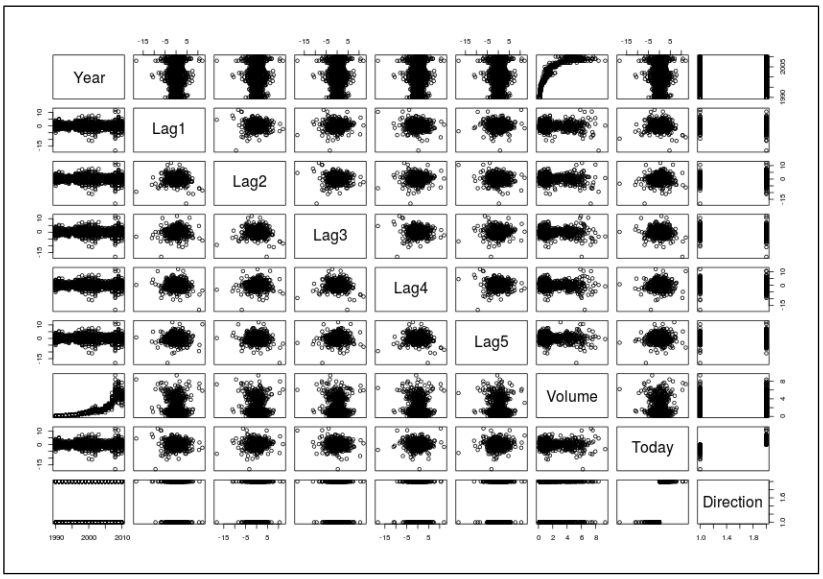
Patrick Rogan
UNI: psr2125

STAT4702 Homework 9

4.7.10

a. Looking at the pairs plot and correlation matrix (including factor variable "Direction") Year and Volume appear to have a strong positive correlation while the other variables do appear to be mostly uncorrelated.



	Year	Lag1	Lag2	Lag3	Lag4	Lag5	Volume	Today
Year	1.000000	-0.0322893	-0.033390	-0.030006	-0.0311279	-0.0305191	0.841942	-0.0324599
Lag1	-0.032289	1.0000000	-0.074853	0.058636	-0.0712739	-0.0081831	-0.064951	-0.0750318
Lag2	-0.033390	-0.0748531	1.000000	-0.075721	0.0583815	-0.0724995	-0.085513	0.0591667
Lag3	-0.030006	0.0586357	-0.075721	1.000000	-0.0753959	0.0606572	-0.069288	-0.0712436
Lag4	-0.031128	-0.0712739	0.058382	-0.075396	1.0000000	-0.0756750	-0.061075	-0.0078259
Lag5	-0.030519	-0.0081831	-0.072499	0.060657	-0.0756750	1.0000000	-0.058517	0.0110127
Volume	0.841942	-0.0649513	-0.085513	-0.069288	-0.0610746	-0.0585174	1.000000	-0.0330778
Today	-0.032460	-0.0750318	0.059167	-0.071244	-0.0078259	0.0110127	-0.033078	1.0000000

b. Fitting with glm, only Lag2 appears to be statistically significant.

c. The overall fraction of correct predictions is 0.56107. The confusion matrix correctly identifies the vast majority of up days while incorrectly identifying the vast majority of the down days.

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

d.

```
        Direction.2009
glm.pred Down Up
    Down    9  5
    Up     34 56

    Correct: 0.625
```

e.

```
          Direction.2009
lda.class Down Up
     Down    9  5
     Up     34 56

    Correct: 0.625
```

f.

```
          Direction.2009
qda.class Down Up
     Down    0  0
     Up     43 61

    Correct: 0.58654
```

g.

```
         Direction.2009
knn.pred Down Up
    Down   21 30
    Up     22 31

    Correct: 0.50
```

h. Both logistic regression and LDA appear to produce the best results on this data. As both methods produce linear decision boundaries, it is possible that their forms were similar enough to result in the same predictions.

i. The following variable selections were attempted and implemented in glm, lda, qda and KNN:

- Lag1 and $Lag1^2$ terms were added to glm, lda and qda; no decrease in test error rate was observed.
- Adding $Lag1^2$ alone improves lda and glm test accuracy to their highest values (0.64423), but it would be bad practice to add a higher order term without the linear term.
- All Lag parameters were checked in linear and squared forms, these did not improve test accuracy scores beyond the baseline 0.625.
- The best test score using an acceptable model came from qda with the following form:
  $Lag2 + Lag2^2 + Lag2 * Lag3$

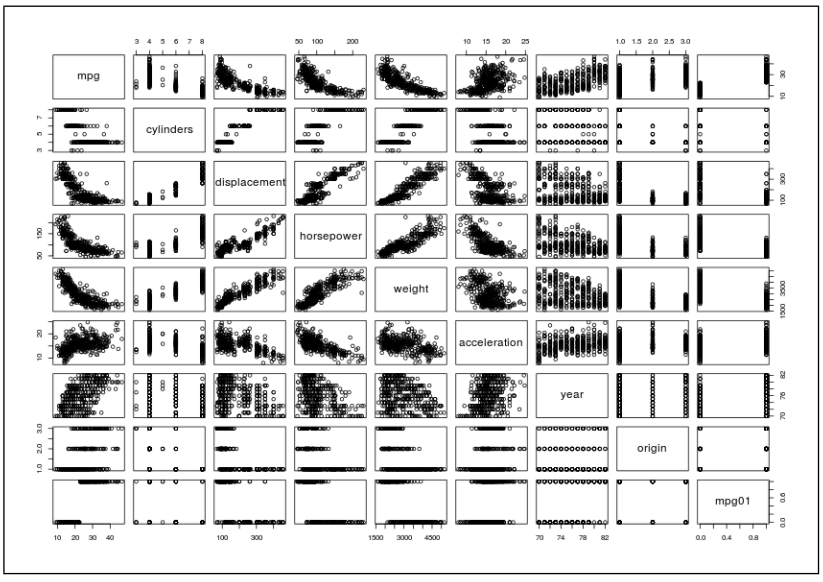
```
qda.class Down Up
     Down    8  3
     Up     35 58

    Correct: 0.63462
```

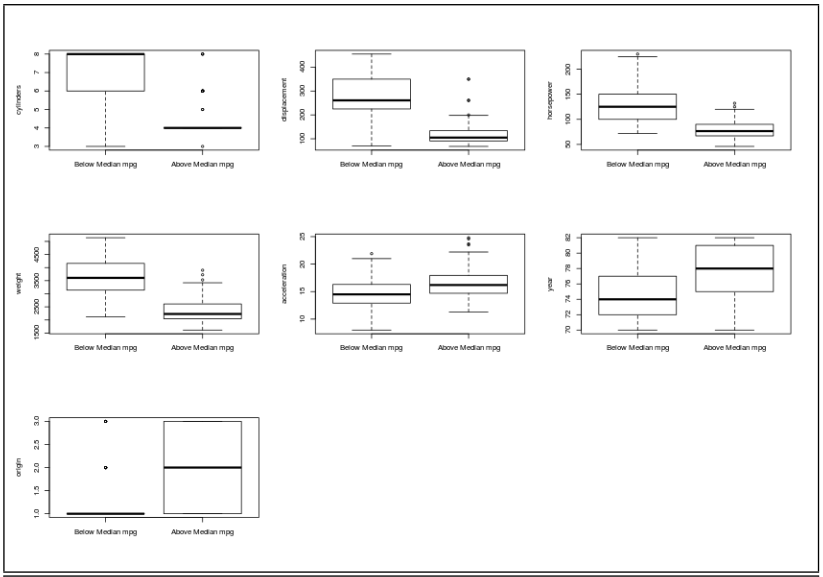
KNN Revisit: Looking at KNN with k = 1-8 and including/excluding Lag1, the test accuracy of KNN appears to increase up to k = 4. Inclusion of Lag1 increases the test error rate; without Lag1 the highest test accuracy is 0.61538. For k > 4 the accuracy begins to decrease.

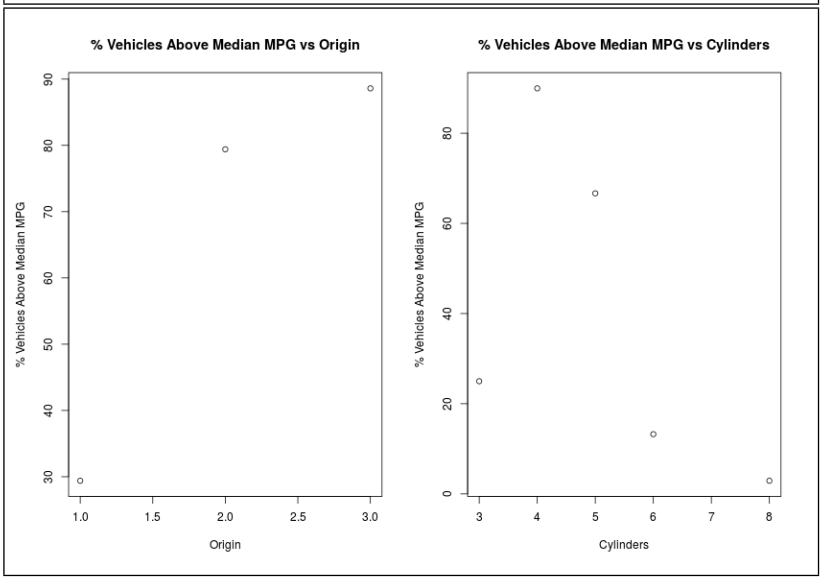
4.7.11

a. See supporting code.

b. Most, if not all of the features appear to have some impact on mpg01. Features that may be useful in predicting mpg01 include Origin, Displacement, Horsepower, and Weight and Year. Acceleration and Cylinders may also have an effect.







c. Since we've identified Year as a variable we'd like to use for prediction and the rows are ordered in terms of year, it doesn't make sense to simply split the data into two groups without randomization. See supporting code section for how the split was performed.

d. Using origin, displacement, horsepower, weight and year on an 80:20 train:test data split, an error rate of 0.051 was obtained.

```
          mpg01.test
lda.class  0  1
         0 36  1
         1  3 38
```

e. Using qda with the same setup as d results in an error rate of 0.103.

```
          mpg01.test
qda.class  0  1
         0 37  6
         1  2 33
```

f. Using logistic regression with the same setup as d results in an error rate of 0.051.

```
         mpg01.test
glm.pred  0  1
        0 37  2
        1  2 37
```

g. Using KNN with values of k from 1-16, the lowest training error rate achieved was at k values of 9 and 12. The error rate was 0.103. The confusion matrix for both k = 9 and 12 was the same:

```
          mpg01.test
knn.pred  0  1
       0 37  6
       1  2 33
```

4.7.13 The following investigation was performed:

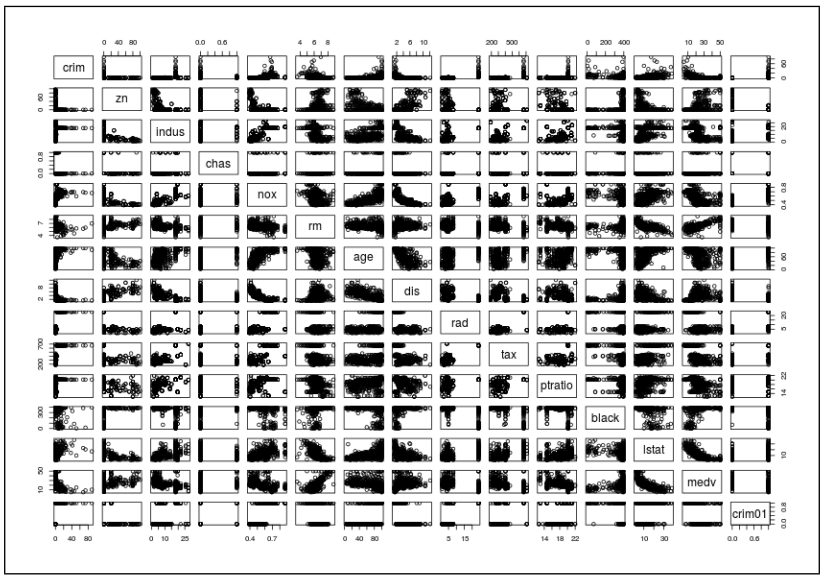
- The Boston data set was classified into two sets, one for the crime rate above the median, and the other below.
- Predictors for above/below median crime rate were examined and four sets were identified for investigation:
  - All predictors.
  - Predictors that appear to have strong and weak associations to above/below median crime rate (indus, nox, age, dis, rad, tax, ptratio, and lstat).
  - Predictors that appear to have strong associations to above/below median crime rate (indus, nox, age, dis, rad, and tax).
  - Of the predictors that appear to have strong associations to above/below median crime rate, a greedy approach was taken to removing predictors in an attempt to produce lower test error.
- The data was split into test/train groups using two methods:
  - Split into the first 319 entries for training and then the remaining entries from 320:406 for test. This split was observed to produce generally inferior results (error rates of approximately 0.05 higher than randomized split for all classifiers) and is not discussed further.
  - Split in a randomized 80:20 train:test split in the same manner used in part d of 4.7.11. This was done as there appeared to be streaks of above/below median crime rate in the data.

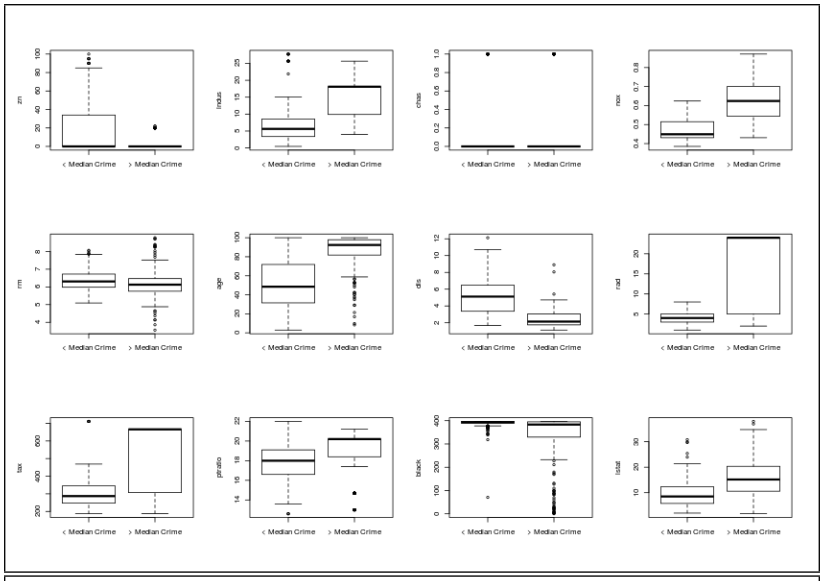
Results:

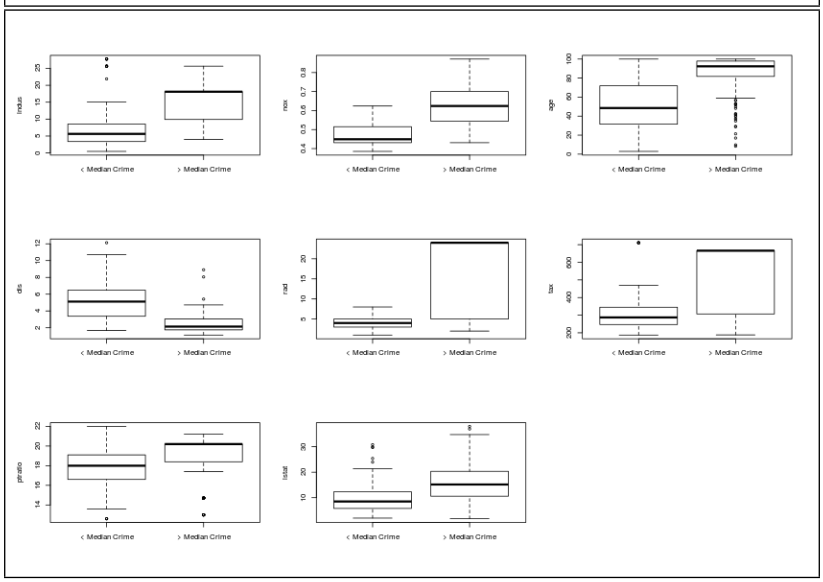
Generally speaking, KNN (with low k) produced the lowest training error followed by Logistic Regression and then LDA. This makes sense as the Boston data set has obvious structure. The Boston data set has within it clusters of entries that are extremely similar. By selecting our training data at random, we had access to potentially all of these regions, resulting in extremely low error rates for low k KNN. For example, using all predictors with k = 1, KNN had a 0.02 error rate. The error rate was then observed to increase as k increased. This suggests that KNN (or at least KNN with small k) should probably not be used for this data set as there is the potential to be overconfident in the model accuracy.

LDA achieved a min error rate of 0.13 in the case where only strong and weak predictors were used whereas logistic regression achieved a min error rate of 0.10 when all predictors were used. Attempting to further downselect predictors after the subset containing strong/weak predictors did not produce lower test errors in any of the cases.

Overall each model could be used to produce a low test error rate. However, the structure in the data is suspicious. Were this not an exercise, more time would need to be dedicated to understanding the data before applying any of these methods.








```
#########################################################################################################
######################################## Supporting Code ################################################
#########################################################################################################
# STAT4702 Homework 9
# Patrick Rogan
# UNI: psr2125

# 4.7.6 ----------------------------------------------------------------------
beta0 = -6
beta1 = 0.05
beta2 = 1
x1=40
x2=3.5
exp(beta0+beta1*x1+beta2*x2)/(1+exp(beta0+beta1*x1+beta2*x2))

uniroot(function(x) exp(-6+0.05*x+1*3.5)/(1+exp(-6+0.05*x+1*3.5))-0.5, c(0,100))

# 4.7.7 ----------------------------------------------------------------------
x = 4
mu1 = 10 #average %profit for dividend issuing companies
mu2 = 0.0 #average %profit for non-dividend issuing companies
var_hat = 36
pi1 = 0.8 # %companies issuing dividends
pi2 = 0.2

pi1*exp(-1/2*1/var_hat*(x-mu1)^2)/
```

```
  (pi1*exp(-1/2*1/var_hat*(x-mu1)^2) + pi2*exp(-1/2*1/var_hat*(x-mu2)^2))

# 4.7.10 ----------------------------------------------------------------------
# Part a
setwd("~/Desktop/STAT4702/HW09")
library(ISLR)
attach(Weekly)

names(Weekly)
summary(Weekly)

png("Weekly1.png",width=832,height=574, units = "px")
pairs(Weekly)
dev.off()
cor(Weekly[,-9])

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

glm.probs = predict(glm.fit, type="response")
glm.pred = rep("Down",length(glm.probs))
glm.pred[glm.probs > 0.5] = "Up" #same up/down labeling convention used
# in Smarket

# Part C
table(glm.pred,Direction)
mean(glm.pred == Direction)

# Part D
train = (Year<2009)
Weekly.2009 = Weekly[!train,]
Direction.2009 = Direction[!train]

glm.fit = glm(Direction ~ Lag2, family = binomial, data = Weekly, subset = train)
glm.probs = predict(glm.fit, Weekly.2009, type = "response")

glm.pred = rep("Down",length(Direction.2009))
glm.pred[glm.probs > 0.5] = "Up"

table(glm.pred, Direction.2009)
mean(glm.pred == Direction.2009)

# Part E
library(MASS)
lda.fit = lda(Direction ~ Lag2, data = Weekly, subset = train)
lda.pred = predict(lda.fit, Weekly.2009)
lda.class = lda.pred$class

table(lda.class, Direction.2009)
mean(lda.class == Direction.2009)

# Part F
qda.fit = qda(Direction ~ Lag2, data = Weekly, subset = train)
qda.class = predict(qda.fit, Weekly.2009)$class

table(qda.class, Direction.2009)
mean(qda.class == Direction.2009)

# Part G
library(class)
train.X = as.matrix(cbind(Lag2)[train,])
test.X = as.matrix(cbind(Lag2)[!train,])
```

```
train.Direction = Direction[train]

set.seed(1)
knn.pred = knn(train.X, test.X, train.Direction, k = 1)
table(knn.pred, Direction.2009)

(21+31)/(21+30+22+31)

# Part I
qda.fit = qda(Direction ~ Lag2+I(Lag2^2)+Lag2*Lag3, data = Weekly, subset = train)
qda.class = predict(qda.fit, Weekly.2009)$class

table(qda.class, Direction.2009)
mean(qda.class == Direction.2009)

# Revisit KNN
train.X = as.matrix(cbind(Lag2)[train,])
test.X = as.matrix(cbind(Lag2)[!train,])
train.Direction = Direction[train]

set.seed(1)
knn.pred = knn(train.X, test.X, train.Direction, k = 4)
table(knn.pred, Direction.2009)
(20+44)/(20+17+23+44)

# 4.7.11 ----------------------------------------------------------------
Auto = read.csv("/home/pat/Desktop/STAT4702/ISL_Datasets/Auto.csv",
                header = T, na.strings = "?")
Auto = na.omit(Auto)
attach(Auto)

# Part A
mpg01 = rep(0,length(mpg))
mpg01[(mpg > median(mpg))] = 1

Auto = data.frame(Auto,mpg01)

# Part B
png("Auto1.png",width=832,height=574, units = "px")
pairs(Auto[,-9]) #Exclude name
dev.off()

png("Auto2.png",width=832,height=574, units = "px")
par(mfrow = c(3,3))
for (i in 2:8){
  boxplot(Auto[mpg01==0,i],Auto[mpg01==1,i],ylab = colnames(Auto)[i],
          names=c("Below Median mpg","Above Median mpg"))
}
dev.off()

png("Auto3.png",width=832,height=574, units = "px")
par(mfrow = c(1,2))
perOri = rep(0,3)
for (j in 1:3){
  perOri[j] = sum(mpg01[origin == j])/sum(origin == j)*100
}
plot(c(1,2,3),perOri,xlab="Origin",ylab="% Vehicles Above Median MPG",
     main="% Vehicles Above Median MPG vs Origin")

perCyl = rep(0,5)
for (j in sort(unique(cylinders))){
  perCyl[j] = sum(mpg01[cylinders == j])/sum(cylinders == j)*100
}
plot(sort(unique(cylinders)),perCyl[c(-1,-2,-7)],xlab="Cylinders",ylab="% Vehicles Above Median MPG",
```

```
     main="% Vehicles Above Median MPG vs Cylinders")
dev.off()

# Part C
library(caret) #use createDataPartition from caret librart to create randomized 80:20
# split so that year can be used as a predictor
set.seed(1)
train.ix = as.integer(createDataPartition(Auto$mpg01, p = .8, list = FALSE, times = 1))
train = rep(FALSE,length(mpg01))
train[train.ix] = TRUE
Auto.test = Auto[!train,]
mpg01.test = mpg01[!train]

# Part D
lda.fit = lda(mpg01 ~ origin + displacement + horsepower + weight + year,
              data = Auto, subset = train)

lda.pred = predict(lda.fit, Auto.test)
lda.class = lda.pred$class

table(lda.class, mpg01.test)
mean(lda.class == mpg01.test)

# Part E
qda.fit = qda(mpg01 ~ origin + displacement + horsepower + weight + year,
              data = Auto, subset = train)
qda.class = predict(qda.fit, Auto.test)$class

table(qda.class, mpg01.test)
mean(qda.class == mpg01.test)

# Part F
glm.fit = glm(mpg01 ~ origin + displacement + horsepower + weight + year,
              family = binomial, data = Auto, subset = train)
glm.probs = predict(glm.fit, Auto.test, type = "response")

glm.pred = rep(0,length(mpg01.test))
glm.pred[glm.probs > 0.5] = 1

table(glm.pred, mpg01.test)
mean(glm.pred == mpg01.test)

# Part G
train.X = as.matrix(cbind(origin, displacement, horsepower, weight, year)[train,])
test.X = as.matrix(cbind(origin, displacement, horsepower, weight, year)[!train,])
train.mpg01 = mpg01[train]

set.seed(1)
knn.pred = knn(train.X, test.X, train.mpg01, k = 16)
table(knn.pred, mpg01.test)

# 4.7.13 ------------------------------------------------------------------
library(MASS)
attach(Boston)

crim01 = rep(0,length(crim))
crim01[crim > median(crim)] = 1
Boston = data.frame(Boston,crim01)

png("Boston1.png",width=832,height=574, units = "px")
pairs(Boston) #Exclude name
dev.off()

png("Boston2.png",width=832,height=574, units = "px")
```

```
par(mfrow = c(3,4))
for (i in 2:13){
  boxplot(Boston[crim01==0,i],Boston[crim01==1,i],ylab = colnames(Boston)[i],
          names=c("< Median Crime","> Median Crime"))
}
dev.off()

# Possible variables include indus, nox, age, dis, rad, tax, pratio, and lstat
# replot to weed out other variables
png("Boston3.png",width=832,height=574, units = "px")
par(mfrow = c(3,3))
for (i in c(3,5,7,8,9,10,11,13)){
  boxplot(Boston[crim01==0,i],Boston[crim01==1,i],ylab = colnames(Boston)[i],
          names=c("< Median Crime","> Median Crime"))
}
dev.off()

# To further simplify, we'll rule out lstat and pratio leaving:
# indus, nox, age, dis, rad, tax

# So, we'll look at 4 subsets,
# i. All predictors
# ii. First subset of predictors
# iii. Second subset of predictors
# iv. Several subsets of iii in search of the best result

set.seed(1) #Looking at crim01 there are obvious streaks of 0s and 1s, we sample randomly
# to reduce any impact on our final results
train.ix = as.integer(createDataPartition(Boston$crim01, p = .8, list = FALSE, times = 1))
train = rep(FALSE,length(crim01))
train[train.ix] = TRUE
#train[320:406] = TRUE # second method, just look at last 86 points for test
Boston.test = Boston[!train,]
crim01.test = crim01[!train]

# i. All predictors
glm.fit = glm(crim01 ~ zn + indus + chas + nox + rm + age + dis+
                rad + tax + ptratio + black +lstat + medv,
              family = binomial, data = Boston, subset = train)
glm.probs = predict(glm.fit, Boston.test, type = "response")

glm.pred = rep(0,length(crim01.test))
glm.pred[glm.probs > 0.5] = 1

table(glm.pred, crim01.test)
mean(glm.pred == crim01.test)

lda.fit = lda(crim01 ~ zn + indus + chas + nox + rm + age + dis+
                rad + tax + ptratio + black +lstat + medv,
              data = Boston, subset = train)

lda.pred = predict(lda.fit, Boston.test)
lda.class = lda.pred$class

table(lda.class, crim01.test)
mean(lda.class == crim01.test)

train.X = as.matrix(cbind(zn, indus, chas, nox, rm, age, dis,
                          rad, tax, ptratio, black, lstat, medv)[train,])
test.X = as.matrix(cbind(zn, indus, chas, nox, rm, age, dis,
                         rad, tax, ptratio, black, lstat, medv)[!train,])
train.crim01 = crim01[train]

set.seed(1)
```

```
knn.pred = knn(train.X, test.X, train.crim01, k = 1)
table(knn.pred, crim01.test)

# ii. First subset of predictors: indus, nox, age, dis, rad, tax, ptratio, and lstat
glm.fit = glm(crim01 ~ indus + nox + age + dis+
                rad + tax + ptratio +lstat,
              family = binomial, data = Boston, subset = train)
glm.probs = predict(glm.fit, Boston.test, type = "response")

glm.pred = rep(0,length(crim01.test))
glm.pred[glm.probs > 0.5] = 1

table(glm.pred, crim01.test)
mean(glm.pred == crim01.test)

lda.fit = lda(crim01 ~ indus + nox + age + dis+
                rad + tax + ptratio +lstat,
              data = Boston, subset = train)

lda.pred = predict(lda.fit, Boston.test)
lda.class = lda.pred$class

table(lda.class, crim01.test)
mean(lda.class == crim01.test)

train.X = as.matrix(cbind(indus, nox, age, dis,
                          rad, tax, ptratio, lstat)[train,])
test.X = as.matrix(cbind(indus, nox, age, dis,
                         rad, tax, ptratio, lstat)[!train,])
train.crim01 = crim01[train]

set.seed(1)
knn.pred = knn(train.X, test.X, train.crim01, k = 1)
table(knn.pred, crim01.test)

# iii. Second subset of predictors: indus, nox, age, dis, rad, tax
glm.fit = glm(crim01 ~ indus + nox + age + dis+
                rad + tax,
              family = binomial, data = Boston, subset = train)
glm.probs = predict(glm.fit, Boston.test, type = "response")

glm.pred = rep(0,length(crim01.test))
glm.pred[glm.probs > 0.5] = 1

table(glm.pred, crim01.test)
mean(glm.pred == crim01.test)

lda.fit = lda(crim01 ~ indus + nox + age + dis+
                rad + tax,
              data = Boston, subset = train)

lda.pred = predict(lda.fit, Boston.test)
lda.class = lda.pred$class

table(lda.class, crim01.test)
mean(lda.class == crim01.test)

train.X = as.matrix(cbind(indus, nox, age, dis,
                          rad, tax)[train,])
test.X = as.matrix(cbind(indus, nox, age, dis,
                         rad, tax)[!train,])
train.crim01 = crim01[train]

set.seed(1)
```

```
knn.pred = knn(train.X, test.X, train.crim01, k = 1)
table(knn.pred, crim01.test)

# iv. Several subsets of iii in search of the best result, use greedy approach
# remove predictors that improve the model starting from tax working up to indus
# note this was done with the randomized train indices. Turns out the removal of
# any one of these parameters does not improve all of the models.
glm.fit = glm(crim01 ~ indus + nox + age + dis+
                rad + tax,
              family = binomial, data = Boston, subset = train)
glm.probs = predict(glm.fit, Boston.test, type = "response")

glm.pred = rep(0,length(crim01.test))
glm.pred[glm.probs > 0.5] = 1

table(glm.pred, crim01.test)
mean(glm.pred == crim01.test)

lda.fit = lda(crim01 ~ indus + nox + age + dis+
                rad + tax,
              data = Boston, subset = train)

lda.pred = predict(lda.fit, Boston.test)
lda.class = lda.pred$class

table(lda.class, crim01.test)
mean(lda.class == crim01.test)

train.X = as.matrix(cbind(indus, nox, age, dis,
                          rad, tax)[train,])
test.X = as.matrix(cbind(indus, nox, age, dis,
                         rad, tax)[!train,])
train.crim01 = crim01[train]

set.seed(1)
knn.pred = knn(train.X, test.X, train.crim01, k = 1)
table(knn.pred, crim01.test)
```