ISLR Chapter 4

Exercise 1

Using a little bit of algebra, use (4.2) to achieve (4.3). In other words, the logistic function representation and logit representation for the logistic regression model are equivalent. Where:

$$(4.2)\ p(X) = \frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}}$$

$$(4.3)\ \frac{p(X)}{1-p(X)} = e^{\beta_0+\beta_1 X}$$

So, $$\frac{p(X)}{1-p(X)} = \frac{\frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}}}{1-\frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}}} = \frac{\frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}}}{\frac{1+e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}}-\frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}}} = \frac{\frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}}}{\frac{1}{1+e^{\beta_0+\beta_1 X}}} = e^{\beta_0+\beta_1 X}$$

Exercise 6

Suppose we collect data for a group of students in a statistics class with variables $X1 =$ hours studied, $X2 =$ undergrad GPA, and $Y =$ receive an A. We fit a logistic regression and produce estimated coefficients:

$$\hat{\beta}_0 = -6 \quad \hat{\beta}_1 = 0.05 \quad \hat{\beta}_2 = 1$$

6. a) Estimate the probability that a student who studies for 40 h and has an undergrad GPA of 3.5 gets an A in the class.

Remember from the previous exercise that:

$$p(X) = \frac{\exp(\beta_0 + \beta_1 X_1 + \beta_2 X_2)}{1 + \exp(\beta_0 + \beta_1 X_1 + \beta_2 X_2)}$$

Estimating the probability for $X = [40 hours, 3.5 GPA]$ yields:

$$p(X) = \frac{\exp(-6 + 0.05X_1 + X_2)}{1 + \exp(-6 + 0.05X_1 + X_2)} = \frac{\exp(-6 + 0.05 * 40 + 3.5)}{1 + \exp(-6 + 0.05 * 40 + 3.5)} = \frac{\exp(-0.5)}{1 + \exp(-0.5)} = 37.75\%$$

6. b) How many hours would the student in part (a) need to study to have a 50% chance of getting an A in the class?

Estimating X1 where $X = [X_1 hours, 3.5 GPA]$ such that $p(X) = 0.5$ yields:

$$0.50 = \frac{\exp(-6 + 0.05X_1 + 3.5)}{1 + \exp(-6 + 0.05X_1 + 3.5)} \Leftrightarrow 0.50(1 + \exp(-2.5 + 0.05X_1)) = \exp(-2.5 + 0.05X_1)$$

$$\Leftrightarrow 0.50 + 0.50\exp(-2.5 + 0.05X_1)) = \exp(-2.5 + 0.05X_1) \Leftrightarrow 0.50 = 0.50\exp(-2.5 + 0.05X_1)$$

$$\Leftrightarrow log(1) = -2.5 + 0.05X_1 \Leftrightarrow X_1 = 2.5/0.05 = 50 hours$$

Exercise 10

This question should be answered using the Weekly data set, which is part of the ISLR package. This data is similar in nature to the Smarket data from this chapter's lab, except that it contains 1,089 weekly returns for 21 years, from the beginning of 1990 to the end of 2010.

10.a) Produce some numerical and graphical summaries of the Weekly data. Do there appear to be any patterns?

In [1]:

```
#No warnings for print version
options(warn=-1)

#First, load the necessary R-packages:
#install.packages("ISLR")
#install.packages("MASS")
#install.packages("class")

#call the packages you need
library("ISLR")

#Take a first look at the data - this time no need to clean the data
fix(Weekly)    # allows to edit the data
attach(Weekly) # eliminates the need of refering to a variable like Weekly$variable.

# Use summary function to produce a numerical summary for each variable
summary(Weekly)
```

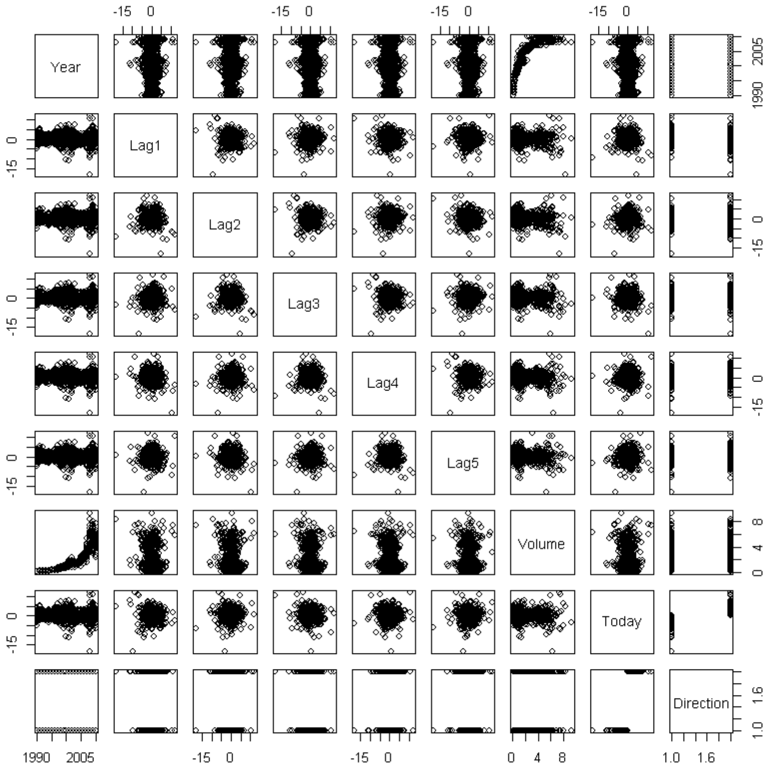
```
      Year          Lag1               Lag2               Lag3
 Min.   :1990   Min.   :-18.1950   Min.   :-18.1950   Min.   :-18.1950
 1st Qu.:1995   1st Qu.: -1.1540   1st Qu.: -1.1540   1st Qu.: -1.1580
 Median :2000   Median :  0.2410   Median :  0.2410   Median :  0.2410
 Mean   :2000   Mean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

In [2]:
```
# Use cor function to produce a table of correlations for all variables (excluding Direction non-numerical variable)
cor(Weekly[,-9])
```

A matrix: 8 × 8 of type dbl

	Year	Lag1	Lag2	Lag3	Lag4	Lag5	Volume	Today
Year	1.00000000	-0.032289274	-0.03339001	-0.03000649	-0.031127923	-0.030519101	0.84194162	-0.032459894
Lag1	-0.03228927	1.000000000	-0.07485305	0.05863568	-0.071273876	-0.008183096	-0.06495131	-0.075031842
Lag2	-0.03339001	-0.074853051	1.00000000	-0.07572091	0.058381535	-0.072499482	-0.08551314	0.059166717
Lag3	-0.03000649	0.058635682	-0.07572091	1.00000000	-0.075395865	0.060657175	-0.06928771	-0.071243639
Lag4	-0.03112792	-0.071273876	0.05838153	-0.07539587	1.000000000	-0.075675027	-0.06107462	-0.007825873
Lag5	-0.03051910	-0.008183096	-0.07249948	0.06065717	-0.075675027	1.000000000	-0.05851741	0.011012698
Volume	0.84194162	-0.064951313	-0.08551314	-0.06928771	-0.061074617	-0.058517414	1.00000000	-0.033077783
Today	-0.03245989	-0.075031842	0.05916672	-0.07124364	-0.007825873	0.011012698	-0.03307778	1.000000000

In [3]:
```
# Use pairs function to produce a graphical summary
pairs(Weekly)
```



Answer: Yes, it appears that Year and Volume have a non-linear and positive relation.

10. b) Use the full data set to perform a logistic regression with Direction as the response and the five lag variables plus Volume as predictors. Use the summary function to print the results. Do any of the predictors appear to be statistically significant? If so, which ones?

In [4]:
```
# Estimate a generalized linear regression model where the third input family is a description of the error distributi
on
# and link function to be used in the model, supplied as the result of a call to a family function - here use binomia
l.
# Why binomial? Because our independent variable Direction takes two values.

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

# Use summary function to print the results
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
```

Answer: the predictor Lag 2 appears to have some statistical significance with a p-value smaller than 3%.

10. c) Compute the confusion matrix and overall fraction of correct predictions. Explain what the confusion matrix is telling you about the types of mistakes made by logistic regression.

In [5]:
```
# use predict function on results of previous regression in 10.b)
glm.probs = predict(glm.fit, type="response")

#create dataframe that is a vector always taking string "Down" same size as #obs
glm.pred = rep("Down", length(glm.probs))

#substitute for "Up" whenever the estimated probability is above 0.5
glm.pred[glm.probs>.5] = "Up"

#construct a summary table with the prediction against the actual values of the variable Direction
table(glm.pred, Direction)
```

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

Answer:

Percentage of correct predictions: (54+557)/(54+557+48+430) = 56.1%.

During weeks when the market goes up, the logistic regression is right about 557/(557+48) = 92.1% of the time.

During weeks when the market goes down, the logistic regression is right about 54/(430+54) = 11.2% of the time.

10. d) Now fit the logistic regression model using a training data period from 1990 to 2008, with Lag2 as the only predictor. Compute the confusion matrix and the overall fraction of correct predictions for the held out data (that is, the data from 2009 and 2010).

In [6]:
```
# generate condition for our training data
train = (Year < 2009)

# create dataframe for the Weekly data from 2009 and 2010 (usage of ! to define the "opposite")
Weekly.0910 = Weekly[!train,]

# run regression on the training data subset
glm.fit = glm(Direction~Lag2,
              data=Weekly,
              family=binomial,
              subset=train)

# create dataframe
glm.probs = predict(glm.fit, Weekly.0910,type="response")

# fill with our predictions
glm.pred = rep("Down", length(glm.probs))
glm.pred[glm.probs>.5] = "Up"

# construct confusion table using only the subset data
Direction.0910 = Direction[!train]
table(glm.pred, Direction.0910)

# overall fraction of correct predictions
mean(glm.pred == Direction.0910)
```

```
        Direction.0910
glm.pred Down Up
    Down    9  5
    Up     34 56
```

0.625

10. e) Repeat (d) using LDA.

In [7]:
```r
#call the packages you need
library("MASS")

# same approach as before but now using LDA method
lda.fit = lda(Direction ~ Lag2, data=Weekly, subset=train)
lda.pred = predict(lda.fit, Weekly.0910)
table(lda.pred$class, Direction.0910)
mean(lda.pred$class == Direction.0910)
```

```
      Direction.0910
       Down Up
  Down    9  5
  Up     34 56
```

0.625

10. f) Repeat (d) using QDA.

In [8]:
```r
# same approach as before but now using QDA method
qda.fit = qda(Direction~Lag2, data=Weekly, subset=train)
qda.class = predict(qda.fit, Weekly.0910)$class
table(qda.class, Direction.0910)
mean(qda.class == Direction.0910)
```

```
         Direction.0910
qda.class Down Up
     Down    0  0
     Up     43 61
```

0.586538461538462

10. g) Repeat (d) using KNN with K = 1.

In [9]:
```
#call the packages you need
library("class")

# same approach as before but now using KNN method with K=1
train.X = as.matrix(Lag2[train])
test.X = as.matrix(Lag2[!train])
train.Direction = Direction[train]

#set seed to get the same results -  if several observations are tied as nearest neighbors, then R will randomly break the tie.
#In order to break the tie the same way each time, you set the seed so you can reproduce the results exactly
set.seed(1)

#KNN prediction uses a different function
# documentationn here: https://www.rdocumentation.org/packages/class/versions/7.3-17/topics/knn
knn.pred = knn(train.X, test.X, train.Direction, k=1)
table(knn.pred, Direction.0910)
mean(knn.pred == Direction.0910)
```

```
        Direction.0910
knn.pred Down Up
    Down   21 30
    Up     22 31
```

0.5

10. h) Which of these methods appears to provide the best results on this data?

Logistic regression and LDA have the smallest test error rates.

10. i) Experiment with different combinations of predictors, including possible transformations and interactions, for each of the methods. Report the variables, method, and associated confusion matrix that appears to provide the best results on the held out data. Note that you should also experiment with values for K in the KNN classifier.

In [10]:
```
# Logistic regression with Lag2:Lag1
glm.fit = glm(Direction~Lag2:Lag1, data=Weekly, family=binomial, subset=train)
glm.probs = predict(glm.fit, Weekly.0910, type="response")
glm.pred = rep("Down", length(glm.probs))
glm.pred[glm.probs>.5] = "Up"
Direction.0910 = Direction[!train]
table(glm.pred, Direction.0910)
mean(glm.pred == Direction.0910)
```

```
        Direction.0910
glm.pred Down Up
    Down    1  1
    Up     42 60
```

0.586538461538462

In [11]:
```
# LDA with Lag2 interaction with Lag1
lda.fit = lda(Direction ~ Lag2:Lag1, data=Weekly, subset=train)
lda.pred = predict(lda.fit, Weekly.0910)
mean(lda.pred$class == Direction.0910)
```

0.576923076923077

In [12]:
```
# QDA with sqrt(abs(Lag2))
qda.fit = qda(Direction~Lag2+sqrt(abs(Lag2)), data=Weekly, subset=train)
qda.class = predict(qda.fit, Weekly.0910)$class
table(qda.class, Direction.0910)
mean(qda.class == Direction.0910)
```

```
         Direction.0910
qda.class Down Up
     Down   12 13
     Up     31 48
```

0.576923076923077

In [13]:
```
# KNN k =10, as before KNN uses a different command
set.seed(1)
knn.pred = knn(train.X, test.X, train.Direction, k=10)
table(knn.pred, Direction.0910)
mean(knn.pred == Direction.0910)
```

```
        Direction.0910
knn.pred Down Up
    Down   17 21
    Up     26 40
```

0.548076923076923

In [14]:
```
# KNN k = 100
set.seed(1)
knn.pred = knn(train.X, test.X, train.Direction, k=100)
table(knn.pred, Direction.0910)
mean(knn.pred == Direction.0910)
```

```
        Direction.0910
knn.pred Down Up
    Down   10 11
    Up     33 50
```

0.576923076923077

Answer: Out of these permutations, the original LDA and logistic regression have better performance in terms of test error rate.

Exercise 12

This problem involves writing functions.

12. a) Write a function, Power(), that prints out the result of raising 2 to the 3rd power. In other words, your function should compute 2^3 and print out the results.

Hint: Recall that x^a raises x to the power a. Use the print() function to output the result.

In [15]:
```
Power = function() {
  2^3
}
print(Power())
```

```
[1] 8
```

12 .b) Create a new function, Power2(), that allows you to pass any two numbers, x and a, and prints out the value of x^a. You can do this by beginning your function with the line (Power2 =function (x,a){) You should be able to call your function by entering, for instance, (Power2 (3,8)) on the command line. This should output the value of 38, namely, 6,561.

In [16]:
```
Power2 = function(x, a) {
  x^a
}
Power2(3,8)
```

6561

12. c) Using the Power2() function that you just wrote, compute 10^3, 8^{17}, and 131^3.

In [17]:
```
Power2(10, 3)
Power2(8, 17)
Power2(131, 3)
```

1000

2251799813685248

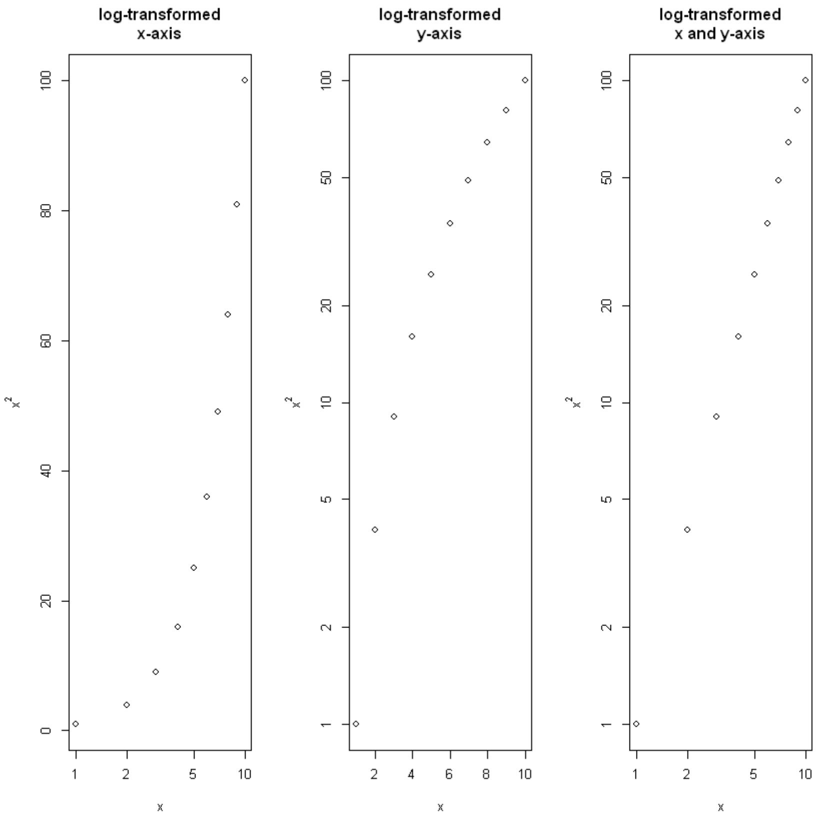
2248091

12. d) Now create a new function, Power3(), that actually returns the result x^a as an R object, rather than simply printing it to the screen. That is, if you store the value x^a in an object called result within your function, then you can simply return() this result, using the following line: (return(result)). This should be the last line in your function, before the } symbol.

In [18]:
```
Power3 = function(x, a) {
  result = x^a
  return(result)
}
```

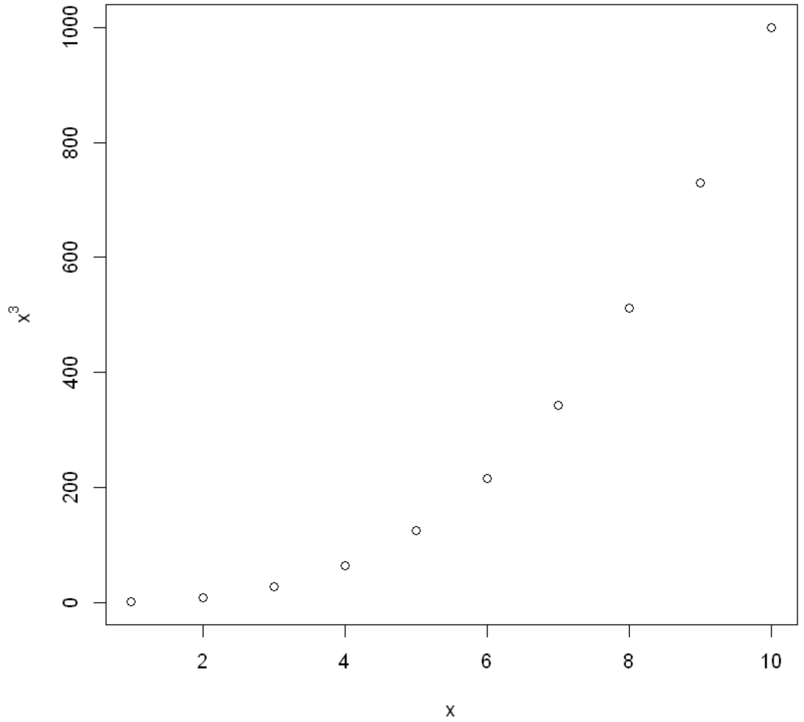
12. e) Now using the Power3() function, create a plot of f(x) = x^2. The x-axis should display a range of integers from 1 to 10, and the y-axis should display x^2. Label the axes appropriately, and use an appropriate title for the figure. Consider displaying either the x-axis, the y-axis, or both on the log-scale. You can do this by using log="x", log="y", or log="xy" as arguments to the plot() function.

In [19]:
```
x = 1:10
par(mfrow=c(1,3))# graph parameters
plot(x, Power3(x, 2),  log="x", ylab=expression(x^2), xlab=expression(x),
     main="log-transformed\nx-axis")
plot(x, Power3(x, 2),  log="y", ylab=expression(x^2), xlab=expression(x),
     main="log-transformed\ny-axis")
plot(x, Power3(x, 2),  log="xy", ylab=expression(x^2), xlab=expression(x),
     main="log-transformed\nx and y-axis")
par(mfrow=c(1,1))# reset graphic parameters
```



12. f) Create a function, PlotPower(), that allows you to create a plot of x against x^a for a fixed a and for a range of values of x. For instance, if you call (PlotPower (1:10 ,3)) then a plot should be created with an x-axis taking on values 1, 2, . . . , 10, and a y-axis taking on values $1^3, 2^3, \ldots, 10^3$

```
In [20]: PlotPower = function(x, a) {
           ylab_text <- bquote('x'^.(a))   # write y-axis title
           plot(x, Power3(x, a),
                ylab = ylab_text)
         }
         PlotPower(1:10, 3)
```



Exercise 13

Using the Boston data set, fit classification models in order to predict whether a given suburb has a crime rate above or below the median. Explore logistic regression, LDA, and KNN models using various subsets of the predictors. Describe your findings.

In [21]:
```
summary(Boston)
attach(Boston)
crime01 = rep(0, length(crim))
crime01[crim>median(crim)] = 1
Boston = data.frame(Boston, crime01)

train = 1:(dim(Boston)[1]/2)
test = (dim(Boston)[1]/2+1):dim(Boston)[1]
Boston.train = Boston[train,]
Boston.test = Boston[test,]
crime01.test = crime01[test]
```

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

In [22]:
```
# logistic regression
glm.fit = glm(crime01~.-crime01-crim,
              data=Boston, family=binomial, subset=train)
glm.probs = predict(glm.fit, Boston.test, type="response")
glm.pred = rep(0, length(glm.probs))
glm.pred[glm.probs > 0.5] = 1
mean(glm.pred != crime01.test)
```

0.181818181818182

Answer: 18.2% test error rate.

In [23]:
```
glm.fit = glm(crime01~.-crime01-crim-chas-tax,
              data=Boston, family=binomial, subset=train)
glm.probs = predict(glm.fit, Boston.test, type="response")
glm.pred = rep(0, length(glm.probs))
glm.pred[glm.probs > 0.5] = 1
mean(glm.pred != crime01.test)
```

0.185770750988142

Answer: 18.6% test error rate.

In [24]:
```
# LDA
lda.fit = lda(crime01~.-crime01-crim, data=Boston, subset=train)
lda.pred = predict(lda.fit, Boston.test)
mean(lda.pred$class != crime01.test)
```

0.134387351778656

Answer: 13.4% test error rate.

In [25]:
```
lda.fit = lda(crime01~.-crime01-crim-chas-tax, data=Boston, subset=train)
lda.pred = predict(lda.fit, Boston.test)
mean(lda.pred$class != crime01.test)
```

0.122529644268775

Answer: 12.3% test error rate.

In [26]:
```
lda.fit = lda(crime01~.-crime01-crim-chas-tax-lstat-indus-age,
              data=Boston, subset=train)
lda.pred = predict(lda.fit, Boston.test)
mean(lda.pred$class != crime01.test)
```

0.118577075098814

Answer: 11.9% test error rate.

In [27]:
```
# KNN
library(class)
train.X = cbind(zn, indus, chas, nox, rm, age, dis, rad, tax, ptratio, black,
                lstat, medv)[train,]
test.X = cbind(zn, indus, chas, nox, rm, age, dis, rad, tax, ptratio, black,
                lstat, medv)[test,]
train.crime01 = crime01[train]
set.seed(1)
# KNN(k=1)
knn.pred = knn(train.X, test.X, train.crime01, k=1)
mean(knn.pred != crime01.test)
```

0.458498023715415

Answer: 45.8% test error rate.

In [28]:
```
# KNN(k=10)
set.seed(1)
knn.pred = knn(train.X, test.X, train.crime01, k=10)
mean(knn.pred != crime01.test)
```

0.110671936758893

Answer: 11.1% test error rate.

In [29]:
```
# KNN(k=100)
set.seed(1)
knn.pred = knn(train.X, test.X, train.crime01, k=100)
mean(knn.pred != crime01.test)
```

0.486166007905138

Answer: 48.6% test error rate.

In general, the best models are the ones with the smaller test error rates. In our case, this means that the lda.fit and the KNN with K=10 are the best modles.