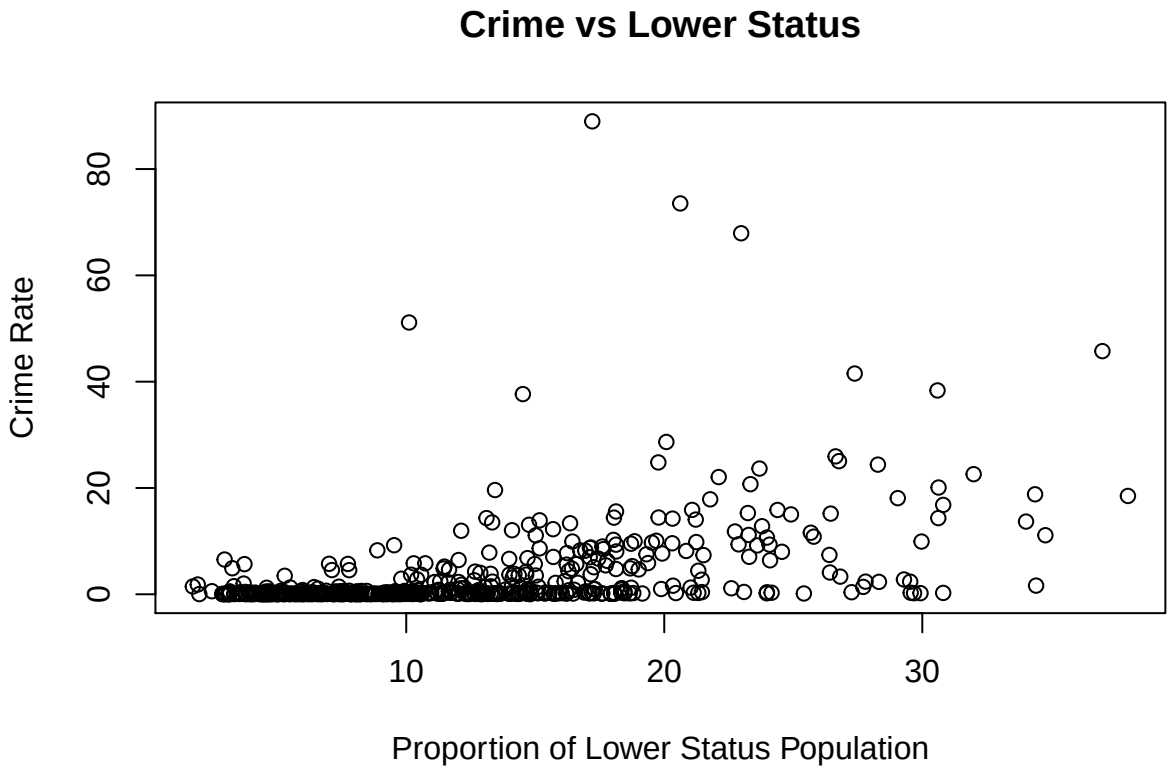



```
plot(bos$rm,bos$medv, xlab = "Room Number", ylab = "Median Value of Owner-Occupied Homes", main = "Rooms
```

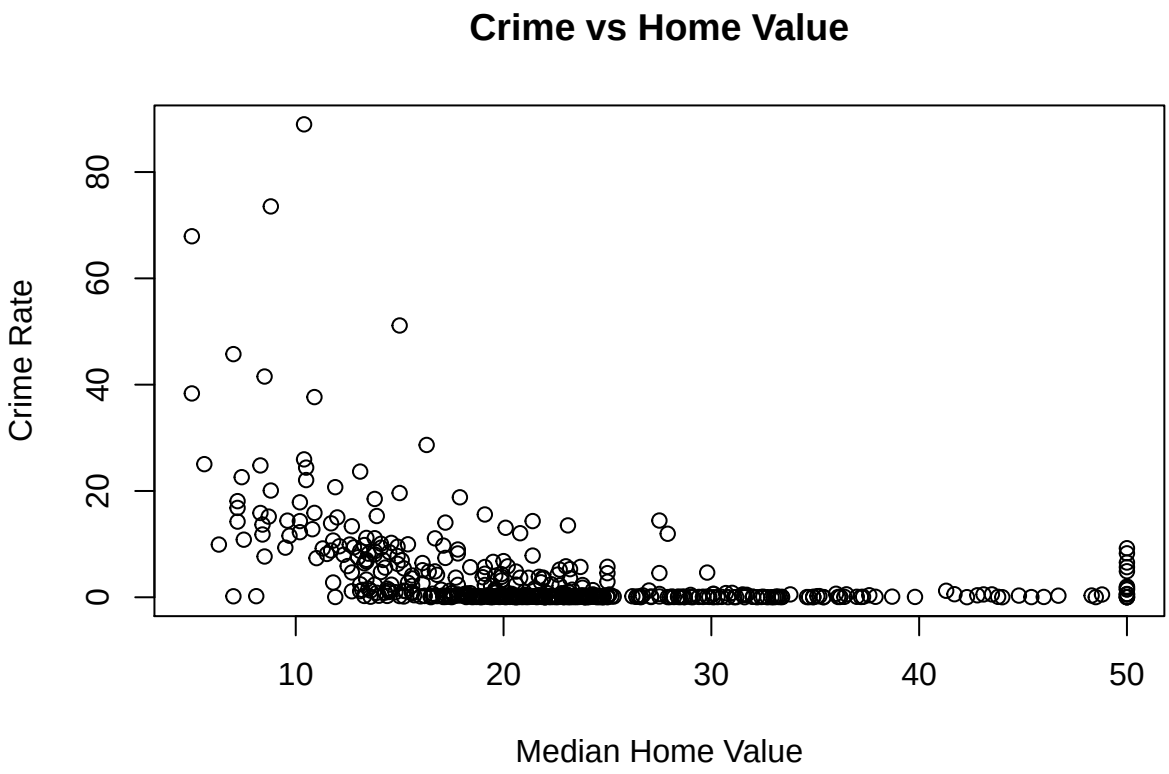


As expected the number of rooms in the homes of the Boston suburbs correlated negatively with lower status and positively with median house value.

```
plot(bos$lstat,bos$crim, ylab = "Crime Rate", xlab = "Proportion of Lower Status Population", main = "C
```

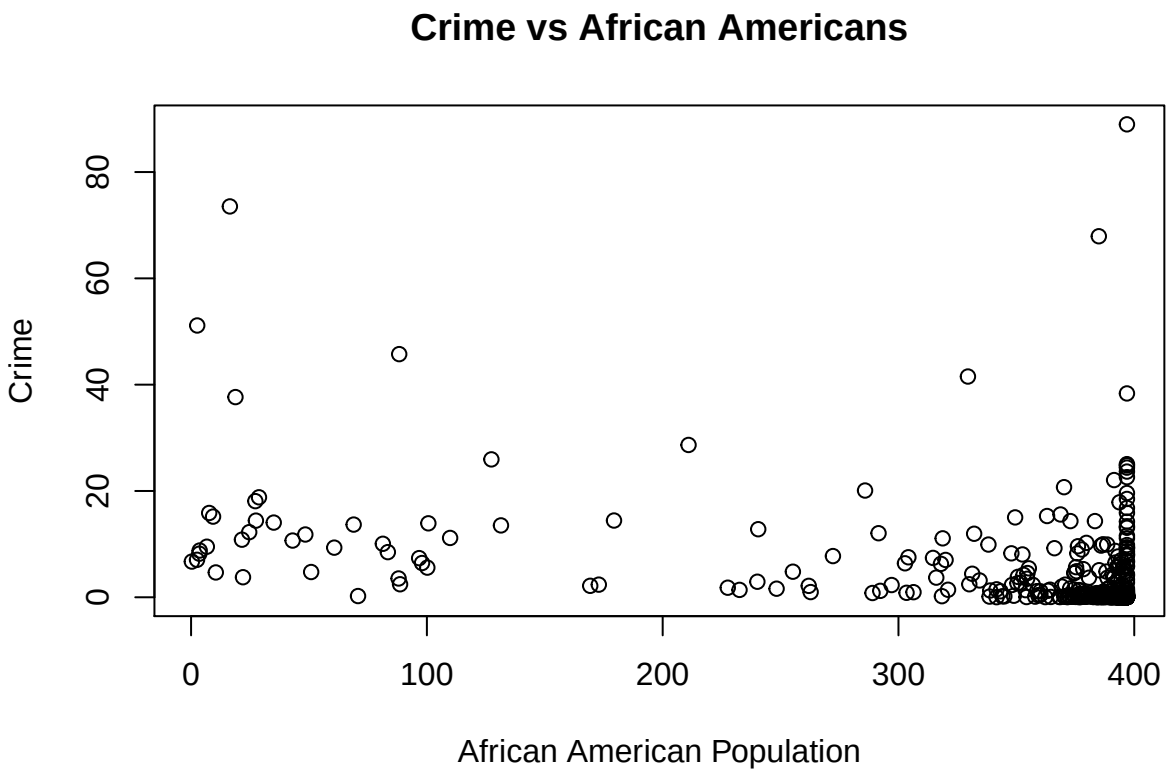



```
plot(bos$medv,bos$crim, ylab = "Crime Rate", xlab = "Median Home Value", main = "Crime vs Home Value")
```



And, as expected, crime correlated positively with lower status population percentage and negatively with median home value.

```
plot(bos$black,bos$crim, ylab = "Crime", xlab = "African American Population", main = "Crime vs African
```



Interestingly there appeared to be a range of African American population proportion where crime dropped significantly. Some may take this as ossible evidence that more homogenous areas tend to have higher crime; however, we must note that this dip coincides with a lack of data points.

```
plot(bos$age,bos$crim, ylab = "Crime", xlab = "Homes made before 1940", main = "Crime vs Homes made befo
```


```
plot(bos$dis,bos$crim, ylab = "Crime", xlab = "Distance from Major Employment Centers", main = "Crime vs
```

Interestingly, there was a clear positive relationship between the proportion of owner-occupied homes that were made before 1940 and crime.

There was also a clear negative relationship between crime and distance from the major urban areas of Boston. It seems likely that older homes are are closer to these areas (being developed first) which explains the crime and age relationship. When plotting age and distance, that idea is confirmed.

```
plot(bos$dis,bos$crim, ylab = "Homes made before 1940", xlab = "Distance from Major Employment centers"
```

D

Do any of the suburbs of Boston appear to have particularly high crime rates? Tax rates? Pupil-teacher ratios? Comment on the range of each predictor.

E

How many of the suburbs in this data set bound the Charles river?

```
charles = bos[bos$chas ==1,]
nrow(charles)
```

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

35 suburbs are adjacent to Charles River.

F

What is the median pupil-teacher ratio among the towns in this data set?

```
median(bos$ptratio)
```

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

Returns the median pupil-teacher ratio (19.05).

G

> Which suburb of Boston has lowest median value of owneroccupied homes? What are the values of the other predictors for that suburb, and how do those values compare to the overall ranges for those predictors? Comment on your findings.

```
which.min(bos$medv)
```

```
## [1] 399
```

The town at index 399 is the town with the lowest valued homes.

```
bos[which.min(bos$medv),]
```

```
##        crim zn indus chas   nox    rm age    dis rad tax ptratio black
## 399 38.3518  0  18.1    0 0.693 5.453 100 1.4896  24 666    20.2 396.9
##     lstat medv
## 399 30.59    5
```

And that is the summary of the town.

```
summary(bos)
```

Compared to the rest of the Boston suburbs, several things pop out. Crime is 10 times higher than average. There are NO large residential lots. It is extremely old with every home being build before 1940, and it has the highest proportion of African Americans of the sampled suburbs.

H

> In this data set, how many of the suburbs average more than seven rooms per dwelling? More than eight rooms per dwelling? Comment on the suburbs that average more than eight rooms per dwelling.

```
nrow(bos[bos$rm > 7,]) #returns 64 for number of suburbs with more than 7 rooms
```

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

Returns the number of suburbs that have more than 7 rooms on average (64).

```
nrow(bos[bos$rm > 8,]) #returns 13 for number of suburbs with more than 8 rooms
```

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

Returns the number of suburbs that have more than 8 rooms of average(13)

```
bos[bos$rm > 8,] #data for those suburbs with more than 8 rooms
summary(bos) #Something to compare these suburbs with
```

```
summary(bos[bos$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
```

These suburbs have much less crime and industry compared to others. Twice as many suburbs as would be expected are near the Charles River. They appear closer to employment areas than most. There is a far lower proportion of lower status populations in these suburbs. Homes are worth significantly more. Much of this is expected since larger homes are likely owned by wealthier people.

3.15

```
rm(list=ls())
library(MASS)
b =Boston
names(b)
```

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

Part a.

For each predictor, fit a simple linear regression model to predict the response. Describe your results. In which of the models is there a statistically significant association between the predictor and the response? Create some plots to back up your assertions.

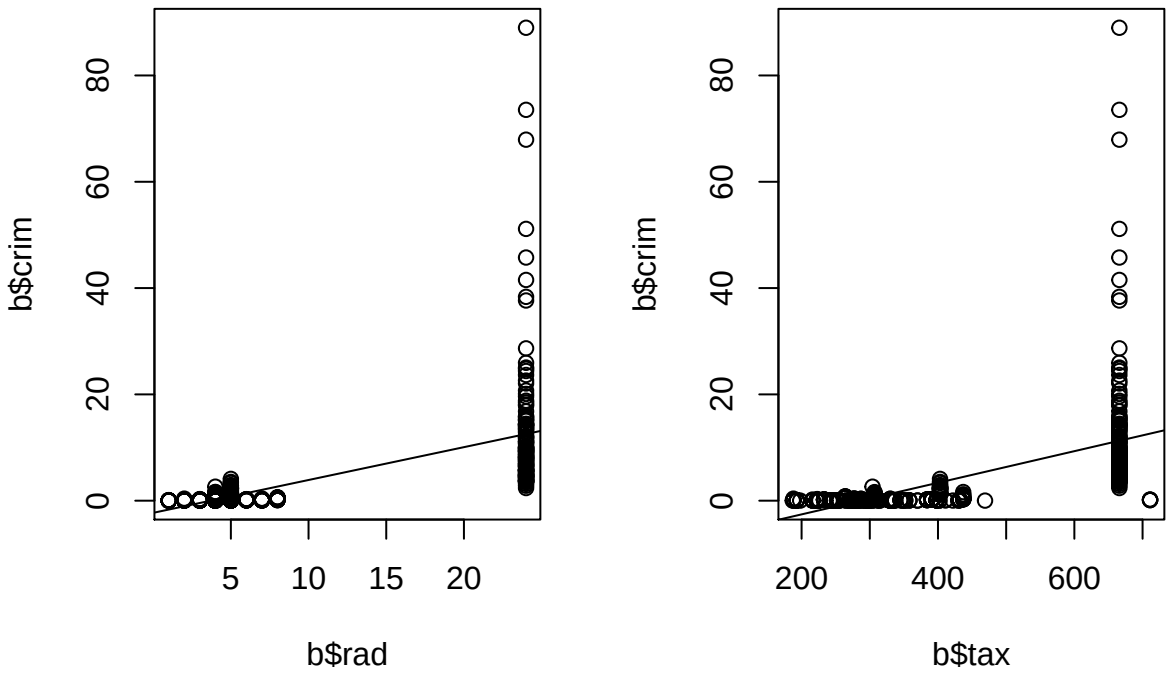
```r
modList = vector(mode = "list", length = 13)
#Couldn't figure out how to change the model in a loop, so I had to go old fashioned
modList[[1]] <- lm(crim~zn,data =b)
modList[[2]] <- lm(crim~indus, data = b)
modList[[3]] <- lm(crim~chas, data = b)
modList[[4]] <- lm(crim~nox, data = b)
modList[[5]] <- lm(crim~rm, data = b)
modList[[6]] <- lm(crim~age, data = b)
modList[[7]] <- lm(crim~dis, data = b)
modList[[8]] <- lm(crim~rad, data = b)
modList[[9]] <- lm(crim~tax, data = b)
modList[[10]] <- lm(crim~ptratio, data = b)
modList[[11]] <- lm(crim~black, data = b)
modList[[12]] <- lm(crim~lstat, data = b)
modList[[13]] <- lm(crim~medv, data = b)
#Pull out just the sigma
errors = list()
for (i in seq(1:13)) {
  errors = c(errors, summary(modList[[i]])[6])
}
errors
```

```
## $sigma
## [1] 8.43529
##
## $sigma
## [1] 7.866281
##
## $sigma
## [1] 8.596615
##
## $sigma
## [1] 7.809972
##
## $sigma
## [1] 8.400586
##
## $sigma
## [1] 8.056649
##
## $sigma
## [1] 7.965369
##
## $sigma
## [1] 6.717752
##
## $sigma
## [1] 6.996901
##
## $sigma
## [1] 8.240212
##
## $sigma
```

```
## [1] 7.94615
##
## $sigma
## [1] 7.664461
##
## $sigma
## [1] 7.934451
```

From the list of errors, the rad and tax variables seem to be the best predictors of crime.

```
par(mfrow=c(1,2))
plot(b$rad, b$crim)
abline(lm(b$crim~b$rad))
plot(b$tax, b$crim)
abline(lm(b$crim~b$tax))
```



Even though they're our best predictors they don't seem too accurate.

B

Fit a multiple regression model to predict the response using all of the predictors. Describe your results. For which predictors can we reject the null hypothesis H0 - Beta_j = 0?

```r
allMod <- lm(crim~., data = b)

summary(allMod)
```

```
##
## Call:
## lm(formula = crim ~ ., data = b)
##
## 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
```

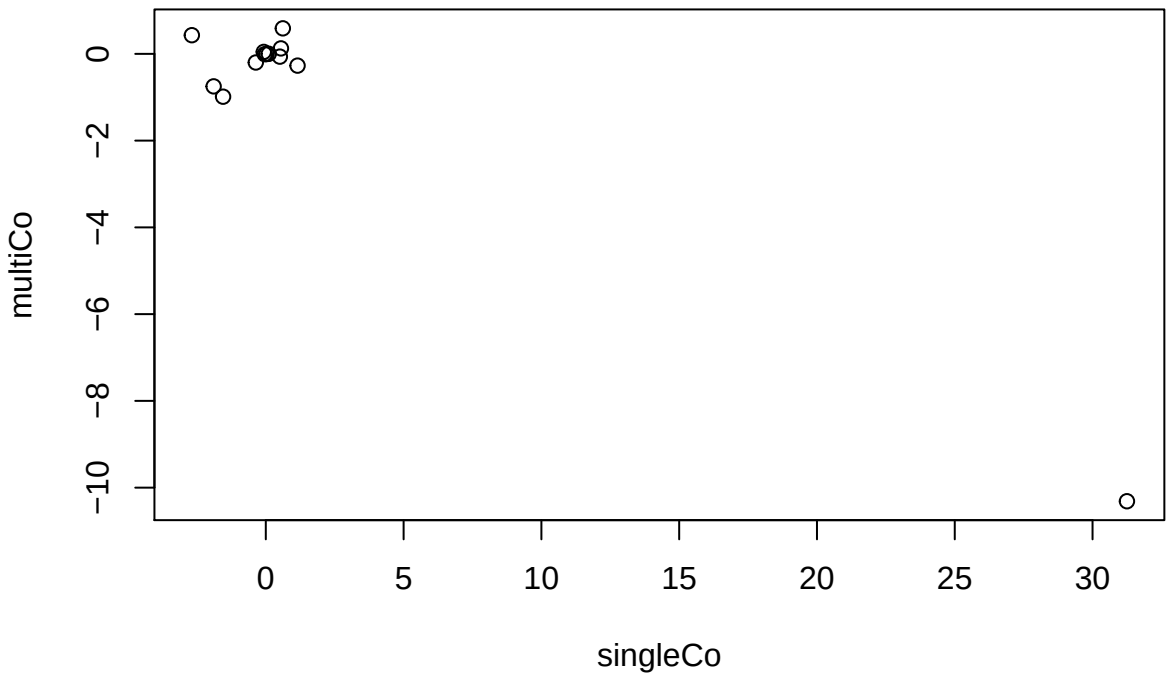
From the summary, all variables marked by at least one * we can reject H_0. These variables are zn, indus, dis, rad, black, and medv.

C

```r
singleCo <- list()

for (i in seq(1,13)) {
  singleCo[i] <- summary(modList[[i]])$coefficients[2]
}


multiCo <- summary(allMod)$coefficients[2:14]
par(mfrow= c(1,1))
plot(singleCo, multiCo)
```



The coefficients for multiple linear regression seem consistently smaller than those for univariate liner regression. The nox variable's coefficient seemed is the outlier point. I am not sure what to make of it.

```
modListp = list()
b$chas = as.factor(b$chas)
modListp[[1]] <- lm(crim~poly(zn,3),data =b)
summary(modListp[[1]])
```

```
##
## Call:
## lm(formula = crim ~ poly(zn, 3), data = b)
##
## 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
```

```
modListp[[2]] <- lm(crim~poly(indus,3), data = b)
summary(modListp[[2]])
```

```
##
## Call:
## lm(formula = crim ~ poly(indus, 3), data = b)
##
## 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
```

```
modListp[[3]] <- lm(crim~chas, data = b)
summary(modListp[[3]])
```

```
##
## Call:
## lm(formula = crim ~ chas, data = b)
##
## 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
```

```
modListp[[4]] <- lm(crim~poly(nox,3), data = b)
summary(modListp[[4]])
```

```
##
## Call:
## lm(formula = crim ~ poly(nox, 3), data = b)
##
## 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
```

```
modListp[[5]] <- lm(crim~poly(rm,3), data = b)
summary(modListp[[5]])
```

```
##
## Call:
## lm(formula = crim ~ poly(rm, 3), data = b)
##
## 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
```

```
modListp[[6]] <- lm(crim~poly(age,3), data = b)
summary(modListp[[6]])
```

```
##
## Call:
## lm(formula = crim ~ poly(age, 3), data = b)
##
## 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
```

```
modListp[[7]] <- lm(crim~poly(dis,3), data = b)
summary(modListp[[7]])
```

```
##
## Call:
## lm(formula = crim ~ poly(dis, 3), data = b)
##
## 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
```

```
modListp[[8]] <- lm(crim~poly(rad,3), data = b)
summary(modListp[[8]])
```

```
##
## Call:
## lm(formula = crim ~ poly(rad, 3), data = b)
##
## 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
```

```
modListp[[9]] <- lm(crim~poly(tax,3), data = b)
summary(modListp[[9]])
```

```
##
## Call:
## lm(formula = crim ~ poly(tax, 3), data = b)
##
## 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
```

```
modListp[[10]] <- lm(crim~poly(ptratio,3), data = b)
summary(modListp[[10]])
```

```
##
## Call:
## lm(formula = crim ~ poly(ptratio, 3), data = b)
##
## 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
```

```
modListp[[11]] <- lm(crim~poly(black,3), data = b)
summary(modListp[[11]])
```

```
##
## Call:
## lm(formula = crim ~ poly(black, 3), data = b)
##
## 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
```

```
modListp[[12]] <- lm(crim~poly(lstat,3), data = b)
summary(modListp[[12]])
```

```
##
## Call:
## lm(formula = crim ~ poly(lstat, 3), data = b)
##
## 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
```

```
modListp[[13]] <- lm(crim~poly(medv,3), data = b)
summary(modListp[[13]])
```

```
##
## Call:
## lm(formula = crim ~ poly(medv, 3), data = b)
##
## 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
```

All of the variables except black show signs of a non-linear association.

6.9

```
rm(list = ls())
library(glmnet)
```

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

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

```
## Loaded glmnet 2.0-5
```

```
college <- read.csv("College.csv", row.names = 1)
```

The first column contains the names of the universities.

A.

```
set.seed(18)
trainSize = dim(college)[1] / 2 #making the training and test sizes equal, not necessarily necessary
train = sample(1:dim(college)[1], trainSize) #randomly pulling out half of the indices of all the possi
test = -train #The opposite mask
training = college[train, ] #Now my training set has only the rows defined by the train indices.
testing = college[test, ] #now my test set is the rows that weren't put into training
```

B.

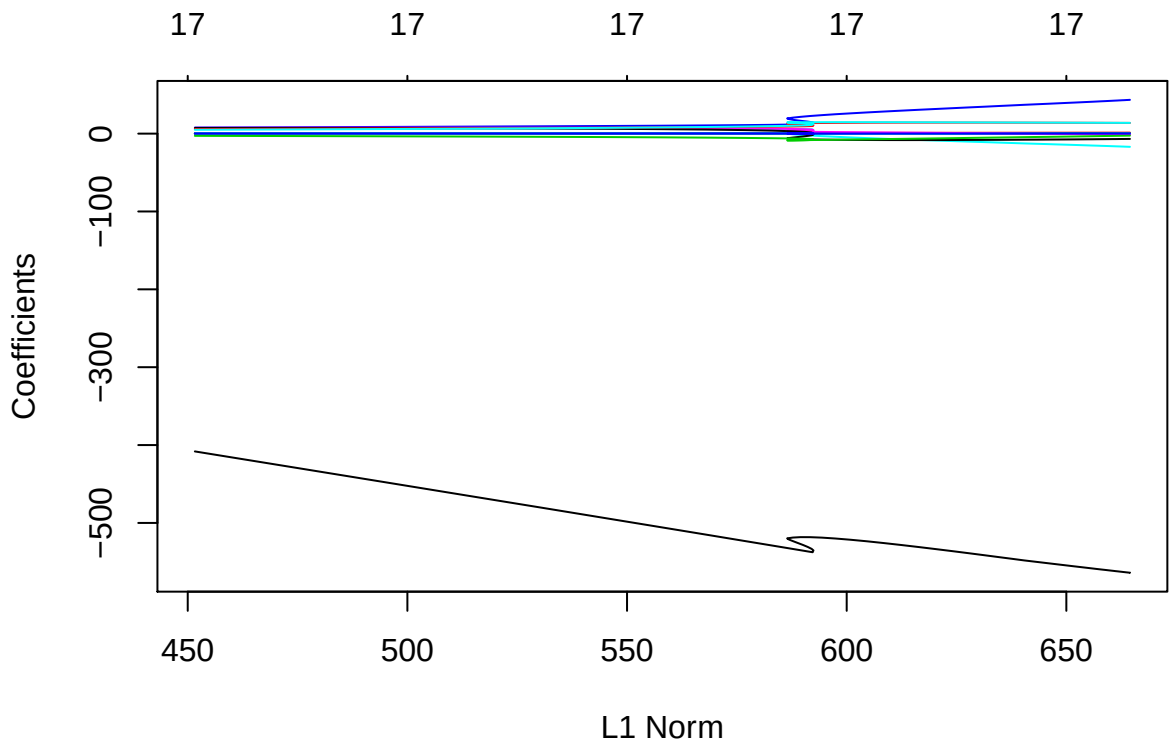
```
m1 = lm(Apps~., data=training) #making the linear regression model
p1 = predict(m1, testing) #predicting given the testing Xs
e1 <- sqrt(mean((p1-testing[, "Apps"])^2)) #the RMSE of this model
e1
```

```
## [1] 1268.529
```

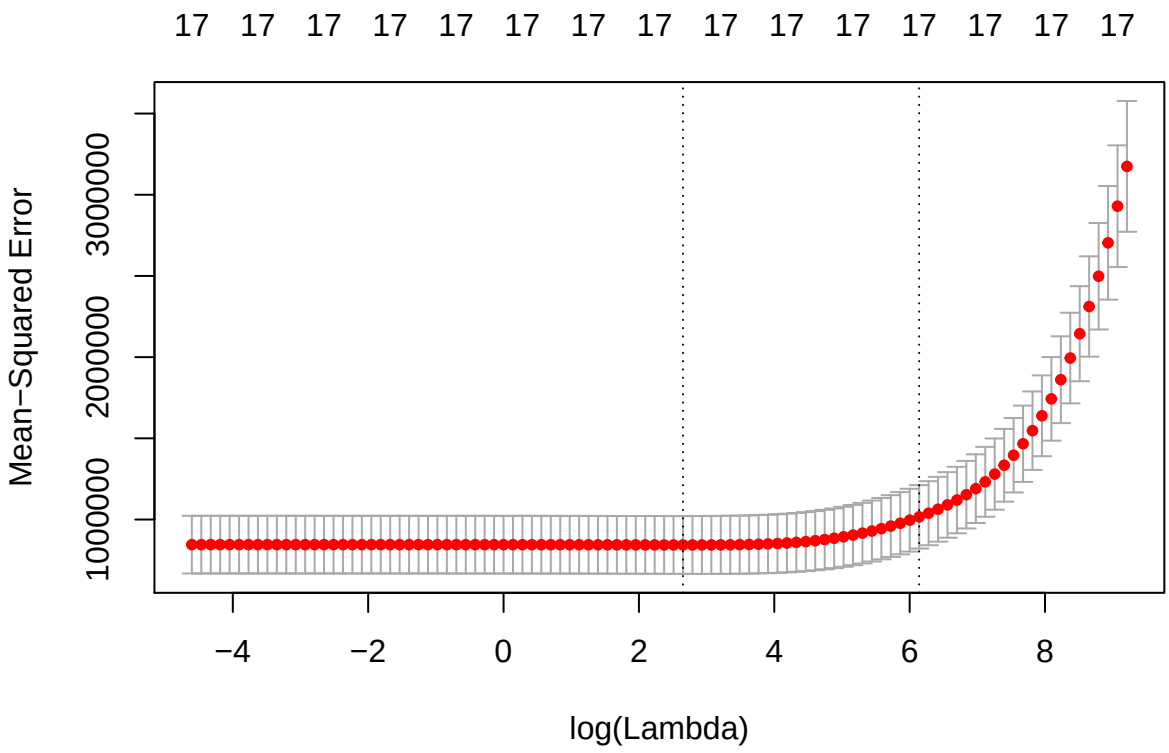
So this returns the RMSE for a linear model where Y is "Apps" and all other variables are the Xs.

C.

```
trainMat = model.matrix(Apps~., data=training)[,-1] #building a model matrix of the training set
trainY = training$Apps
testMat = model.matrix(Apps~., data=testing)[,-1] #building a model matrix of the test set
grid = 10^seq(4, -2, length=100) #Making 100 possible lambdas from 0.01 to 10000
ridge <- glmnet(trainMat, trainY, alpha = 0, lambda = grid, thresh = 1e-12, standardize = TRUE)
plot(ridge)
```



```
cvRidge = cv.glmnet(trainMat, trainY, alpha = 0, lambda = grid, thresh = 1e-12, standardize = TRUE)
plot(cvRidge)
```




```
bestLam = cvRidge$lambda.min
bestLam
```

```
## [1] 14.17474
```

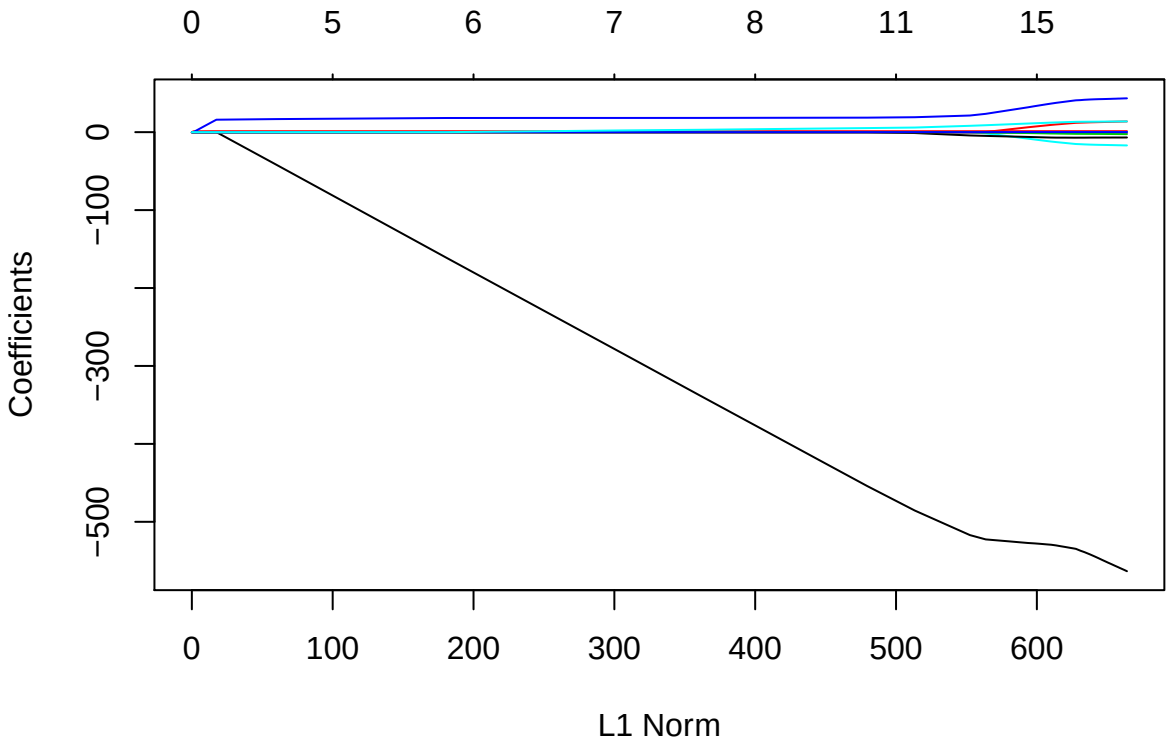
```
p2 = predict(ridge, newx= testMat,s= bestLam)
e2 <- sqrt(mean((testing[,"Apps"]-p2)^2))
e2
```

```
## [1] 1289.872
```

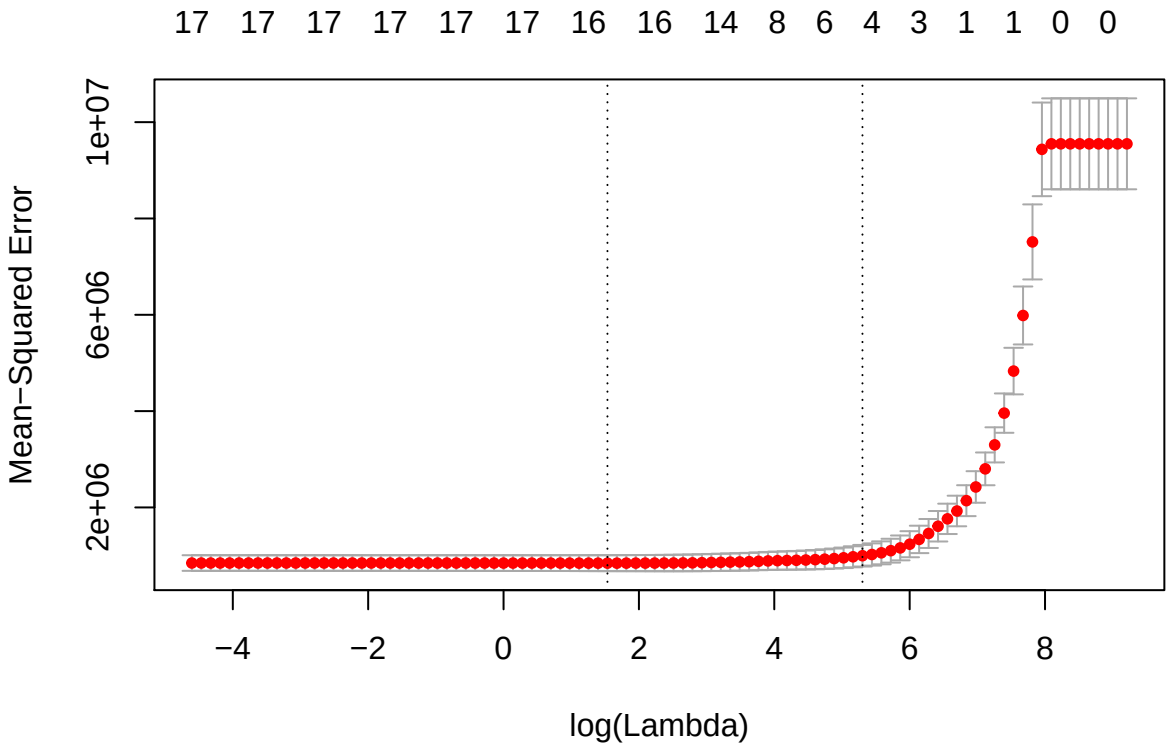
This RMSE is higher than the RMSE for the linear model.

D

```
lass <- glmnet(trainMat, training[,"Apps"], alpha = 1 , lambda = grid)
plot(lass)
```




```
set.seed(18)
cvLass <- cv.glmnet(trainMat, training[,"Apps"], alpha =1 , lambda = grid)
plot(cvLass)
```




```
bessLam <- cvLass$lambda.min
bessLam
```

```
## [1] 4.641589
```

```
p3 <- predict(lass, s= bessLam, newx = testMat)
e3 <- sqrt(mean((p3-testing[,"Apps"])^2))
e3
```

```
## [1] 1265.968
```

```
out <-  glmnet(model.matrix(Apps~., data = college), college[,17], alpha =1, lambda = grid)
predict(out, s=bessLam, type = "coefficients")[1:19,]
```

```
## (Intercept) (Intercept)  PrivateYes      Accept      Enroll   Top10perc
##   8.5923810   0.0000000   0.0000000   0.0000000   0.0000000   0.0000000
##   Top25perc F.Undergrad P.Undergrad    Outstate  Room.Board       Books
##   0.0000000   0.0000000   0.0000000   0.0000000   0.0000000   0.0000000
##    Personal         PhD    Terminal   S.F.Ratio perc.alumni      Expend
##   0.0000000   0.0000000   0.0000000   0.0000000   0.0000000   0.9991105
##   Grad.Rate
##   0.0000000
```

Significantly lower than the error of the ridge.

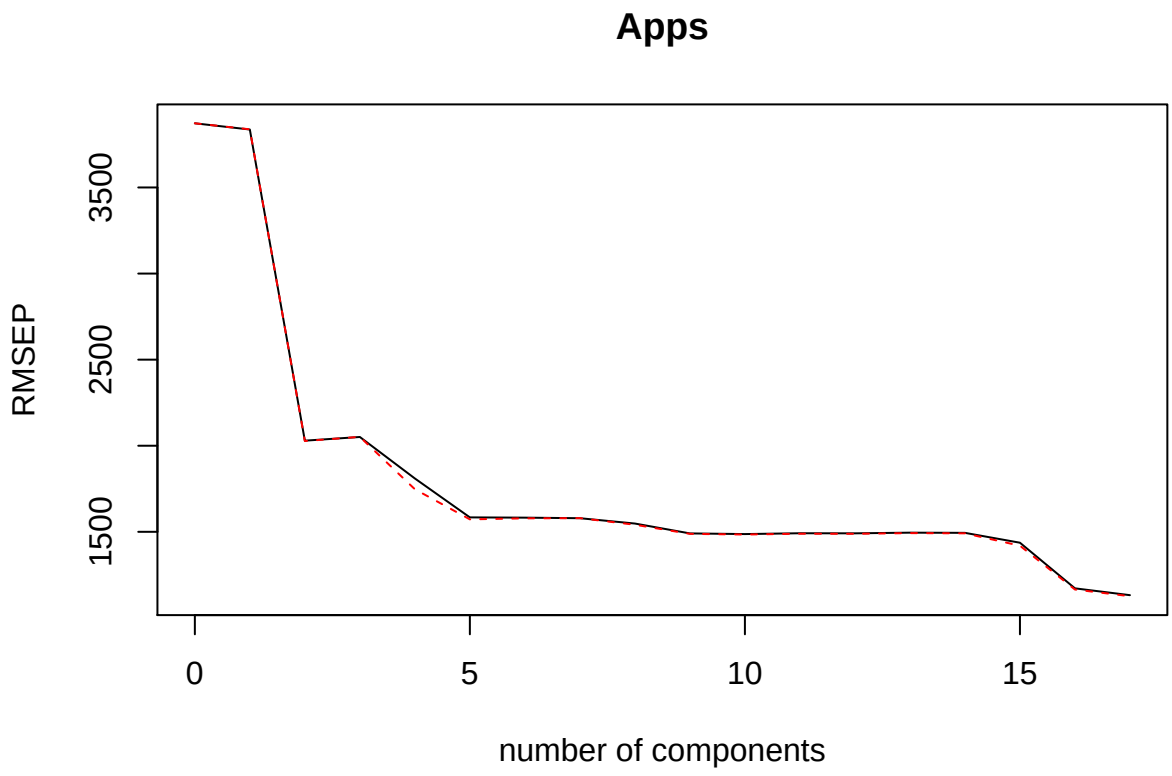
E

```
library(pls)
```

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

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

```
set.seed(18)
pcrM <- pcr(Apps~., data = college, scale = TRUE, validation = "CV")
validationplot(pcrM)
```

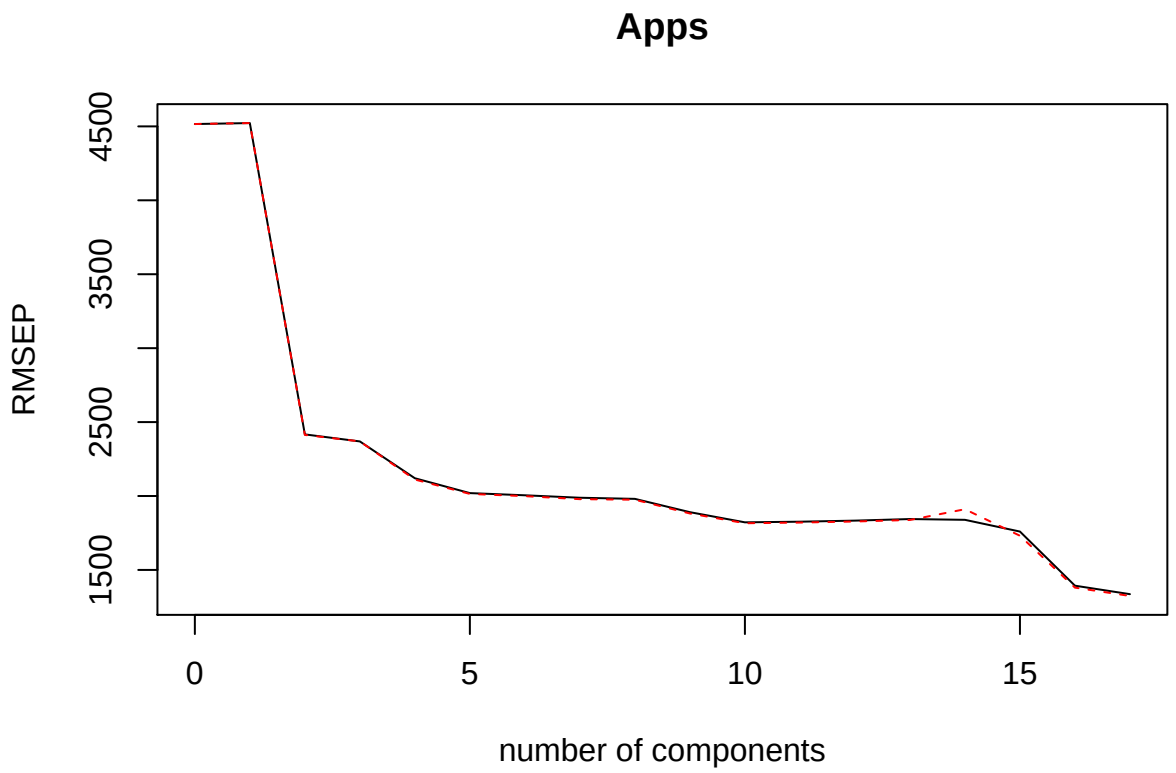



```
summary(pcrM)
```

```
## Data:    X dimension: 777 17
##  Y dimension: 777 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            3873     3837     2029     2051     1810     1584     1582
## adjCV         3873     3838     2027     2052     1747     1572     1578
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV        1579     1548     1491      1487      1492      1491      1495
## adjCV     1579     1541     1488      1484      1489      1488      1492
##        14 comps  15 comps  16 comps  17 comps
## CV         1494      1437      1172      1132
## adjCV      1491      1419      1165      1126
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X      31.670    57.30    64.30    69.90    75.39    80.38    83.99
## Apps    2.316    73.06    73.07    82.08    84.08    84.11    84.32
##       8 comps  9 comps  10 comps  11 comps  12 comps  13 comps  14 comps
## X       87.40    90.50     92.91     95.01     96.81      97.9     98.75
## Apps    85.18    85.88     86.06     86.06     86.10      86.1     86.13
##       15 comps  16 comps  17 comps
## X        99.36     99.84    100.00
## Apps     90.32     92.52     92.92
```

```
pcrTrain <- pcr(Apps~., data = testing, scale = TRUE, validation = "CV")
validationplot(pcrTrain)
```



```
summary(pcrTrain)
```

```
## Data:    X dimension: 389 17
##  Y dimension: 389 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            4516     4522     2417     2369     2120     2020     2005
## adjCV         4516     4522     2413     2369     2111     2014     1999
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV        1988     1981     1891      1822      1826      1833      1844
## adjCV     1979     1975     1882      1816      1820      1828      1838
##        14 comps  15 comps  16 comps  17 comps
## CV         1839      1760      1393      1336
## adjCV      1911      1731      1380      1324
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X      32.647    58.88    65.81    71.68    76.90    81.20    84.93
## Apps    1.233    72.65    73.91    79.65    81.02    81.95    82.23
##       8 comps  9 comps  10 comps  11 comps  12 comps  13 comps  14 comps
## X       88.47    91.28     93.46     95.32     96.93     98.00     98.72
## Apps    82.37    84.09     85.08     85.25     85.26     85.34     85.75
##       15 comps  16 comps  17 comps
## X        99.37     99.85    100.00
## Apps     91.49     93.07     93.68
```

```
p4 <- predict(pcrTrain, testMat, ncomp = 10)
e4 <- sqrt(mean((p4-testing[,"Apps"])^2))
e4
```

```
## [1] 1739.542
```

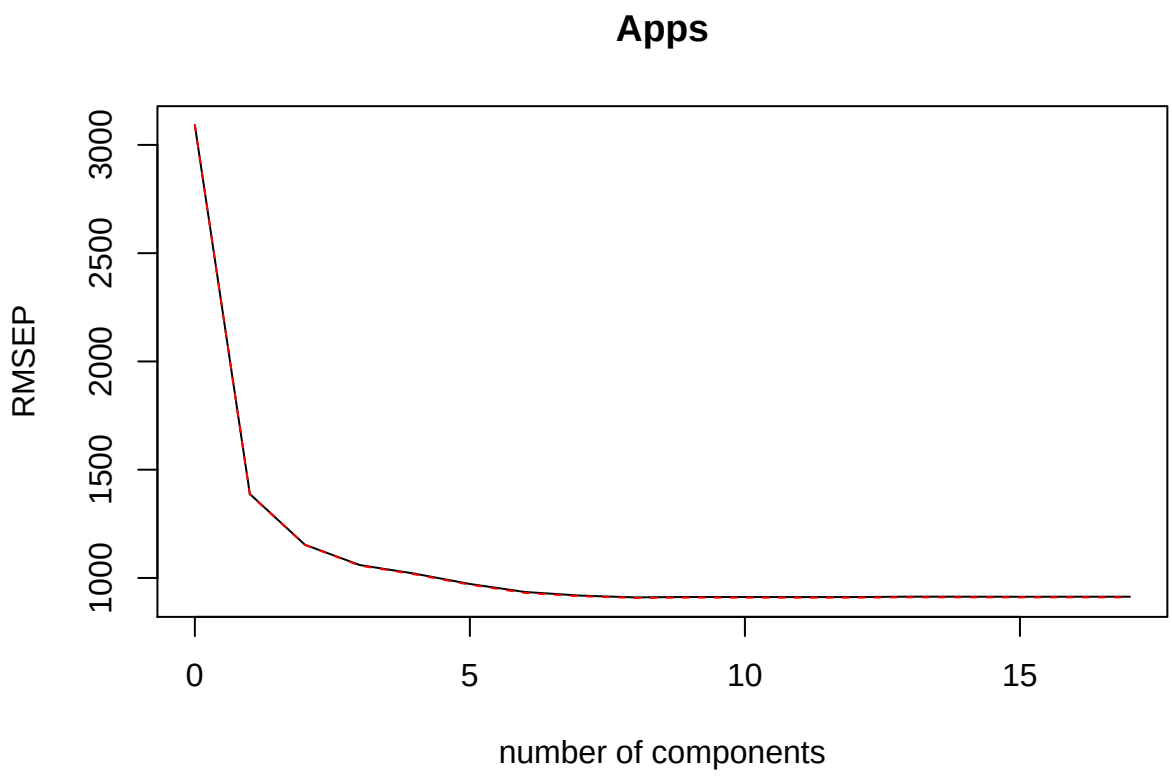
F

```
pslM <- plsr(Apps~., data = training, scale = TRUE, validation = "CV")
summary(pslM)
```

```
## Data:    X dimension: 388 17
##  Y dimension: 388 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            3092     1388     1154     1060     1020    972.2    935.2
```

```
## adjCV         3092     1386     1153     1057     1016    968.7    930.7
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       918.6    910.2    912.1     911.8     911.6     911.8     913.6
## adjCV    915.4    907.6    909.2     909.0     908.9     909.0     910.6
##       14 comps  15 comps  16 comps  17 comps
## CV       913.5     913.1     913.3     913.3
## adjCV    910.5     910.2     910.4     910.4
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X       26.10    41.87    62.40    67.28    71.10    74.46    78.07
## Apps    80.66    86.82    89.19    90.21    91.31    92.01    92.17
##       8 comps  9 comps  10 comps  11 comps  12 comps  13 comps  14 comps
## X       80.70    82.77     86.03     90.28     91.54     93.48     94.94
## Apps    92.23    92.24     92.25     92.25     92.26     92.26     92.26
##       15 comps  16 comps  17 comps
## X        96.74     97.90    100.00
## Apps     92.26     92.26     92.26
```

```
validationplot(pslM)
```

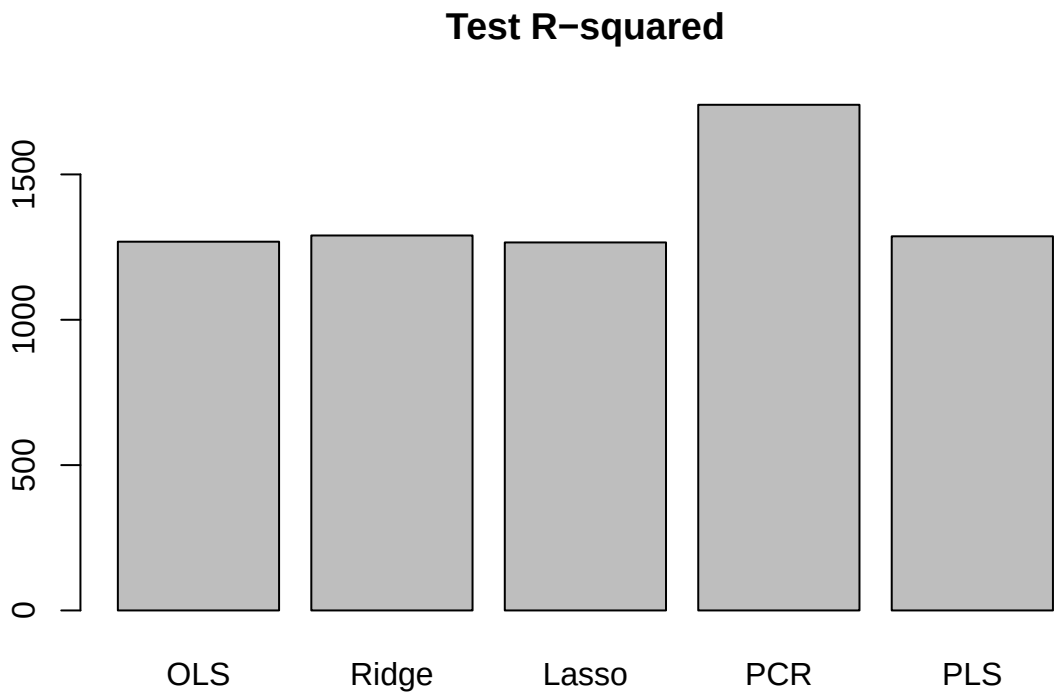



```
p5 <- predict(pslM, testMat, ncomp = 8)
e5 <- sqrt(mean((p5-testing[,"Apps"])^2))
e5
```

```
## [1] 1287.036
```

G

```
barplot(c(e1,e2,e3,e4,e5), names=c("OLS", "Ridge", "Lasso", "PCR", "PLS"), main="Test R-squared")
```



The only significantly different result came from PCR. It appears we can generally predict +/- 2400 applications.

6.11

A

```
rm(list = ls())
library(MASS)
library(leaps)
library(glmnet)
library(pls)
```

```
head(Boston)
```

```
##      crim zn indus chas   nox    rm  age    dis rad tax ptratio  black
## 1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3 396.90
## 2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8 396.90
```

```
## 3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8 392.83
## 4 0.03237  0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7 394.63
## 5 0.06905  0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7 396.90
## 6 0.02985  0  2.18    0 0.458 6.430 58.7 6.0622   3 222    18.7 394.12
##   lstat medv
## 1  4.98 24.0
## 2  9.14 21.6
## 3  4.03 34.7
## 4  2.94 33.4
## 5  5.33 36.2
## 6  5.21 28.7
```

```
set.seed(18)
train <- sample(c(TRUE,FALSE), nrow(Boston), rep = TRUE) #Building a training and test set
trainBos <- Boston[train,]
testBos <-Boston[!train,]
###
predict.regsubsets = function(object, newdata, id, ...) {
    form = as.formula(object$call[[2]])
    mat = model.matrix(form, newdata)
    coefi = coef(object, id = id)
    mat[, names(coefi)] %*% coefi
}
###
set.seed(18)
#Validation Set approach
subset <- regsubsets(crim~., trainBos, nvmax = 13)  #Best subset in the training data
subMat <- model.matrix(crim~.,data = testBos) #Build a model matrix for the testdata
errors <- rep(NA,13) #initialize a place to store errors
for (i in 1:13){
  coefi <- coef(subset, id =i)  #grab the coefficients from subset
  pr =subMat[, names(coefi)]%*%coefi #predictions
  errors[i]= sqrt(mean((testBos[1]-pr)^2)) #RMSE storage
}
which.min(errors) #find the minimum error index
```

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

```
e1 <- errors[which.min(errors)]
coef(subset,which.min(errors)) #get the coefficients of the minimum error model
```

```
## (Intercept)          zn       indus         nox         dis         rad
## 15.62751278  0.02968058 -0.08816059 -8.63266890 -0.71593143  0.43094394
##     ptratio       black       lstat        medv
## -0.22858982 -0.01332476  0.21108723 -0.09111055
```

```
bestSub <- regsubsets(crim~.,Boston, nvmax=13) #Running it with the entire data set
coef(bestSub, which.min(errors)) #This is the best one
```

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

```r
#Cross validation
k = 10 #Number of folds
folds = sample(rep(1:k, length = nrow(Boston))) #This is how we pull out samples from the fold
cv.errors = matrix(NA, k, 13) #empty matrix to store the results
for (i in 1:k) {
    best.fit = regsubsets(crim~., data= Boston[folds != i,], nvmax = 13)
    for (j in 1:13) {
        pred = predict(best.fit, Boston[folds == i, ], id = j)
        cv.errors[i, j] = mean((Boston$crim[folds == i] - pred)^2)
    }
}

rmse = sqrt(apply(cv.errors, 2, mean))
which.min(rmse)
```

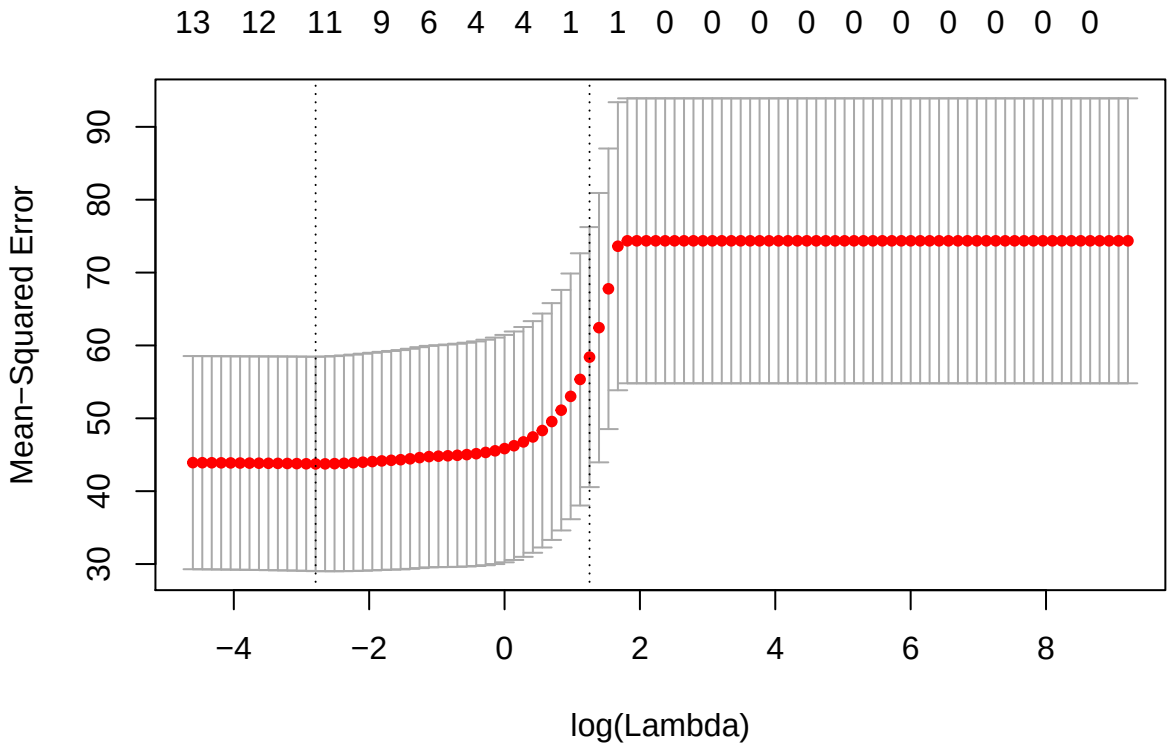
```
## [1] 9
```

```r
e2 <- rmse[which.min(rmse)]
```

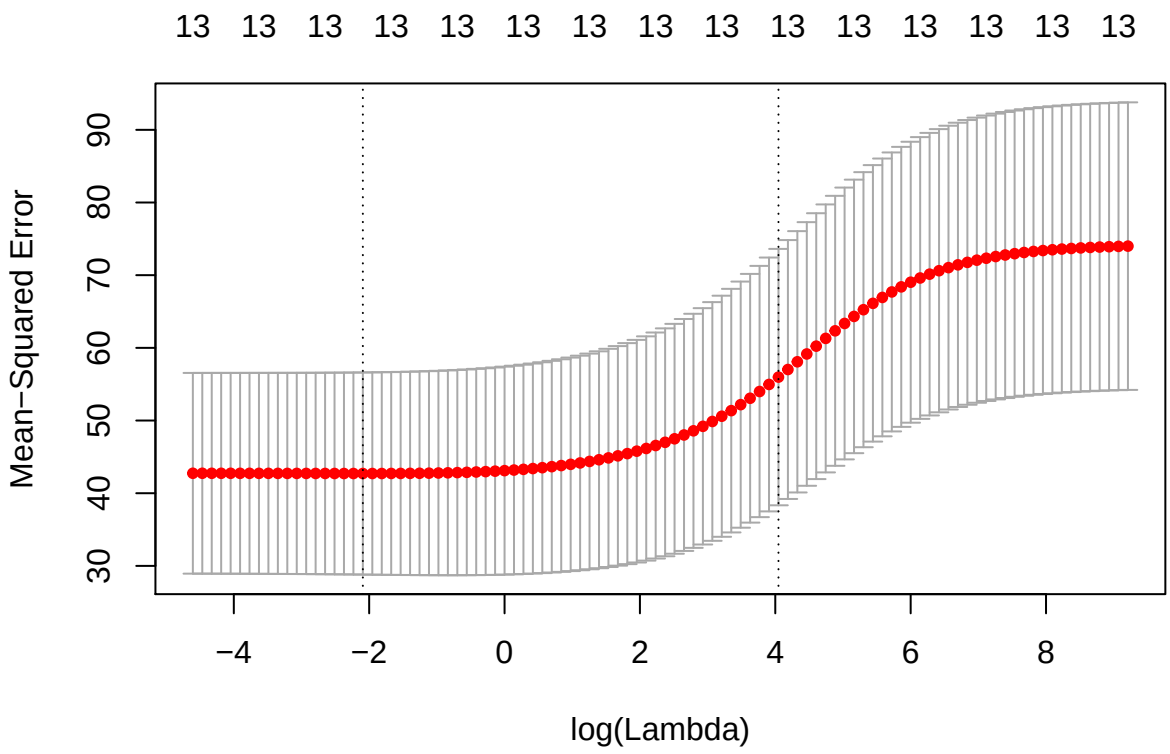
The best model determined through the validation set was a 6 variable model. The best model determined through cross validation was a 9 variable model. The 6 variable model had slightly better error than the 9 variable model.

```r
#RIDGE and LASSO
set.seed(18)
grid = 10^seq(4, -2, length=100)

x = model.matrix(crim~., data = Boston)
y = Boston$crim
cvLasso = cv.glmnet(x, y, type = "mse", alpha =1, lambda = grid)
plot(cvLasso)
```




```
e3 <- sqrt(cvLasso$cvm[cvLasso$lambda == cvLasso$lambda.min])
cvRidge = cv.glmnet(x, y, type.measure = "mse", alpha = 0, lambda = grid)
plot(cvRidge)
```




```
e4 <- sqrt(cvRidge$cvm[cvRidge$lambda == cvRidge$lambda.min])
```

Ridge is slightly better than Lasso.

```
set.seed(18)
pcr= pcr(crim~., data = Boston, scale=TRUE, validation = "CV")
summary(pcr) #min is at 13 variables
```

```
## Data:    X dimension: 506 13
##  Y dimension: 506 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           8.61    7.188    7.186    6.756    6.741    6.766    6.779
## adjCV        8.61    7.186    7.184    6.752    6.733    6.762    6.773
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       6.781    6.652    6.671     6.673     6.663      6.62     6.545
## adjCV    6.774    6.644    6.663     6.665     6.654      6.61     6.534
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X       47.70    60.36    69.67    76.45    82.99    88.00    91.14
```

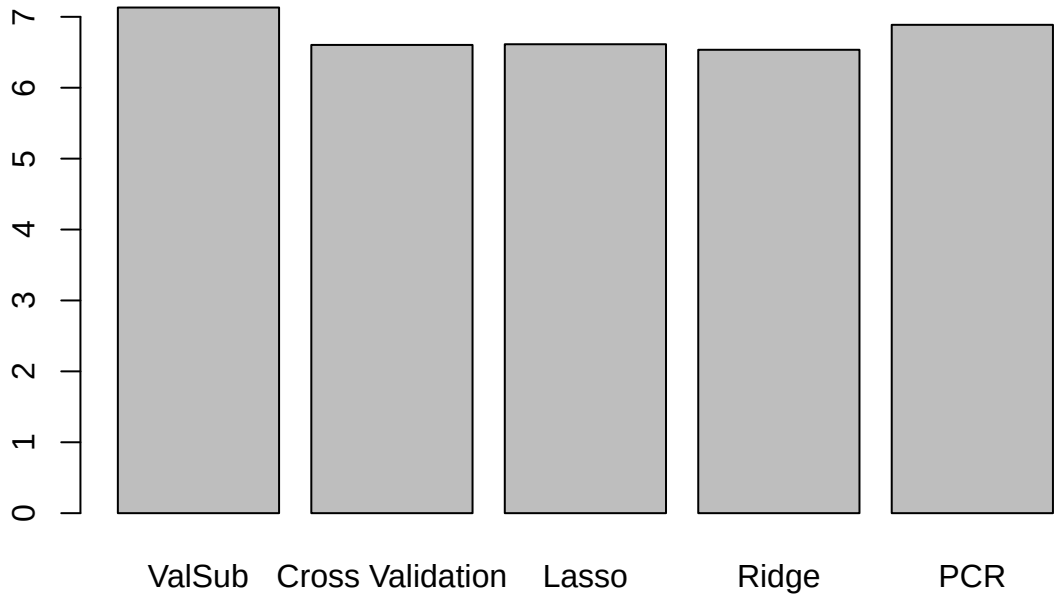
```
## crim    30.69    30.87    39.27    39.61    39.61    39.86    40.14
##       8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## X       93.45    95.40     97.04     98.46     99.52     100.0
## crim    42.47    42.55     42.78     43.04     44.13      45.4
```

```
pcrPred= predict(pcr, testBos, ncomp =13)
e5 <- sqrt(mean((pcrPred-testBos[,1])^2))
```

B

Error for Lasso, ridge and the nine variable model found through the cross validation are similar, but the validation subset is a simpler model, so that is the one I want to use.

```
barplot(c(e1,e2,e3,e4,e5), names = c("ValSub", "Cross Validation", "Lasso", "Ridge", "PCR"))
```



C

I only use nine variables because the ridge while slightly better utilizes all the components which is significantly more complicated with 13 components. Simpler is generally better, so I used the nine unit model.

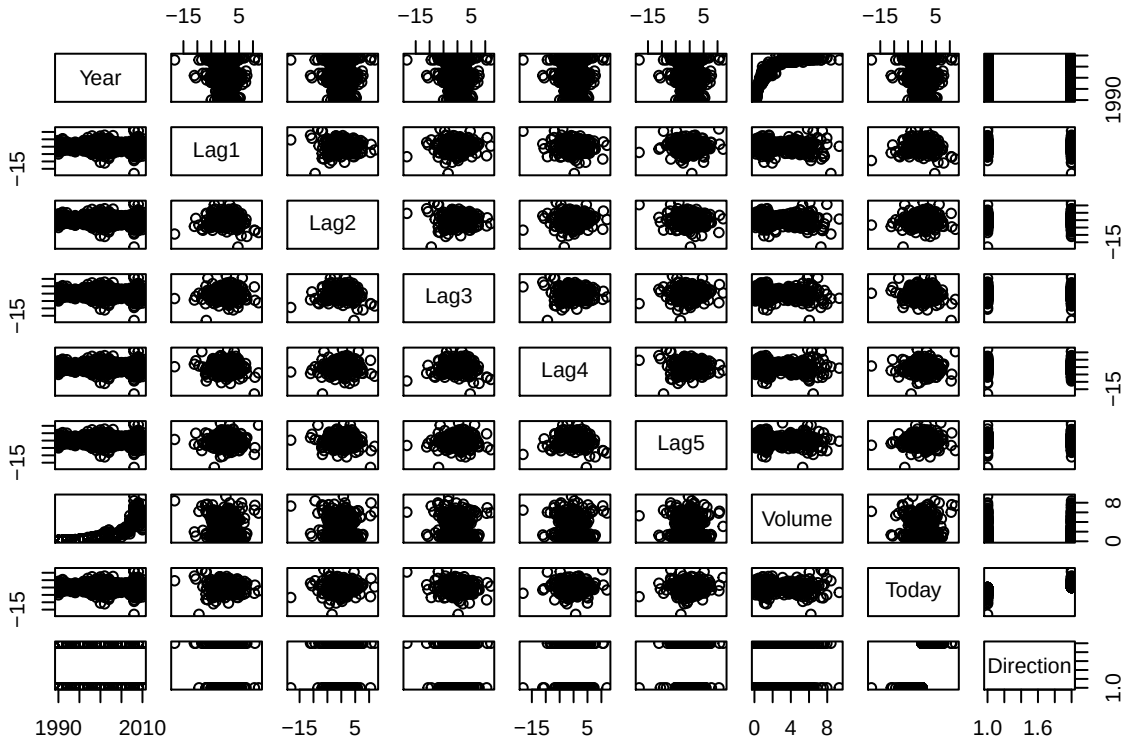
4.10

A

```
library(ISLR)
library(glmnet)
library(class)
head(Weekly)
```

```
##   Year   Lag1   Lag2   Lag3   Lag4   Lag5    Volume  Today Direction
## 1 1990  0.816  1.572 -3.936 -0.229 -3.484 0.1549760 -0.270      Down
## 2 1990 -0.270  0.816  1.572 -3.936 -0.229 0.1485740 -2.576      Down
## 3 1990 -2.576 -0.270  0.816  1.572 -3.936 0.1598375  3.514        Up
## 4 1990  3.514 -2.576 -0.270  0.816  1.572 0.1616300  0.712        Up
## 5 1990  0.712  3.514 -2.576 -0.270  0.816 0.1537280  1.178        Up
## 6 1990  1.178  0.712  3.514 -2.576 -0.270 0.1544440 -1.372      Down
```

```
pairs(Weekly)
```



The only variables that appear to have any correlation are Year and Volume. All the lag variables produce what appears to be a random blob with each other and a straight line (slope of close to zero) with year.

B

Making a logistic model that looks for the Direction odds with the five lag variables and the volume.

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

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
##     Volume, family = binomial, data = Weekly)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6949  -1.2565   0.9913   1.0849   1.4579
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.26686    0.08593   3.106   0.0019 **
## Lag1        -0.04127    0.02641  -1.563   0.1181
## Lag2         0.05844    0.02686   2.175   0.0296 *
## Lag3        -0.01606    0.02666  -0.602   0.5469
## Lag4        -0.02779    0.02646  -1.050   0.2937
## Lag5        -0.01447    0.02638  -0.549   0.5833
## Volume      -0.02274    0.03690  -0.616   0.5377
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1496.2  on 1088  degrees of freedom
## Residual deviance: 1486.4  on 1082  degrees of freedom
## AIC: 1500.4
##
## Number of Fisher Scoring iterations: 4
```

The only variable of any significance appears to be Lag2. The positive coefficient says that if the market went up before, it's more likely to go up again.

C

```
probs <- predict(glm.fit, type = "response")
glm.predict <- rep("Down", length(probs))
glm.predict[probs > 0.5] = "Up"
t1 <- table(glm.predict, Weekly$Direction)
sums = 0
for (i in 1:nrow(t1)){
  sums = sums+ t1[i,i]
}
sums/length(probs)
```

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

Generally when we predict "Up", we're guessing correctly, but we are guessing "Up" too much and a lot of "Downs" are being predicted as ups. We're barely guessing half of them correctly.

D

```
training <- Weekly[Weekly$Year<2009,]
testing <- Weekly[Weekly$Year>2008,]
train.fit <- glm(Direction~Lag2, data = training, family = binomial)
probs <- predict(train.fit, testing,type = "response")
glm.predict <- rep("Down",length(probs))
glm.predict[probs>0.5]= "Up"
t2 <- table(glm.predict, testing$Direction)
t2
```

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

```
sums = 0
for (i in 1:nrow(t2)){
  sums = sums+t2[i,i]
}
sums/length(probs)
```

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

Slightly better, but still pretty bad and failing the same way.

G

```
set.seed(18)
kn = 1
train.X = as.matrix(Weekly$Lag2[Weekly$Year<2009])
test.X = as.matrix(Weekly$Lag2[Weekly$Year>2008])
train.Direction = Weekly$Direction[Weekly$Year<2009]
knn = knn(train.X, test.X, train.Direction, k = kn)
t3 <- table(knn, testing$Direction)
t3
```

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

```
sums = 0
for (i in 1:nrow(t3)){
  sums = sums+t3[i,i]
}
sums/length(probs)
```

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

H

The Logistic equation was better than our KNN, but at the same time, we only used k=1 which is terrible.

I

```
set.seed(18)

knn = knn(train.X, test.X, train.Direction, k = 10)
t4 <- table(knn, testing$Direction)
sums = 0
for (i in 1:nrow(t4)) {
  sums = sums+t4[i,i]
  }
sums/length(probs)
```

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

```
knn = knn(train.X, test.X, train.Direction, k = 5)
t5 <- table(knn, testing$Direction)
sums = 0
for (i in 1:nrow(t5)) {
  sums = sums+t5[i,i]
  }
sums/length(probs)
```

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

```
knn = knn(train.X, test.X, train.Direction, k = 15)
t5 <- table(knn, testing$Direction)
sums = 0
for (i in 1:nrow(t5)) {
  sums = sums+t5[i,i]
  }
sums/length(probs)
```

```
## [1] 0.5865385
```

```
glm.fit =glm(Direction~Lag1:Lag2+Lag2, data=Weekly, family = binomial)
summary(glm.fit)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1:Lag2 + Lag2, family = binomial,
##     data = Weekly)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.468  -1.268   1.012   1.085   1.572
```

```
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) 0.216626   0.061299   3.534 0.000409 ***
## Lag2        0.061678   0.026570   2.321 0.020269 *
## Lag1:Lag2   0.004965   0.006245   0.795 0.426593
## ---
## 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: 1489.8  on 1086  degrees of freedom
## AIC: 1495.8
##
## Number of Fisher Scoring iterations: 4
```

```
probs <- predict(glm.fit, type = "response")
glm.predict <- rep("Down", length(probs))
glm.predict[probs > 0.5] = "Up"
t6 <- table(glm.predict, Weekly$Direction)
sums = 0
for (i in 1:nrow(t6)){
  sums = sums+ t1[i,i]
}
sums/length(probs)
```

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

8.8

```
rm(list = ls())
library(ISLR)
library(rpart)
library(rpart.plot)
library(randomForest)
```

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

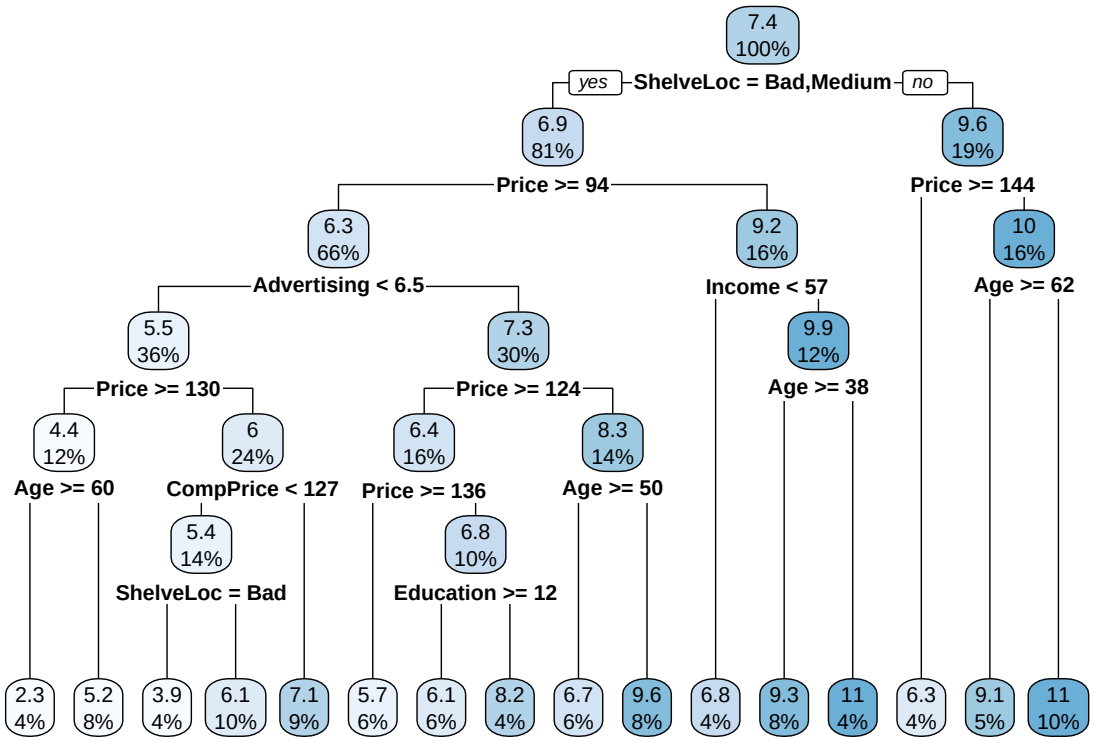
```
## Type rfNews() to see new features/changes/bug fixes.
```

A

```
set.seed(18)
mask = sample(dim(Carseats)[1], dim(Carseats)[1]/2)
test = Carseats[mask, ]
train = Carseats[-mask, ]
```

B

```
#tree
set.seed(18)
m1 <- rpart(Sales~., data = train, method = "anova")
rpart.plot(m1)
```




```
p1 <- predict(m1, test)
sqrt(mean((p1-test["Sales"])^2))
```

```
## [1] 2.12449
```

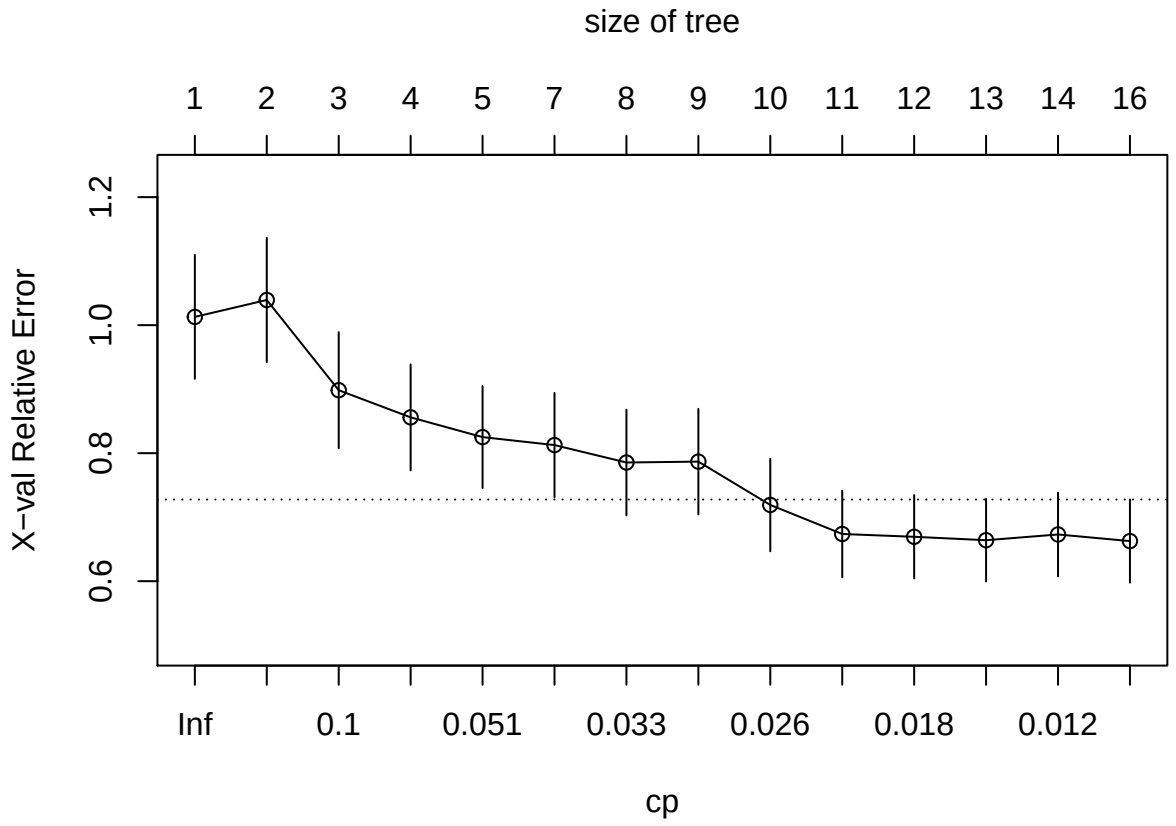
C

```
#Pruning in rpart
set.seed(18)
printcp(m1) #A tree with 11 terminal nodes has the lowest error
```

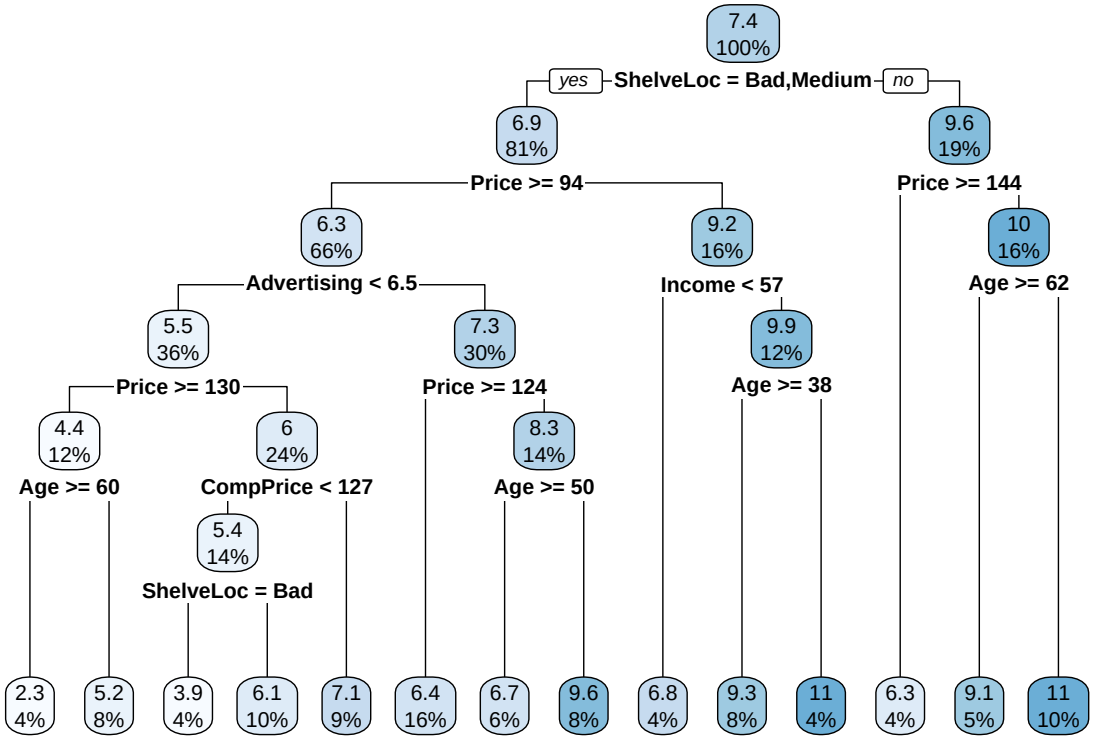
```
##
## Regression tree:
## rpart(formula = Sales ~ ., data = train, method = "anova")
##
```

```
## Variables actually used in tree construction:
## [1] Advertising Age         CompPrice   Education   Income      Price
## [7] ShelveLoc
##
## Root node error: 1445.4/200 = 7.2269
##
## n= 200
##
##          CP nsplit rel error  xerror     xstd
## 1  0.163850      0   1.00000 1.01290 0.096797
## 2  0.143029      1   0.83615 1.03934 0.097144
## 3  0.072858      2   0.69312 0.89836 0.090749
## 4  0.067247      3   0.62026 0.85597 0.082720
## 5  0.038250      4   0.55302 0.82510 0.079689
## 6  0.034496      6   0.47652 0.81266 0.081554
## 7  0.030879      7   0.44202 0.78544 0.082505
## 8  0.030150      8   0.41114 0.78682 0.082512
## 9  0.022541      9   0.38099 0.71912 0.072380
## 10 0.020030     10   0.35845 0.67376 0.067756
## 11 0.016764     11   0.33842 0.66939 0.065128
## 12 0.014796     12   0.32165 0.66405 0.064502
## 13 0.010492     13   0.30686 0.67301 0.065435
## 14 0.010000     15   0.28587 0.66266 0.064947
```

```
plotcp(m1)
```



```
m2 <- prune(m1,cp=.011608) #Couldn't quite figure out how to automatically grab this
rpart.plot(m2)
```




```
p2 <- predict(m2, test)
sqrt(mean((p2-test["Sales"])^2))
```

```
## [1] 2.154735
```

Pruning made the test error slightly worse.

D

```
#Bagging
set.seed(18)
rf <- randomForest(Sales~., data = train, mtry= 10,importance = TRUE)

p3 <- predict(rf, test)
sqrt(mean((p3-test["Sales"])^2))
```

```
## [1] 1.701642
```

```
importance(rf)
```

```
##                %IncMSE IncNodePurity
## CompPrice   17.5940839    119.983520
## Income       1.0284135     73.421312
## Advertising 14.8285764    139.347948
## Population  -0.3858411     62.477575
## Price       52.5647339    462.073775
## ShelveLoc   44.3112855    286.840034
## Age         22.6396851    194.345577
## Education   -0.0592421     39.024541
## Urban       -0.3614932     10.898609
## US           3.7559758      8.987511
```

We see that the most important variables are Price, ShelveLoc, and Age.

E

```
#Random Forest
set.seed(18)
rf2 <- randomForest(Sales~., data = train, importance = TRUE)
p4 <- predict(rf2,test)
sqrt(mean((p4-test["Sales"])^2))
```

```
## [1] 1.879826
```

```
importance(rf2)
```

```
##                %IncMSE IncNodePurity
## CompPrice    7.6011369     125.21019
## Income       2.5034603     113.22793
## Advertising 10.8223702     139.32976
## Population  -0.0663489     101.93688
## Price       36.3883615     357.48074
## ShelveLoc   30.7419842     224.11301
## Age         17.9390404     193.40859
## Education    1.4656259      58.29335
## Urban        0.5056667      15.14273
## US           3.5438426      22.73429
```

As we go down, RMSE goes up. The same variables that were important in the bagging part, are still the most important, but less so comparatively

8.11

A

```
rm(list=ls())
library(ISLR)
library(gbm)
```

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

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

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

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

```
## Loaded gbm 2.1.1
```

```
library(class)
library(glmnet)
Caravan$Purchase = ifelse(Caravan$Purchase == "Yes", 1,0)
train = Caravan[1:1000,]
test=Caravan[1001:nrow(Caravan),]
```

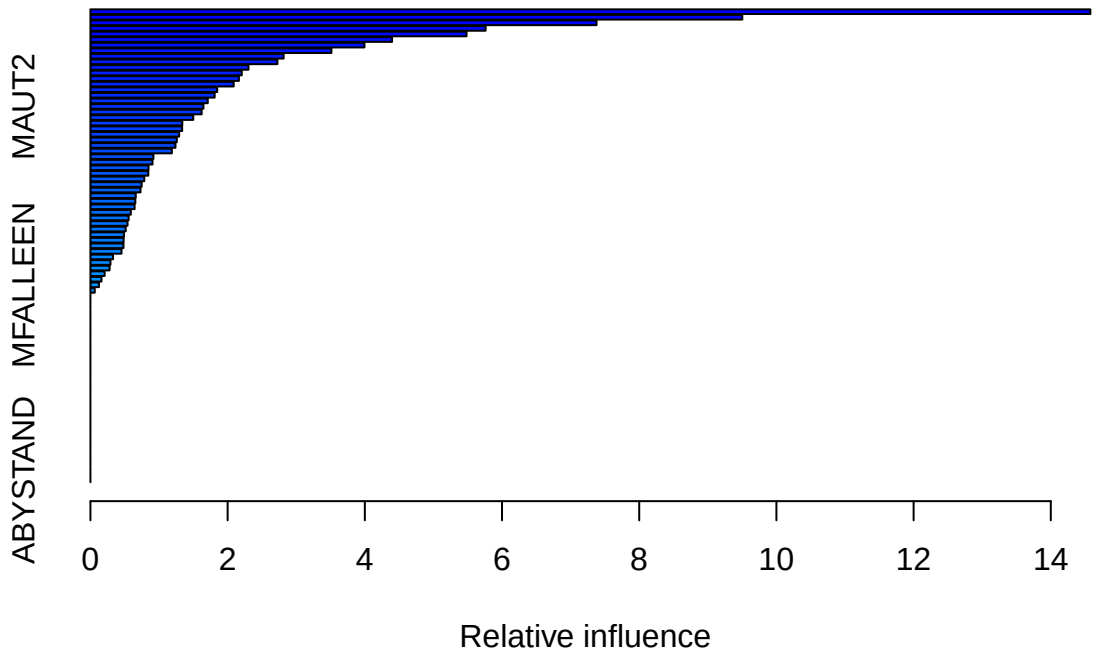
B

```
set.seed(18)
boost <- gbm(Purchase~., data = train, distribution = "bernoulli", 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)
```




```
##                var     rel.inf
## PPERSAUT PPERSAUT 14.57675709
## MKOOPKLA MKOOPKLA  9.50265357
## MOPLHOOG MOPLHOOG  7.37799005
## MBERMIDD MBERMIDD  5.76076392
## PBRAND     PBRAND  5.48277194
## ABRAND     ABRAND  4.39700295
## MGODGE     MGODGE  3.99434675
## MINK3045 MINK3045  3.51290076
## MAUT1       MAUT1  2.81629454
## MOSTYPE   MOSTYPE  2.72570844
## MGODPR     MGODPR  2.30408805
## MBERARBG MBERARBG  2.20649222
## MSKA         MSKA  2.16442017
## PWAPART   PWAPART  2.08836336
## MGODOV     MGODOV  1.84827998
## MBERHOOG MBERHOOG  1.81102959
## MSKC         MSKC  1.71054855
## MAUT2       MAUT2  1.64700459
## MINKGEM   MINKGEM  1.62159662
## PBYSTAND PBYSTAND  1.49907725
## MRELGE     MRELGE  1.33901832
## MAUT0       MAUT0  1.33676679
## MHHUUR     MHHUUR  1.29290755
## MFWEKIND MFWEKIND  1.25905508
## MSKB1       MSKB1  1.23987090
```

```
## MINK7512 MINK7512  1.18733937
## MRELOV     MRELOV  0.91827155
## MFGEKIND MFGEKIND  0.90489988
## MINK4575 MINK4575  0.84760079
## MOSHOOFD MOSHOOFD  0.84361448
## MSKD         MSKD  0.78456685
## MOPLMIDD MOPLMIDD  0.74653323
## MGODRK     MGODRK  0.72951628
## APERSAUT APERSAUT  0.66039728
## MHKOOP     MHKOOP  0.65438724
## MGEMOMV   MGEMOMV  0.64372664
## PMOTSCO   PMOTSCO  0.58857228
## MINK123M MINK123M  0.55765109
## MBERARBO MBERARBO  0.54069629
## MSKB2       MSKB2  0.51415079
## MINKM30   MINKM30  0.48863988
## PLEVEN     PLEVEN  0.48404853
## MZFONDS   MZFONDS  0.48117584
## MGEMLEEF MGEMLEEF  0.45320595
## MBERBOER MBERBOER  0.32956600
## MBERZELF MBERZELF  0.29202110
## MRELSA     MRELSA  0.27929478
## MZPART     MZPART  0.20701606
## MOPLLAAG MOPLLAAG  0.16082307
## MFALLEEN MFALLEEN  0.12402699
## MAANTHUI MAANTHUI  0.06254873
## 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
```

This lists the most important variables with PPERSAUT, MKOOPKLA, and MOPLHOOG being the most important. It appears almost half the variables have no importance at all.

C

```
set.seed(18)
prob= predict(boost, test, n.trees = 1000, type = "response")
p1 = ifelse(prob > 0.2, 1, 0)
table(test$Purchase, p1)
```

```
##    p1
##        0    1
##   0 4414  119
##   1  256   33
```

```
##Prediction
33/(119+33)
```

```
## [1] 0.2171053
```

About 22% of the time when we predict someone will buy, they actually buy.

```
### KNN
set.seed(18)
train.X = as.matrix(Caravan[1:1000,])
test.X = as.matrix(Caravan[1001:nrow(Caravan),])
train.Y = Caravan[1:1000, "Purchase"]
kn =5
knn = knn(train.X, test.X, train.Y, k = kn)
t2 <- table(knn, test$Purchase)
t2
```

```
##
## knn    0    1
##   0 4506  285
##   1   27    4
```

KNN isn't working well. With a k=20, it wouldn't predict any Yes's. With k=5 it's still producing very poor numbers. Intuitively this makes sense since if it has very few Yes's in general, most of its neighbors are likely No's regardless. So having a low K may help, but in the end, it'd be very difficult to predict using KNN.

```
set.seed(18)
log <- glm(Purchase~.,data= train, family = "binomial")
```

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

```
logProb <- predict(log, 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
```

```
p2 <- ifelse(logProb >.2, 1,0)
t3 <- table(p2,test$Purchase)
t3
```

```
##
## p2     0    1
##   0 4183  231
##   1  350   58
```

```
58/(231+58)
```

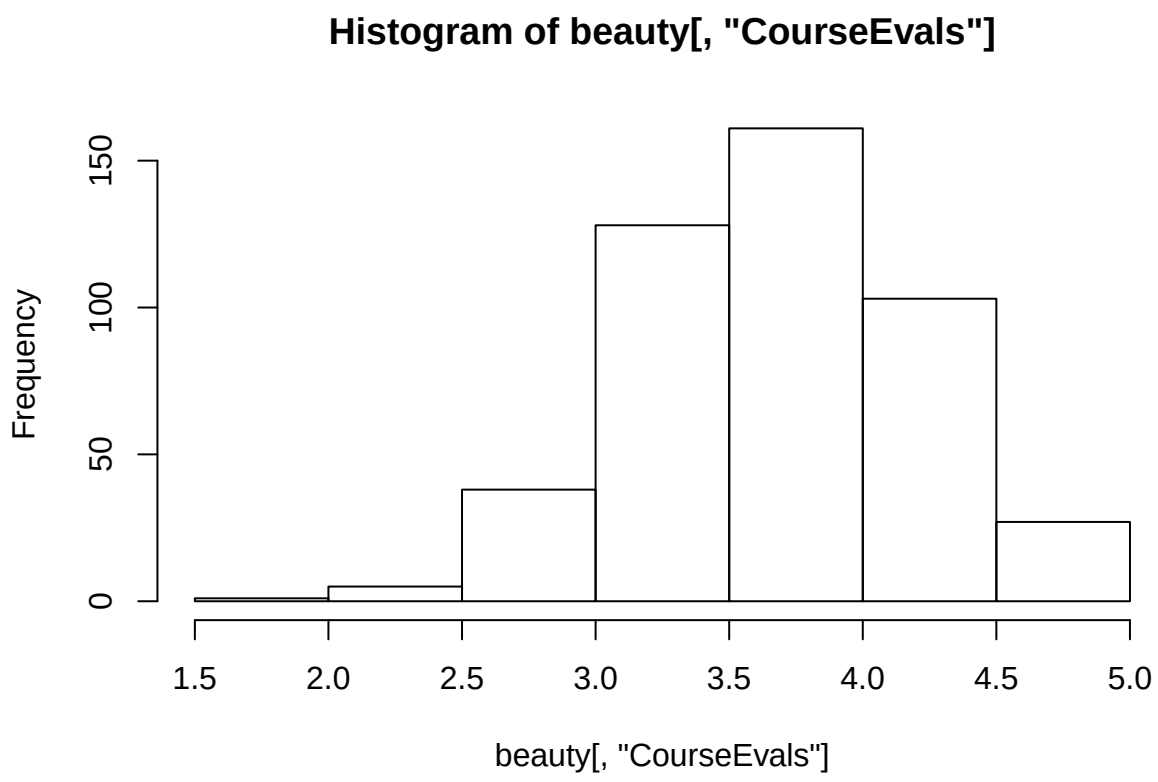
```
## [1] 0.200692
```

For the logistic model it comes close to matching the boosting tree, but not quite at 20% accuracy.

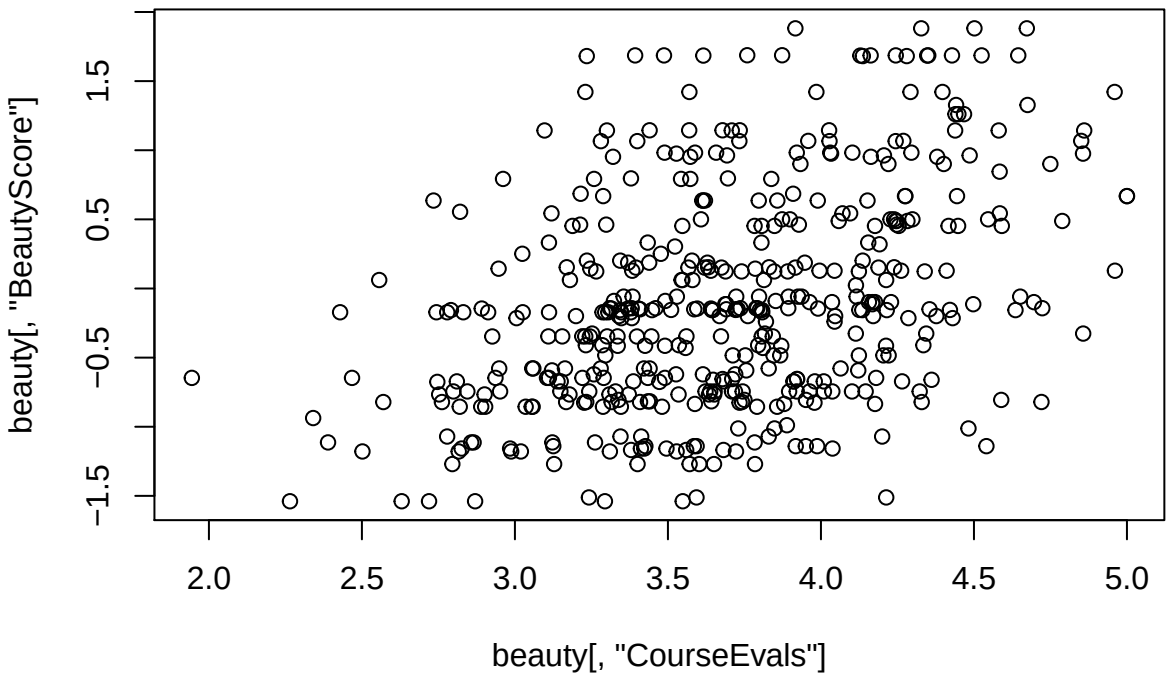
Beauty Pays

1

```
rm(list=ls())
beauty <- read.csv("BeautyData.csv")
library(rpart)
library(randomForest)
##Look at the data first
hist(beauty[,"CourseEvals"])
```




```
plot(beauty[,"CourseEvals"],beauty[,"BeautyScore"])
```




```
set.seed(18)
mask = sample(dim(beauty)[1], dim(beauty)[1]/2)
test = beauty[mask, ]
train = beauty[-mask, ]
```

Random Forest

```
set.seed(18)
rf <- randomForest(CourseEvals~.,data=train, importance= TRUE, mtry=2, n.trees=500)
prf <- predict(rf, test)
sqrt(mean((prf-test["CourseEvals"])^2))
```

```
## [1] 0.4718517
```

```
importance(rf)
```

```
##                %IncMSE IncNodePurity
## BeautyScore 38.597993     16.026367
## female      29.539286      3.798625
## lower       30.132268      4.199279
## nonenglish  14.260685      1.040886
## tenuretrack  2.438957      1.177041
```

RMSE for Random Forest = .471 Beauty clearly the most important variable in node purity. Random Forest isn't the best at understanding what's happening, so I'll now run a linear regression.

Linear Regression

```
lr <- lm(CourseEvals~., data=train)
plr <- predict(lr,test)
sqrt(mean((plr-test["CourseEvals"])^2))
```

```
## [1] 0.4459505
```

```
summary(lr)
```

```
##
## Call:
## lm(formula = CourseEvals ~ ., data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.10671 -0.28343 -0.00742  0.28933  1.03472
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.06192    0.07107  57.151  < 2e-16 ***
## BeautyScore  0.28607    0.03571   8.010 6.05e-14 ***
## female      -0.31044    0.05574  -5.569 7.22e-08 ***
## lower       -0.30691    0.05697  -5.387 1.79e-07 ***
## nonenglish  -0.27576    0.10523  -2.621  0.00937 **
## tenuretrack -0.10578    0.06733  -1.571  0.11756
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4093 on 226 degrees of freedom
## Multiple R-squared:  0.338,  Adjusted R-squared:  0.3233
## F-statistic: 23.08 on 5 and 226 DF,  p-value: < 2.2e-16
```

Of the variables that were significant, the only positive one was beauty. It isn't surprising that a non-native English-speaker is has a negative effect since it might make them more difficult to learn from. It is strange that being female actually has a greater negative effect than being a non-native speaker. That should probably be looked at. But overall, this along with my random forest shows that beauty is a significant factor and results in a positive effect

2

There is no way to actually measure someone on a scale of how good they are at their job. Dr. Hamermesh is saying that while his data suggests that beauty has an effect on these course evaluations, he can say nothing with certainty. It is possible that being better looking actually does make you a better teacher (maybe beautiful people somehow communicate better) or maybe they're not actually better, but students respond differently (students who are attracted to you going to your office hours more or sitting closer to the front). Or perhaps since most attractive professors are younger, they have more energy and enthusiasm. It may behoove us to look at age since at least this is something we can measure.

Housing Price Structure

1

```
rm(list=ls())
set.seed(18)
library(caret)
```

```
## Loading required package: ggplot2
```

```
##
## Attaching package: 'ggplot2'
```

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

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

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

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

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

Trying a linear regression.

```
set.seed(18)
city$Brick.f <- as.factor(city$Brick)
city$Nbhd.f <- as.factor(city$Nbhd)
m1 <- lm(Price~.-Brick+Brick.f-Nbhd+Nbhd.f, data = city)
varImp(m1, scale = TRUE)
```

```
##              Overall
## Home       0.4512701
## Offers     7.5656285
## SqFt       9.0509699
## Bedrooms   2.5505767
## Bathrooms  3.7374836
## Brick.fYes 8.7066231
## Nbhd.f2    0.7106765
## Nbhd.f3    6.4654829
```

```r
summary(m1)
```

```
##
## Call:
## lm(formula = Price ~ . - Brick + Brick.f - Nbhd + Nbhd.f, data = city)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -27897.8  -6074.8    -48.7   5551.8  27536.4
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2037.726   8911.501   0.229 0.819524
## Home          -11.456     25.387  -0.451 0.652616
## Offers      -8350.128   1103.693  -7.566 8.96e-12 ***
## SqFt           53.634      5.926   9.051 3.30e-15 ***
## Bedrooms     4136.461   1621.775   2.551 0.012023 *
## Bathrooms    7975.157   2133.831   3.737 0.000287 ***
## Brick.fYes  17313.540   1988.548   8.707 2.12e-14 ***
## Nbhd.f2     -1729.613   2433.756  -0.711 0.478675
## Nbhd.f3     20534.706   3176.051   6.465 2.33e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10050 on 119 degrees of freedom
## Multiple R-squared:  0.8688, Adjusted R-squared:   0.86
## F-statistic: 98.54 on 8 and 119 DF,  p-value: < 2.2e-16
```

```r
confint(m1)
```

```
##                    2.5 %      97.5 %
## (Intercept) -15607.93640 19683.38913
## Home           -61.72532    38.81249
## Offers      -10535.54898 -6164.70615
## SqFt            41.90048    65.36778
## Bedrooms       925.18504  7347.73765
## Bathrooms     3749.95923 12200.35499
## Brick.fYes   13376.01617 21251.06453
## Nbhd.f2      -6548.69199  3089.46637
## Nbhd.f3      14245.80700 26823.60505
```

We can see that brick is indeed a significant variable and as shown by the confidence interval, it is almost certainly positive. We have enough evidence here to say that people place a premium on brick.

2

We can see the same thing with neighborhood 3. It appears that people will pay $14000-$26000 more for a house in N3.

3

```
m2 <- lm(Price~.-Brick+Brick.f-Nbhd+Nbhd.f+Brick.f*Nbhd.f, data =city)

summary(m2)
```

```
##
## Call:
## lm(formula = Price ~ . - Brick + Brick.f - Nbhd + Nbhd.f + Brick.f *
##     Nbhd.f, data = city)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -27843.9  -5544.3   -526.9   4167.3  28237.8
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)          3593.645   8860.065   0.406  0.68578
## Home                  -12.410     24.975  -0.497  0.62020
## Offers              -8470.621   1086.489  -7.796 2.91e-12 ***
## SqFt                   54.427      5.866   9.278 1.10e-15 ***
## Bedrooms             4660.752   1608.651   2.897  0.00449 **
## Bathrooms            6554.909   2176.681   3.011  0.00319 **
## Brick.fYes          12033.113   4097.033   2.937  0.00399 **
## Nbhd.f2             -1527.046   2721.268  -0.561  0.57577
## Nbhd.f3             16807.264   3466.191   4.849 3.86e-06 ***
## Brick.fYes:Nbhd.f2   2781.540   5090.237   0.546  0.58580
## Brick.fYes:Nbhd.f3  12019.217   5360.949   2.242  0.02685 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9879 on 117 degrees of freedom
## Multiple R-squared:  0.8755, Adjusted R-squared:  0.8648
## F-statistic: 82.24 on 10 and 117 DF,  p-value: < 2.2e-16
```

```
confint(m2)
```

```
##                            2.5 %      97.5 %
## (Intercept)        -13953.24931 21140.53952
## Home                   -61.87071    37.05147
## Offers             -10622.35622 -6318.88665
## SqFt                   42.80853    66.04483
## Bedrooms             1474.90283  7846.60052
## Bathrooms            2244.10715 10865.71176
## Brick.fYes           3919.15329 20147.07297
## Nbhd.f2             -6916.37571  3862.28303
## Nbhd.f3              9942.65540 23671.87317
## Brick.fYes:Nbhd.f2  -7299.40776 12862.48835
## Brick.fYes:Nbhd.f3   1402.13887 22636.29584
```

It appears that people will place a premium on a brick house in neighborhood 3. The confidence interval for Brick&N3 is very wide, but does not cross 0.

4

N2 has been insignificant (95% conf int cross 0). It is reasonable to just ignore it and combine N1 and N2 into one level since only being in N3 seems to have any significant effect on the prediction.

What causes what?

1

Because crime and number of police can influence each other. If we're trying to see if more police reduce crime, we have to also think, wait, but if there's more crime, a city may employ more police. So if we ran a regression on that, it may appear that more police cause more crime (or at least correlate with more crime).

2 & 3

The UPENN researchers needed something that changed the number of active police to change that was irrelevant to crime. Washington DC has terror alert levels where they'll station more police when risk of a terrorist attack goes up. Since this appears to be unrelated to crime (although maybe criminals hide when terrorists may go up), this was what they used. And indeed on high alert days (days when police numbers were up), crime went down (this is shown in the first column). But then the researchers thought, what if tourists and other people stay home when the terror alert is high (i.e. there are fewer victims available). This could cause crime to go down since there are fewer people to rob. They decided to measure the victim availability with the number of people on the METRO. Column 2 shows that as ridership goes up, crime still goes down with higher alert levels.

4

Here they decided to look at the districts of DC. Whatever district 1 is, on high alert, crime seriously goes down. Other districts do not show this significantly. It is possible district 1 contains things such as the White House which are much more likely terrorist targets. The would mean there are likely even more police in those areas. Other districts may include residential areas where police activity may not go up as much and therefore there is a much weaker if at all reduction in crime.