Chapter 4 Exercises

From *An Introduction to Statistical Learning with Applications in R*

Jacob Zeiher

May 24, 2018

Contents

Conceptual

Problem 1

From equation 4.2, we have that the logistic function is given by

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

Then we have:

$$\begin{aligned}
p(X)(1+e^{\beta_0+\beta_1 X}) &= e^{\beta_0+\beta_1 X} \\
p(X)+p(X)e^{\beta_0+\beta_1 X} &= e^{\beta_0+\beta_1 X} \\
p(X) &= e^{\beta_0+\beta_1 X} - p(X)e^{\beta_0+\beta_1 X} \\
p(X) &= e^{\beta_0+\beta_1 X}(1-p(X)) \\
\frac{p(X)}{1-p(X)} &= e^{\beta_0+\beta_1 X}.
\end{aligned}$$

The last line above is exactly equation 4.3, so we can conclude that equations 4.2 and 4.3 are equivalent.

Problem 2

We classify each observation to the class k for which

$$p_k(X) = \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x-\mu_k)^2\right)}{\sum_{l=1}^K \pi_l \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x-\mu_l)^2\right)}$$

is maximized. Since the logarithm function is monotonically increasing, maximizing $p_k(x)$ is equivalent to maximizing $\log(p_k(X))$. That is, maximizing the above equation with respect to k is equivalent to maximizing

$$\log(p_k(X)) = \log\left(\pi_k \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x-\mu_k)^2\right)\right) - \log\left(\sum_{l=1}^K \pi_l \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x-\mu_l)^2\right)\right)$$

with respect to k. Note that the log on the right is independent of k, so maximizing $\log(p_k(X))$ is equivalent to maximizing

$$\begin{aligned}
\omega_k(X) &= \log\left(\pi_k \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x-\mu_k)^2\right)\right) \\
&= \log(\pi_k) + \log(1) - \log\left(\sqrt{2\pi}\sigma\right) - \frac{1}{2\sigma^2}(x-\mu_k)^2 \\
&= \log(\pi_k) - \log\left(\sqrt{2\pi}\sigma\right) - \frac{1}{2\sigma^2}(x^2 - 2x\mu_k + \mu_k^2) \\
&= \log(\pi_k) - \log\left(\sqrt{2\pi}\sigma\right) - \frac{x^2}{2\sigma^2} + \frac{2x\mu_k}{2\sigma^2} - \frac{\mu_k^2}{2\sigma^2} \\
&= \log(\pi_k) - \log\left(\sqrt{2\pi}\sigma\right) - \frac{x^2}{2\sigma^2} + \frac{x\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2}.
\end{aligned}$$

Since the $\log\left(\sqrt{2\pi}\sigma\right)$ and $\frac{x^2}{2\sigma^2}$ in the previous equation are independent of k, maximizing $\log(p_k(X))$ then becomes equivalent to maximizing

$$\delta_k(x) = \log(\pi_k) + \frac{x\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2}.$$

Rearranging the terms of the previous equations, we see that maximizing $p_k(X)$ with respect to k is equivalent to maximizing

$$\delta_k(x) = \frac{x\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k)$$

with respect to k. This is exactly equation 4.13, the discriminant function.

Problem 3

Since we considering the case where $p = 1$, it will always be the case that $\Sigma = I_1$. Moreover, $|\Sigma| = |I_1| = 1$, so we can ignore the class-speicif covariance matrix. From here the result proceeds in a similar manner as in Problem 1. We begin by noting that the QDA with $p = 1$ will classify observations to the class k for which

$$p_k(X) = \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma_k} \exp\left(-\frac{1}{2\sigma_k^2}(x-\mu_k)^2\right)}{\sum_{l=1}^K \pi_l \frac{1}{\sqrt{2\pi}\sigma_l} \exp\left(-\frac{1}{2\sigma_l^2}(x-\mu_l)^2\right)}$$

is maximized. Since the logarithm function is monotonically increasing, maximizing $p_k(X)$ is equivalent to maximizing $\log(p_k(X))$. That is, we want to maximize

$$\log(p_k(X)) = \log\left(\pi_k \frac{1}{\sqrt{2\pi}\sigma_k} \exp\left(-\frac{1}{2\sigma_k^2}(x-\mu_k)^2\right)\right) - \log\left(\sum_{l=1}^K \pi_l \frac{1}{\sqrt{2\pi}\sigma_l} \exp\left(-\frac{1}{2\sigma_l^2}(x-\mu_l)^2\right)\right)$$

with respect to k. Since the right-most logarithm does not depend on k, maximizing $p_k(X)$ becomes equivalent to maximizing

$$\begin{aligned}
\omega_k(X) &= \log\left(\pi_k \frac{1}{\sqrt{2\pi}\sigma_k}\exp\left(-\frac{1}{2\sigma_k^2}(x-\mu_k)^2\right)\right) \\
&= \log(\pi_k) + \log\left(\frac{1}{\sqrt{2\pi}\sigma_k}\right) - \frac{1}{2\sigma_k^2}(x-\mu_k)^2 \\
&= \log(\pi_k) - \log\left(\sqrt{2\pi}\sigma_k\right) - \frac{1}{2\sigma_k^2}(x^2 - 2x\mu_k + \mu_k^2) \\
&= \log(\pi_k) - \log\left(\sqrt{2\pi}\right) - \log(\sigma_k) - \frac{x^2}{2\sigma_k^2} + \frac{2x\mu_k}{2\sigma_k^2} - \frac{\mu_k^2}{2\sigma_k^2}.
\end{aligned}$$

Since the $\log\left(\sqrt{2\pi}\right)$ term in the previous equation is independent of k, maximizing $p_k(X)$ is equivalent to maximizing

$$\delta_k(X) = \log(\pi_k) - \log(\sigma_k) - \frac{x^2}{2\sigma_k^2} + \frac{2x\mu_k}{2\sigma_k^2} - \frac{\mu_k^2}{2\sigma_k^2}.$$

Rearranging terms we see that

$$\delta_k(X) = -\frac{x^2}{2\sigma_k^2} + \frac{2x\mu_k}{2\sigma_k^2} - \frac{\mu_k^2}{2\sigma_k^2} + \log\left(\frac{\pi_k}{\sigma_k}\right).$$

Based on the leading term of the previous equation, we immediately see that the discrimant function for this case of QDA is quadratic.

Problem 4

Part a

On average, we would be using 10% of the data.

Part b

We would be using 10% of the data for each feature, so, in total, we would be using $0.1 \cdot 0.1 = 0.01 = 1\%$ of the data.

Part c

Extending the logic above, we would be predicting the test observation's response using $\frac{1}{10^p} = \frac{1}{10^{100}} = \frac{1}{10^{98}}\%$ of the observations.

Part d

As the number of predictors grows, there are fewer training observations near any given test observations. Then, given a sample size n, the number of neighbors of each observation will decrease as p increases.

Part e

Notice that a hypercube with side length x and p features will contain x^p percent of the observations, on average. Hence, in order to define a hypercube that will contain, on average, 10% of the obersations, we need to solve $x^p = 0.1$ in terms of x. That is, the hypercube must have side length $x = \sqrt[p]{0.1} = 0.1^{1/p}$. So for p=1,2,100, the hypercube will have side lengths

$$
\begin{aligned}
x &= \sqrt[1]{0.1} = 0.1 \\
x &= \sqrt[2]{0.1} \approx 0.316227766 \\
x &= \sqrt[100]{0.1} \approx 0.977237221,
\end{aligned}
$$

respectively. Clearly, as $p \to \infty$, $x \to 1$. That is, as we add more features to the model, we require a larger hypercube to contain an average of 10% of the observations.

Problem 5

Part a

If the true Bayes decision boundary is linear, we expect that LDA will perform better on both the training and test set. There is a possibility that QDA may perform better on the test data because it has higher variance within sample. That is, it will have a tendency to overfit the training data.

Part b

If the true Bayes decision boundary is non-linear, we would expect QDA to perform better on both the training and test set. Depending on the nature of the non-lineaerity, however, QDA still may not perform very well, and KNN may be the optimal choice.

Part c

As the sample size n increases, we would expect the test prediction accuracy of QDA to improve relative to LDA because there will be more data to estimate the parameters of the QDA model. That is, as n increases, we will get progressively better approximations of μ_k and σ_k for all k.

Part d

False. If the true Bayes decision boundary is linear, QDA may have a tendency to overfit the data. That is, it will have a high variance, without a comensurate improvement in bias relative to LDA. In contrast, LDA will have a relatively low variance in this case, without compensating bias.

Problem 6

Part a

The probability this student gets an A is given by:

$$\hat{p}(Y = 1) = \frac{\exp(-6 + (0.05 \cdot 40) + 1)}{1 + \exp(-6 + (0.05 \cdot 40) + 1)} \approx 0.0474259.$$

That is, this student has about a 4.74% chance of receiving an A in the class.

Part b

Using equation 4.3, we need

$$\frac{1/2}{1-1/2} = 1 = \exp\left(-6 + (0.05 \cdot x) + 1\right).$$

Taking the log of both sides we get that

$$\begin{aligned} 0 &= -6 + (0.05 \cdot x) + 1 \\ 5 &= 0.05x \\ x &= 100. \end{aligned}$$

That is, the student from part a would need to study 100 hours to have a 50% chance of receiving an A in the class.

Problem 7

Use equation 4.12:

$$\begin{aligned} p(dividend = 1|X = 4) &= \frac{0.8 \cdot \frac{1}{6\sqrt{2\pi}} \cdot \exp\left(-\frac{1}{2 \cdot 36}(4-10)^2\right)}{0.8 \cdot \frac{1}{6\sqrt{2\pi}} \cdot \exp\left(-\frac{1}{2 \cdot 36}(4-10)^2\right) + 0.2 \cdot \frac{1}{6\sqrt{2\pi}} \cdot \exp\left(-\frac{1}{2 \cdot 36}(4-0)^2\right)} \\ &= \frac{0.8 \cdot \exp\left(-\frac{1}{72} \cdot 36\right)}{0.8 \cdot \exp\left(-\frac{1}{72} \cdot 36\right) + 0.2 \cdot \exp\left(-\frac{1}{72} \cdot 16\right)} \\ &= \frac{0.8 \cdot \exp(-0.5)}{0.8 \cdot \exp(-0.5) + 0.2 \cdot \exp\left(-\frac{16}{72}\right)} \\ &\approx 0.751852. \end{aligned}$$

That is, there is about a 75.19% chance the company will issue a dividend.

Problem 8

It does not seem obvious that one method is superior to the other. Logistic regression has a high error rate on both the training and test data, so the Bayes decision boundary is likely nonlinear. Using KNN with $k = 1$, however, has likely led to overfitting on the test data. That is, the error rate on the training data is likely very low for the KNN model. With an average error of 18%, then the test error for KNN is probably over 30%. If this is the case, we should prefer logistic regression on new observations because it has a lower test error rate. We would do well to verify that this is the case, though.

Problem 9

Part a

With odds of 0.37, we have

$$\frac{p(X)}{1-p(X)} = 0.37,$$

so

$$p(X) = 0.37 - 0.37p(X).$$

Then

$$p(X)(1 + 0.37) = 0.37,$$

so

$$p(X) = \frac{0.37}{1.37} \approx 0.2701.$$

That is, about 27% of people with an odds of 0.37 will default on their credit card payment.

Part b

If an individual has a 16% chance of defaulting on their credit card payment, the odds they will default are given by:

$$\frac{0.16}{1 - 0.16} = \frac{0.16}{0.84} \approx 0.19048.$$

Applied

Problem 10

Part a

```r
#Import the data
  library(ISLR)
#Summary of the data
  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
##  Min.   :-18.1950   Min.   :-18.1950   Min.   :0.08747
##  1st Qu.: -1.1580   1st Qu.: -1.1660   1st Qu.:0.33202
##  Median :  0.2380   Median :  0.2340   Median :1.00268
##  Mean   :  0.1458   Mean   :  0.1399   Mean   :1.57462
##  3rd Qu.:  1.4090   3rd Qu.:  1.4050   3rd Qu.:2.05373
##  Max.   : 12.0260   Max.   : 12.0260   Max.   :9.32821
##      Today          Direction
##  Min.   :-18.1950   Down:484
##  1st Qu.: -1.1540   Up  :605
##  Median :  0.2410
##  Mean   :  0.1499
##  3rd Qu.:  1.4050
##  Max.   : 12.0260
```

```r
  names(Weekly)
```

```
## [1] "Year"      "Lag1"      "Lag2"      "Lag3"      "Lag4"      "Lag5"
## [7] "Volume"    "Today"     "Direction"
```

```r
str(Weekly)
```

```
## 'data.frame':    1089 obs. of  9 variables:
##  $ Year     : num  1990 1990 1990 1990 1990 1990 1990 1990 1990 1990 ...
##  $ Lag1     : num  0.816 -0.27 -2.576 3.514 0.712 ...
##  $ Lag2     : num  1.572 0.816 -0.27 -2.576 3.514 ...
##  $ Lag3     : num  -3.936 1.572 0.816 -0.27 -2.576 ...
##  $ Lag4     : num  -0.229 -3.936 1.572 0.816 -0.27 ...
##  $ Lag5     : num  -3.484 -0.229 -3.936 1.572 0.816 ...
##  $ Volume   : num  0.155 0.149 0.16 0.162 0.154 ...
##  $ Today    : num  -0.27 -2.576 3.514 0.712 1.178 ...
##  $ Direction: Factor w/ 2 levels "Down","Up": 1 1 2 2 2 1 2 2 2 1 ...
```

Part b

```r
#Fit logistic regression model
logit.fit1 <- glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume, data=Weekly, family=binomial)
summary(logit.fit1)
```

```
##
## 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 predictor (other than the constant) that appears to be significant is the Lag2 variable.

Part c

```
#Produce confusion matrix
  logit.probs1 <- predict(logit.fit1,type="response")
  logit.pred1 <- ifelse(logit.probs1>0.5, "Up", "Down")
  table(logit.pred1,Weekly$Direction)
```

```
##
## logit.pred1 Down  Up
##        Down   54  48
##        Up    430 557
```

```
#Overall fraction of correct predictions
  mean(logit.pred1==Weekly$Direction)
```

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

When the prediction is "Up," the model is correct $\frac{557}{430+557} = 56.43\%$ of the time. When the prediction is "Down," the model is correct $\frac{54}{54+48} = 52.94\%$ of the time. The model is more accurate when the prediction is "Up." That is, the model has a higher false negative rate than it does a false positive rate.

Part d

```
#Separate into training and test data
  Weekly.train <- subset(Weekly, Year<=2008)
  Weekly.test <- subset(Weekly, Year>=2009)
#Fit the model. Make predictions.
  logit.fit2 <- glm(Direction~Lag2, data=Weekly.train, family = binomial)
  logit.probs2 <- predict(logit.fit2, Weekly.test, type = "response")
#Make confusion matrix
  logit.pred2 <- ifelse(logit.probs2>0.5,"Up","Down")
  table(logit.pred2, Weekly.test$Direction)
```

```
##
## logit.pred2 Down Up
##        Down    9  5
##        Up     34 56
```

```
#Overall fraction correct
  mean(logit.pred2==Weekly.test$Direction)
```

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

Part e

```
#Import MASS Library
  library(MASS)
#Fit the model. Make predictions.
  lda.fit <- lda(Direction~Lag2, data=Weekly.train)
  lda.probs <- predict(lda.fit, Weekly.test, type = "response")
#Make confusion matrix
  lda.pred <- lda.probs$class
  table(lda.pred, Weekly.test$Direction)
```

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

```
#Overall fraction correct
  mean(lda.pred==Weekly.test$Direction)
```

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

Part f

```
#Fit the model. Make predictions.
  qda.fit <- qda(Direction~Lag2, data=Weekly.train)
  qda.probs <- predict(qda.fit, Weekly.test, type = "response")
#Make confusion matrix
  qda.pred <- qda.probs$class
  table(qda.pred, Weekly.test$Direction)
```

```
##
## qda.pred Down Up
##     Down    0  0
##     Up     43 61
```

```
#Overall fraction correct
  mean(qda.pred==Weekly.test$Direction)
```

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

Part g

```
#Import the class library
  library(class)
#Train model. Make predictions.
  set.seed(1)
  train.X <- as.matrix(Weekly.train$Lag2)
  test.X <- as.matrix(Weekly.test$Lag2)
  knn.pred <- knn(train.X,test.X,Weekly.train$Direction,k=1)
  table(knn.pred,Weekly.test$Direction)
```

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

```
#Overall fraction correct
    mean(knn.pred==Weekly.test$Direction)
```

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

Part h

The logistic regression and linear discriminant analysis models seem to perform the best on the test data. Each has an overall accuracy of about 62.5%.

Part i

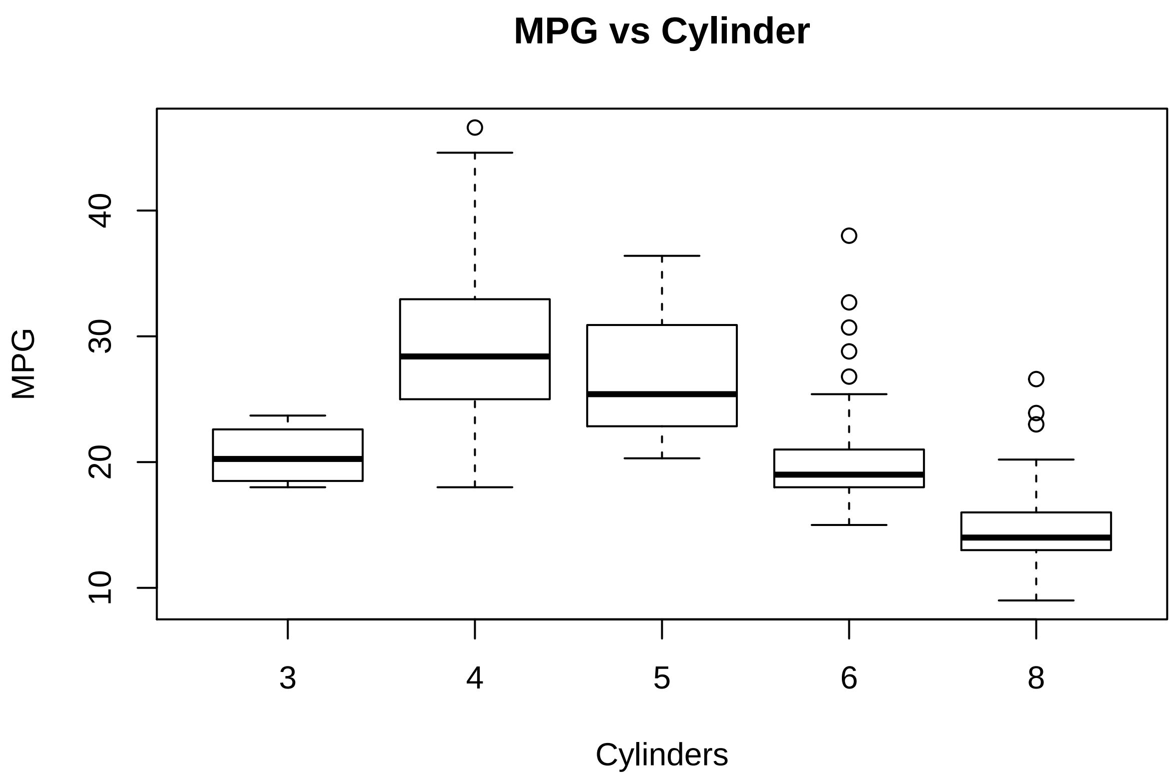
Problem 11

Part a

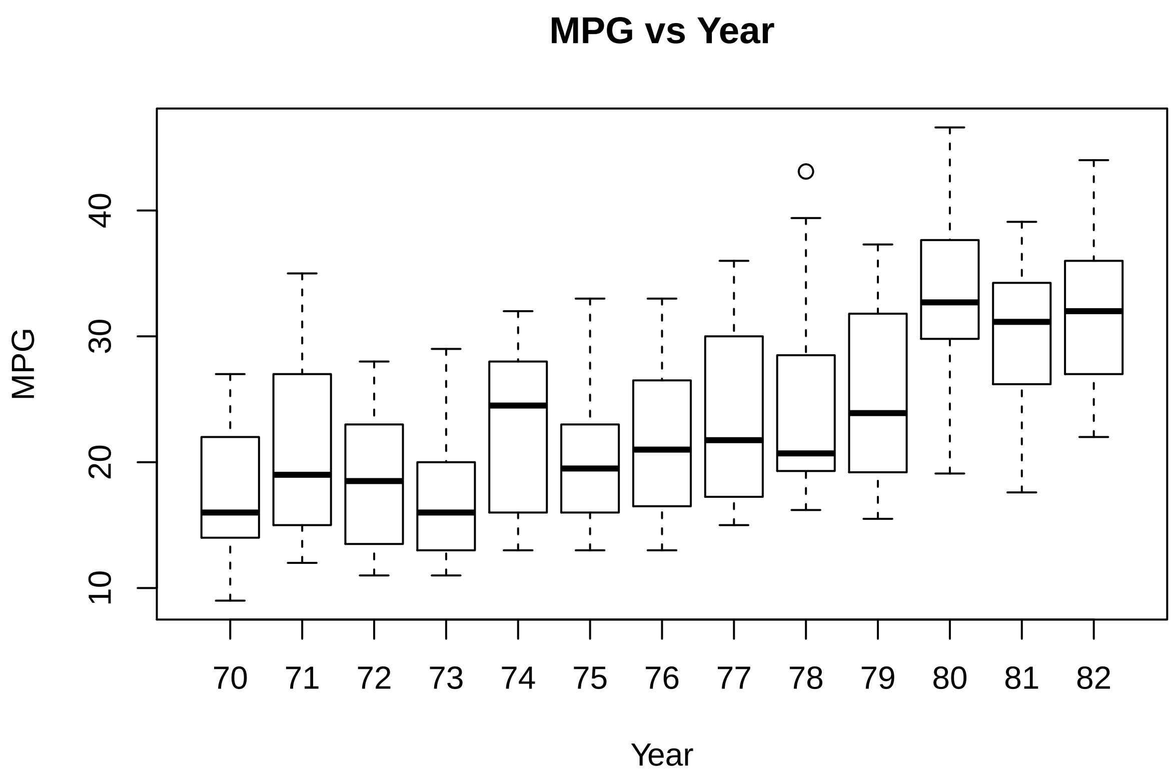
```r
#Import libraries
  library(ISLR)
#Create binary outcome variable
  Auto$mpg01 <- ifelse(Auto$mpg > median(Auto$mpg),1,0)
```

Part b

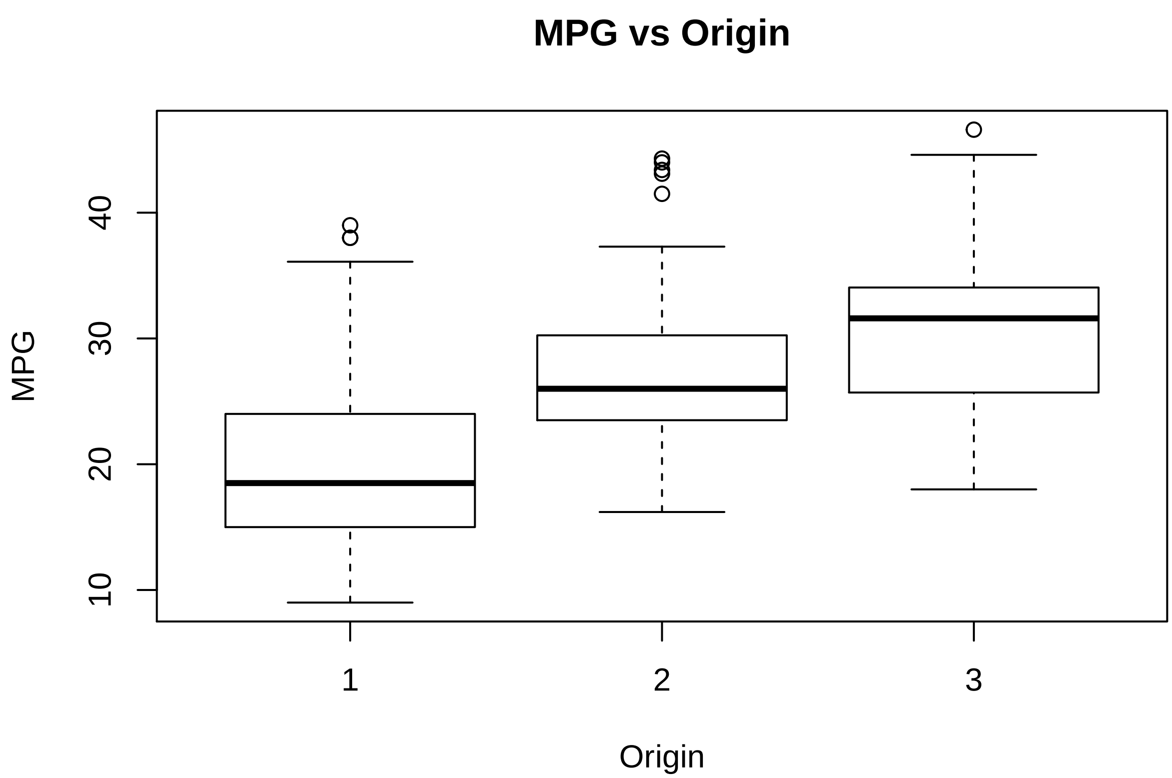
```r
boxplot(mpg~cylinders, data=Auto, main="MPG vs Cylinder", xlab="Cylinders",ylab="MPG")
```



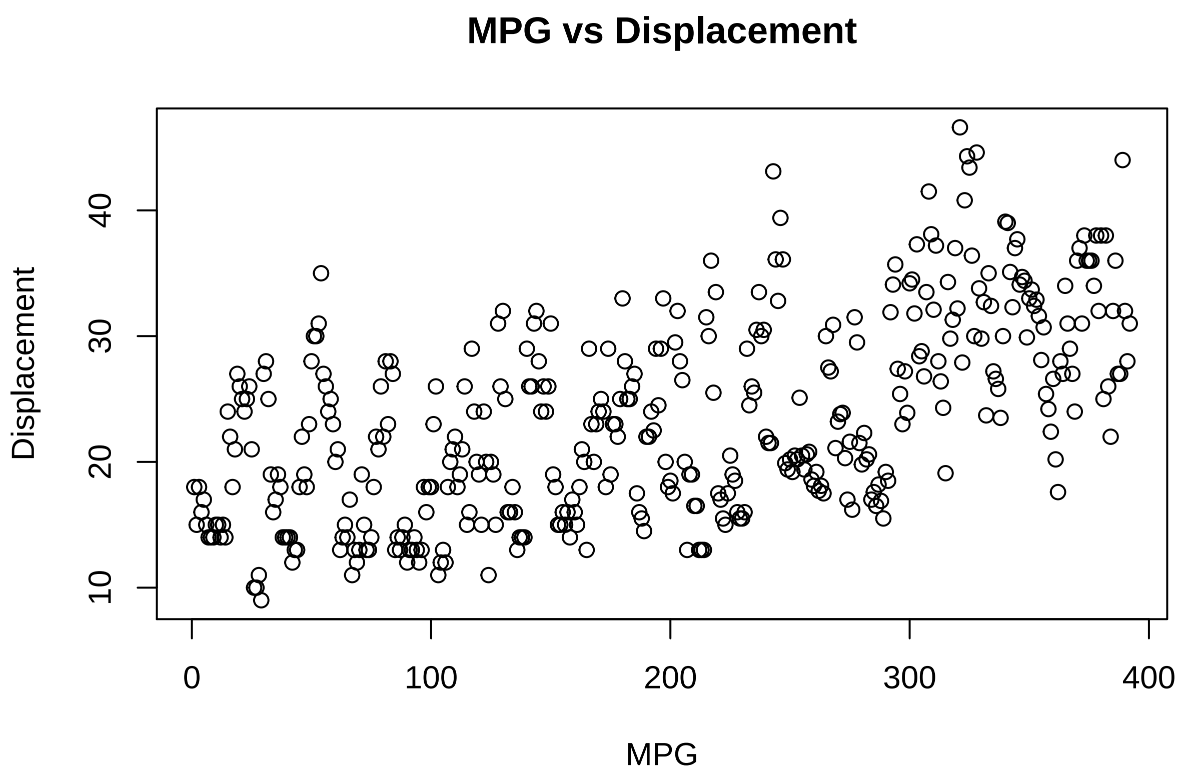

```r
boxplot(mpg~year, data=Auto, main="MPG vs Year", xlab="Year",ylab="MPG")
```



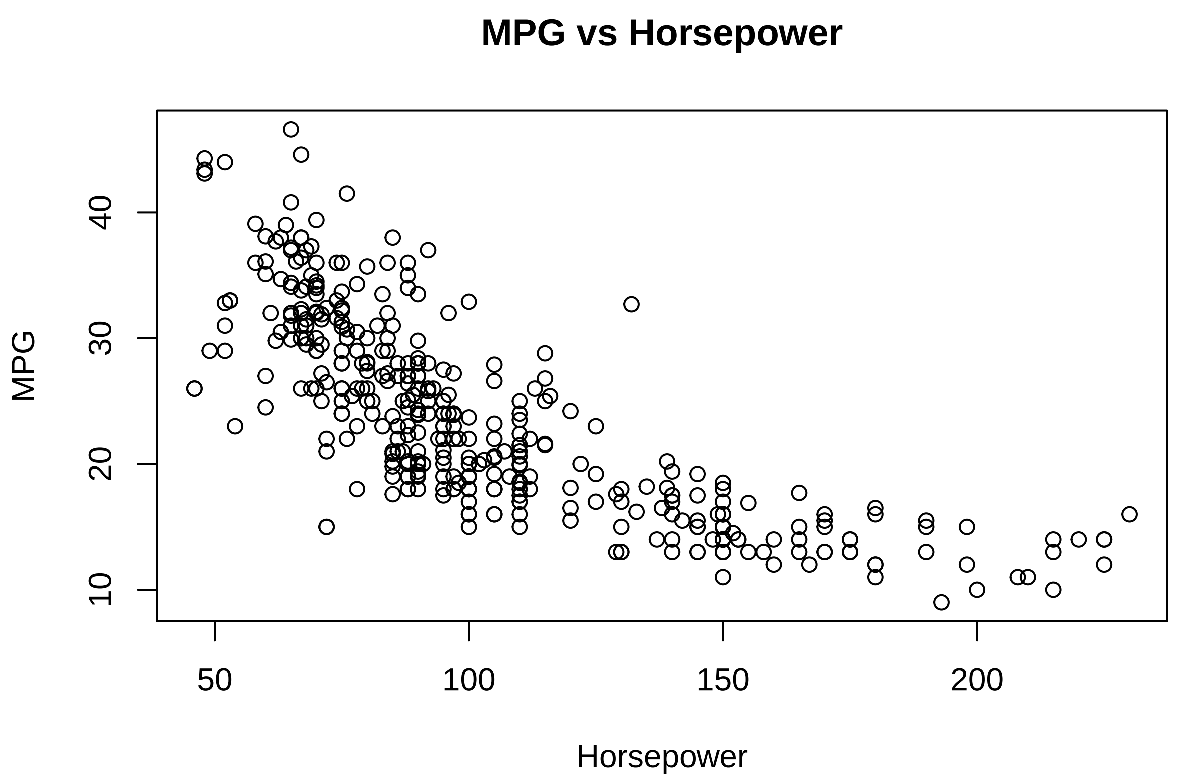

```
boxplot(mpg~origin, data=Auto, main="MPG vs Origin", xlab="Origin",ylab="MPG")
```



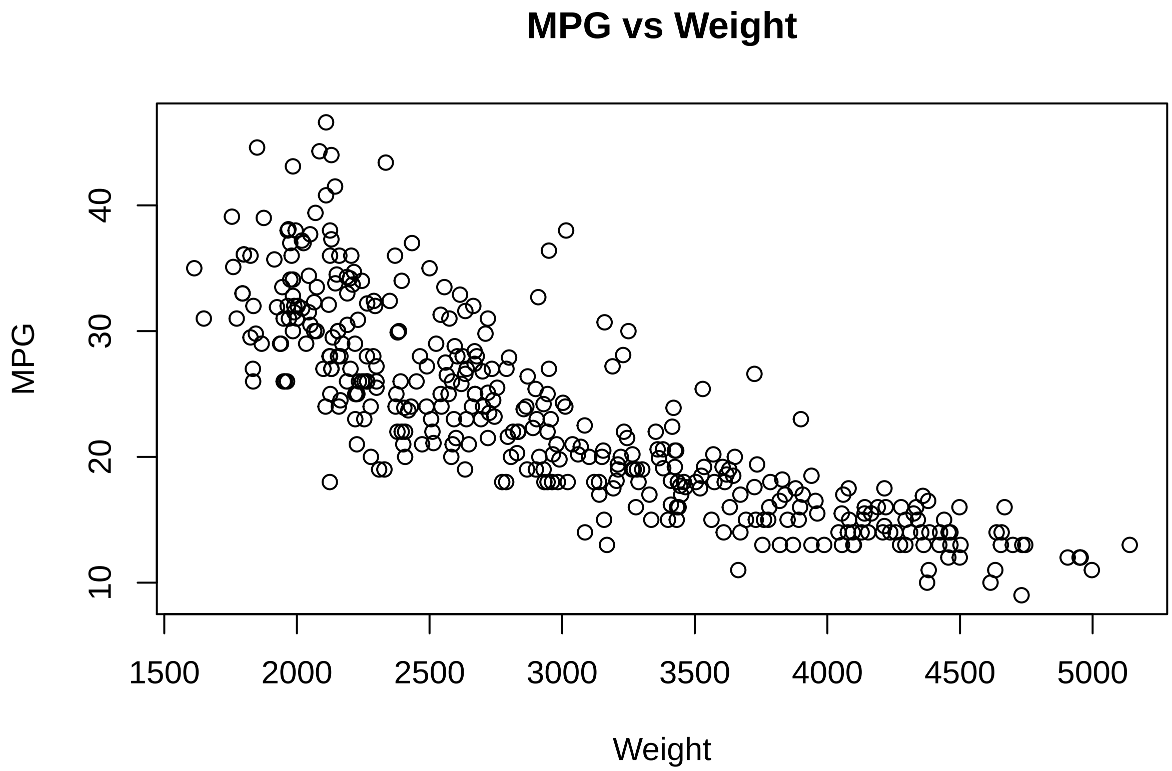

```
plot(Auto$mpg,Auto$diplacement,main="MPG vs Displacement",xlab="MPG",ylab="Displacement",type="p")
```



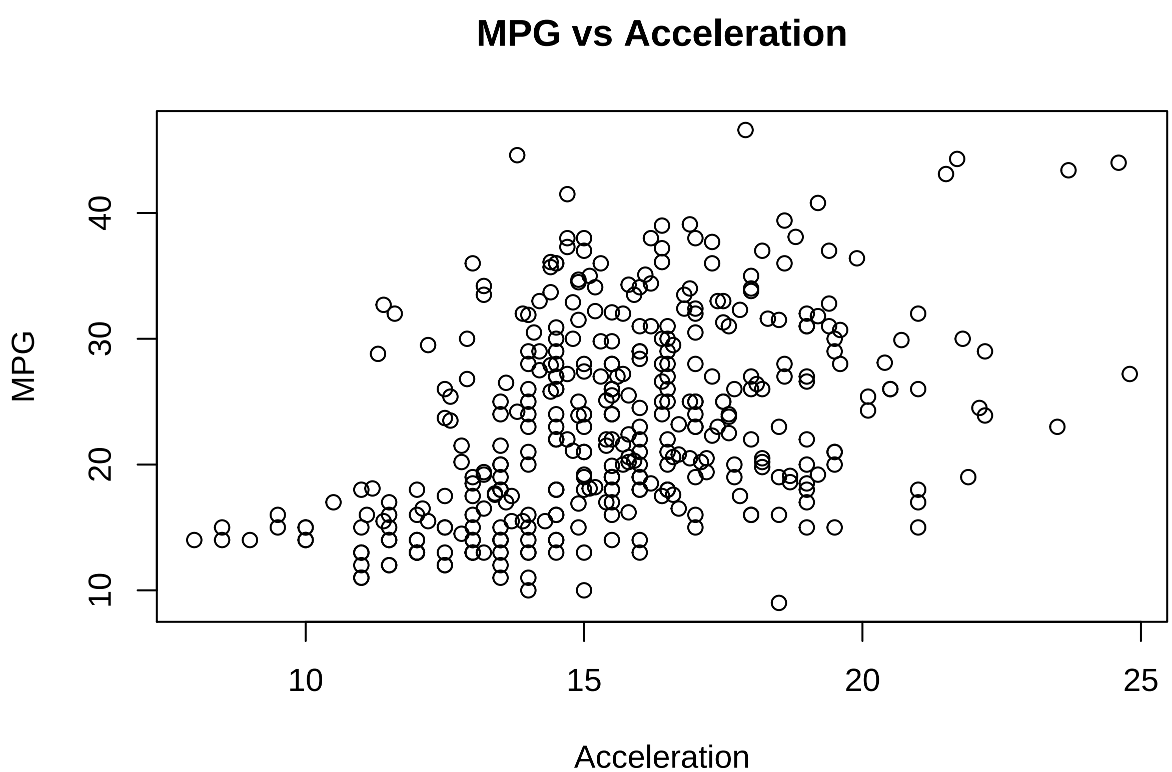

```
plot(Auto$horsepower,Auto$mpg,main="MPG vs Horsepower",ylab="MPG",xlab="Horsepower",type="p")
```




```
plot(Auto$weight,Auto$mpg,main="MPG vs Weight",ylab="MPG",xlab="Weight",type="p")
```




```r
plot(Auto$acceleration,Auto$mpg,main="MPG vs Acceleration",ylab="MPG",xlab="Acceleration",type="p")
```



Number of cylinders appears to have a quadratic relationship with mpg. There seems to be a slight upward trend in mpg over time. Country of origin seems to have a non-trivial effect on mpg. Displacement has a positive, though weak association with mpg. Horsepower and weight have a strong negative (and likely quadratic) relationship with mpg. Acceleration seems to have a positive and weak association with mpg.

Part c

```r
#Partition the data
  set.seed(123)
  sample_size <- floor(0.80 * nrow(Auto))
  train_ind <- sample(seq_len(nrow(Auto)), size=sample_size, replace=FALSE)
  train <- Auto[train_ind,]
  test <- Auto[-train_ind,]
```

Part d

```r
#Fit LDA Model
  library(MASS)
  lda.fit <- lda(mpg01~cylinders+displacement+horsepower+weight+acceleration+year+origin, data=train)
  lda.pred <- predict(lda.fit, test)
  lda.class <- lda.pred$class
  table(lda.class, test$mpg01)
```

```
##
## lda.class  0  1
##         0 36  1
##         1  8 34
```

```r
#Compute the test error
  (8+1)/(8+1+34+36)
```

```
## [1] 0.1139241
```

```r
  mean(lda.class != test$mpg01)
```

```
## [1] 0.1139241
```

The test error for LDA is about 11.39%.

Part e

```r
#Fit QDA Model
  qda.fit <- qda(mpg01~cylinders+displacement+horsepower+weight+acceleration+year+origin, data=train)
  qda.pred <- predict(qda.fit, test)
  qda.class <- qda.pred$class
  table(qda.class, test$mpg01)
```

```
##
## qda.class  0  1
##         0 39  2
##         1  5 33
```

```r
#Compute the test error
  (5+2)/(5+2+33+39)
```

```
## [1] 0.08860759
```

```r
  mean(qda.class != test$mpg01)
```

```
## [1] 0.08860759
```

The test error for QDA is about 8.86%.

Part f

```
#Fit Logistic Regression Model
  logit.fit <- glm(mpg01~cylinders+displacement+horsepower+weight+acceleration+year+origin, data=train,
  logit.probs <- predict(logit.fit, test, type="response")
  logit.class <- ifelse(logit.probs>0.5,1,0)
  table(logit.class, test$mpg01)
```

```
##
## logit.class  0  1
##           0 39  4
##           1  5 31
```

```
#Compute the test error
  (5+4)/(5+4+39+31)
```

```
## [1] 0.1139241
```

```
  mean(logit.class != test$mpg01)
```

```
## [1] 0.1139241
```

The test error for logistic regression is about 11.39%, the same as for LDA.

Part g

```
#Modify data for KNN fit
  library(class)
  train.X <- cbind(train$cylinders, train$displacement, train$horsepower, train$weight, train$accelerat
  test.X <- cbind(test$cylinders, test$displacement, test$horsepower, test$weight, test$acceleration, te
#Fit model with k=1
  knn.pred1 <- knn(train.X, test.X, train$mpg01, k=1)
  table(knn.pred1, test$mpg01) #Confusion matrix for k=1
```

```
##
## knn.pred1  0  1
##         0 37  3
##         1  7 32
```

```
  mean(knn.pred1 != test$mpg01) #Test error for k=1
```

```
## [1] 0.1265823
```

```
#Fit model with k=5
  knn.pred5 <- knn(train.X, test.X, train$mpg01, k=5)
  table(knn.pred5, test$mpg01) #Confusion matrix for k=5
```

```
##
## knn.pred5  0  1
##         0 37  6
##         1  7 29
```

```
  mean(knn.pred5 != test$mpg01) #Test error for k=5
```

```
## [1] 0.164557
```

```r
#Fit model with k=10
  knn.pred10 <- knn(train.X, test.X, train$mpg01, k=10)
  table(knn.pred10, test$mpg01) #Confusion matrix for k=10
```

```
##
## knn.pred10  0  1
##          0 37  3
##          1  7 32
```

```r
  mean(knn.pred10 != test$mpg01) #Test error for k=10
```

```
## [1] 0.1265823
```

```r
#Fit model with k=25
  knn.pred25 <- knn(train.X, test.X, train$mpg01, k=25)
  table(knn.pred25, test$mpg01) #Confusion matrix for k=25
```

```
##
## knn.pred25  0  1
##          0 37  4
##          1  7 31
```

```r
  mean(knn.pred25 != test$mpg01) #Test error for k=25
```

```
## [1] 0.1392405
```

```r
#Fit model with k=50
  knn.pred50 <- knn(train.X, test.X, train$mpg01, k=50)
  table(knn.pred50, test$mpg01) #Confusion matrix for k=50
```

```
##
## knn.pred50  0  1
##          0 37  2
##          1  7 33
```

```r
  mean(knn.pred50 != test$mpg01) #Test error for k=50
```

```
## [1] 0.1139241
```

```r
#Fit model with k=100
  knn.pred100 <- knn(train.X, test.X, train$mpg01, k=100)
  table(knn.pred100, test$mpg01) #Confusion matrix for k=100
```

```
##
## knn.pred100  0  1
##           0 34  2
##           1 10 33
```

```r
  mean(knn.pred100 != test$mpg01) #Test error for k=100
```

```
## [1] 0.1518987
```

```r
#Fit model with k=200
  knn.pred200 <- knn(train.X, test.X, train$mpg01, k=200)
  table(knn.pred200, test$mpg01) #Confusion matrix for k=200
```

```
##
## knn.pred200  0  1
##           0 32  1
##           1 12 34
```

```r
  mean(knn.pred200 != test$mpg01) #Test error for k=200
```

```
## [1] 0.164557
```

The value of k with the smallest test error rate appears to be around $k = 50$.

Problem 12

Part a

```r
#Write the Power() function
  Power <- function(){
    return(2^3)
  }
  Power() #Answer should be 8
```

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

Part b

```r
#Write the Power2() function
  Power2 <- function(x,a){
    return(x^a)
  }
  Power2(3,8) #Answer should be 6561
```

```
## [1] 6561
```

Part c

```r
Power2(10,3) #Answer should be 1000
```

```
## [1] 1000
```

```r
Power2(8,17) #Answer should be 2251799813685248
```

```
## [1] 2.2518e+15
```

```r
Power2(131,3) #Answer should be 2248091
```

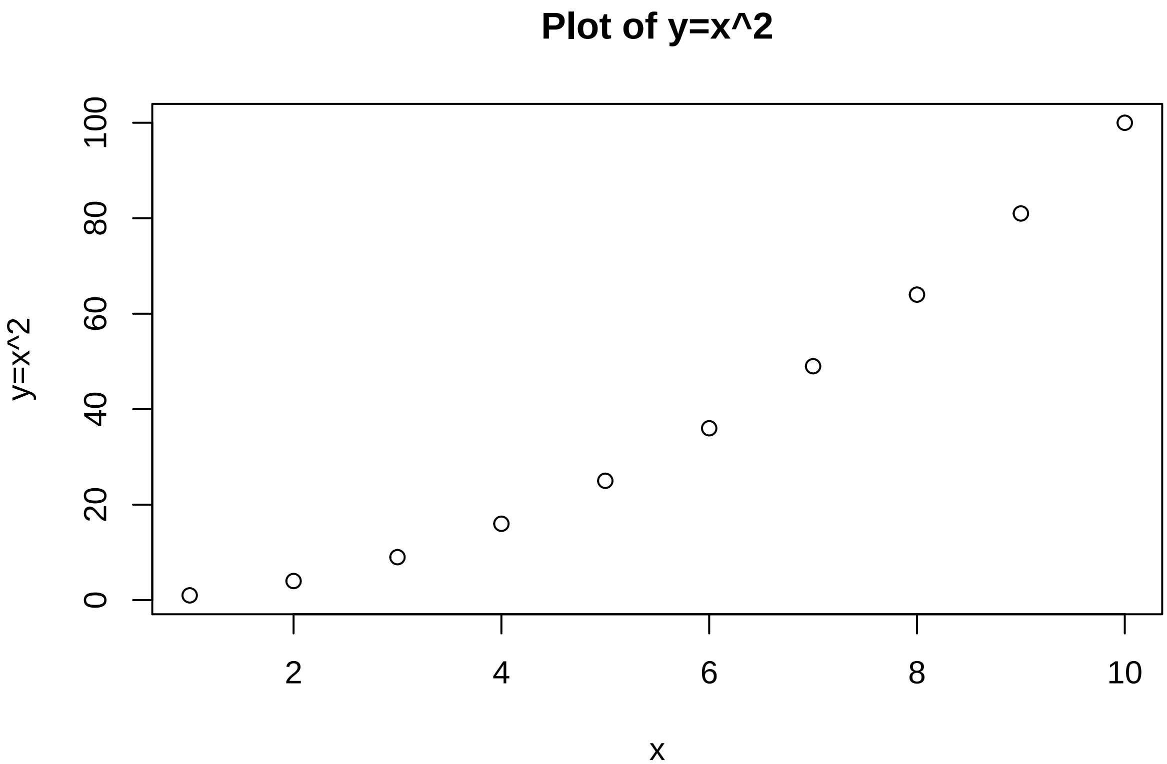
```
## [1] 2248091
```

Part d

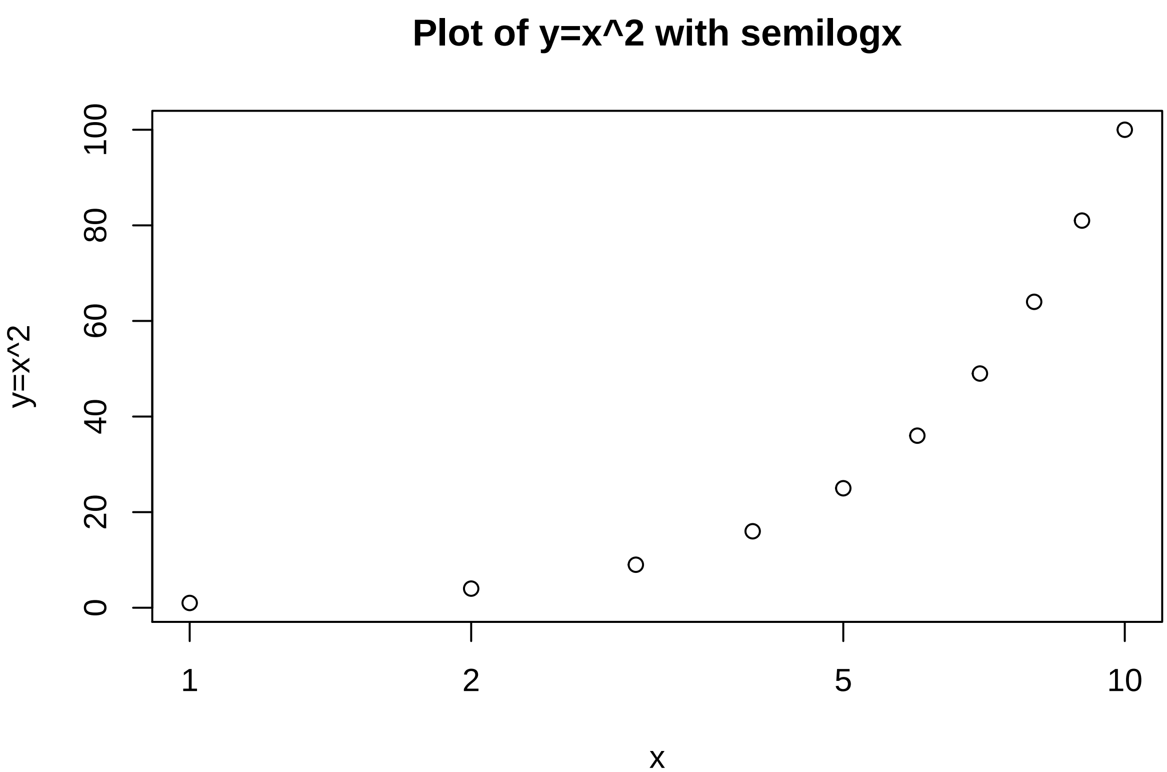
```r
#Write the Power3() function
  Power3 <- function(x,a){
    result <- x^a
    return(result)
  }
```

Part e

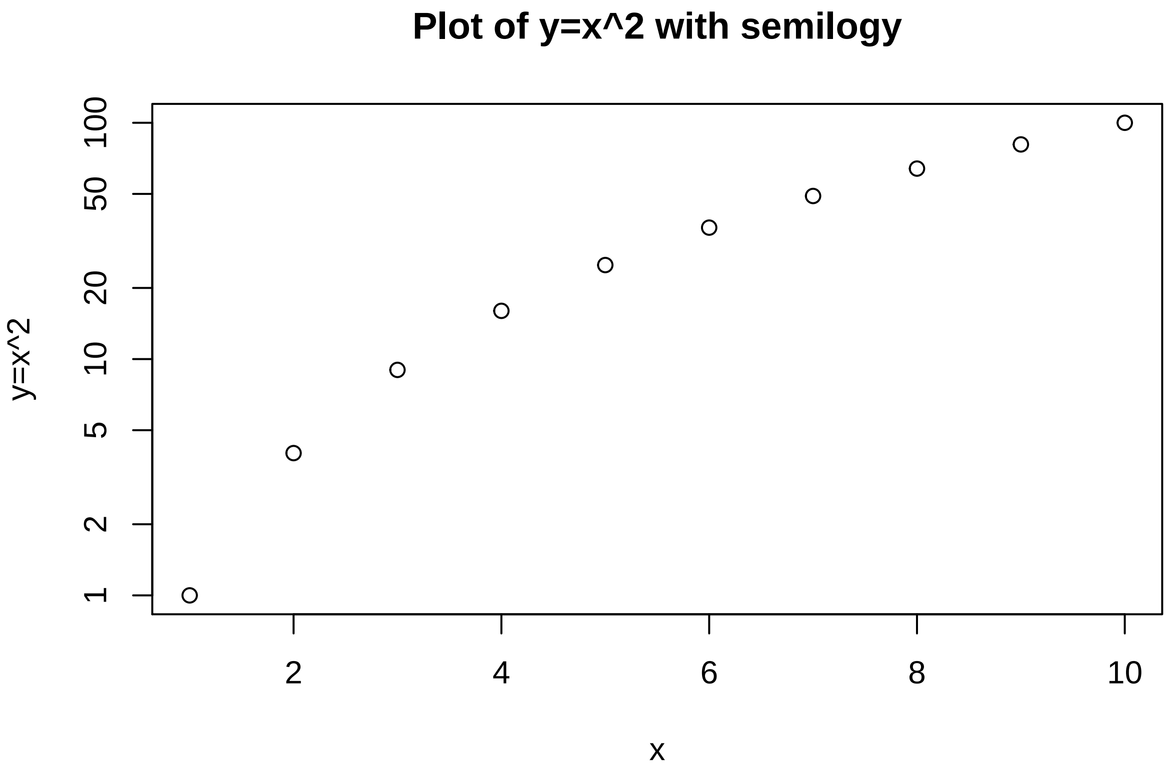
```
plot(1:10, Power3(1:10,2), xlab="x", ylab="y=x^2", main="Plot of y=x^2")
```



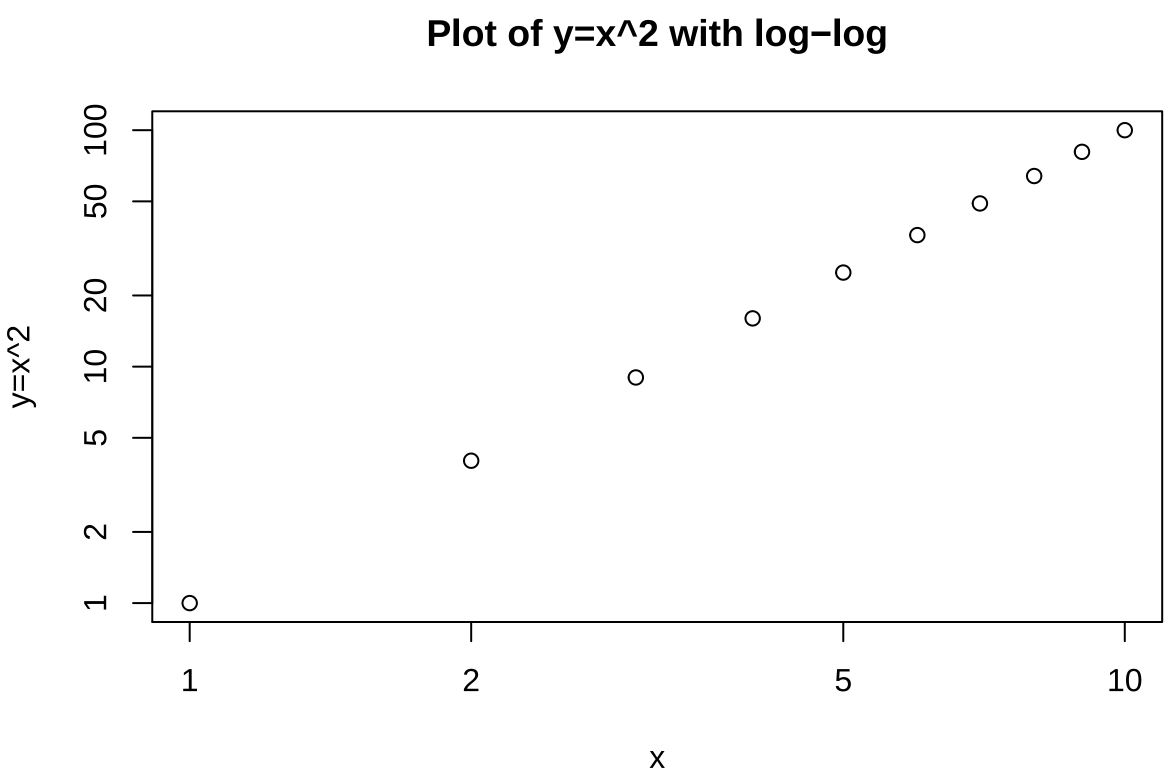

```
plot(1:10, Power3(1:10,2), xlab="x", ylab="y=x^2", main="Plot of y=x^2 with semilogx", log="x")
```



```r
plot(1:10, Power3(1:10,2), xlab="x", ylab="y=x^2", main="Plot of y=x^2 with semilogy", log="y")
```

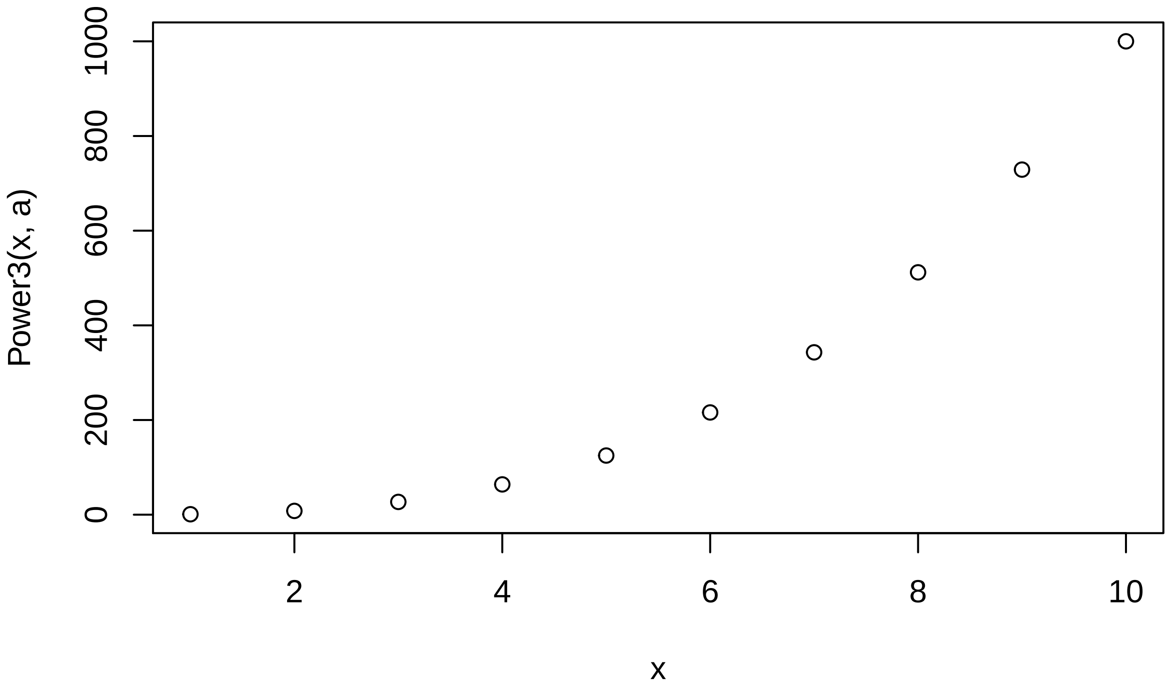



```r
plot(1:10, Power3(1:10,2), xlab="x", ylab="y=x^2", main="Plot of y=x^2 with log-log", log="xy")
```



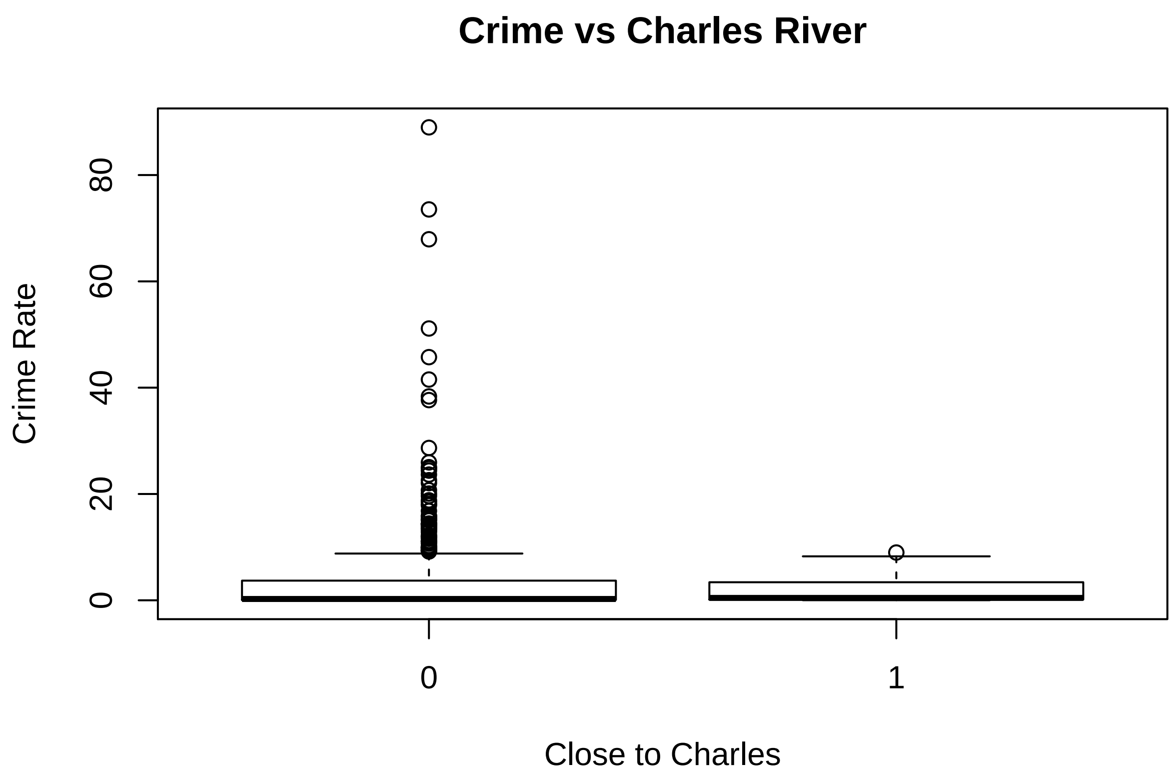
Part f

```r
#Write the PlotPower() function
  PlotPower <- function(x,a){
    plot(x,Power3(x,a), xlab="x")
  }
  PlotPower(1:10,3)
```

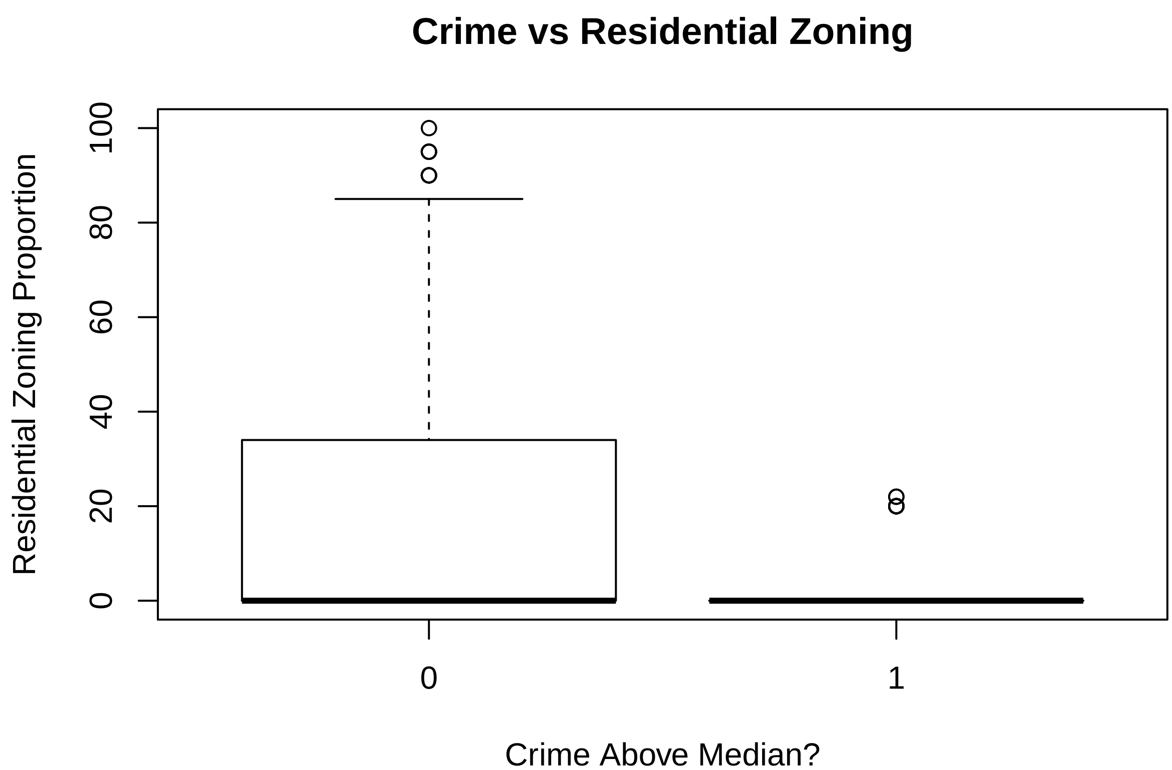


Problem 13

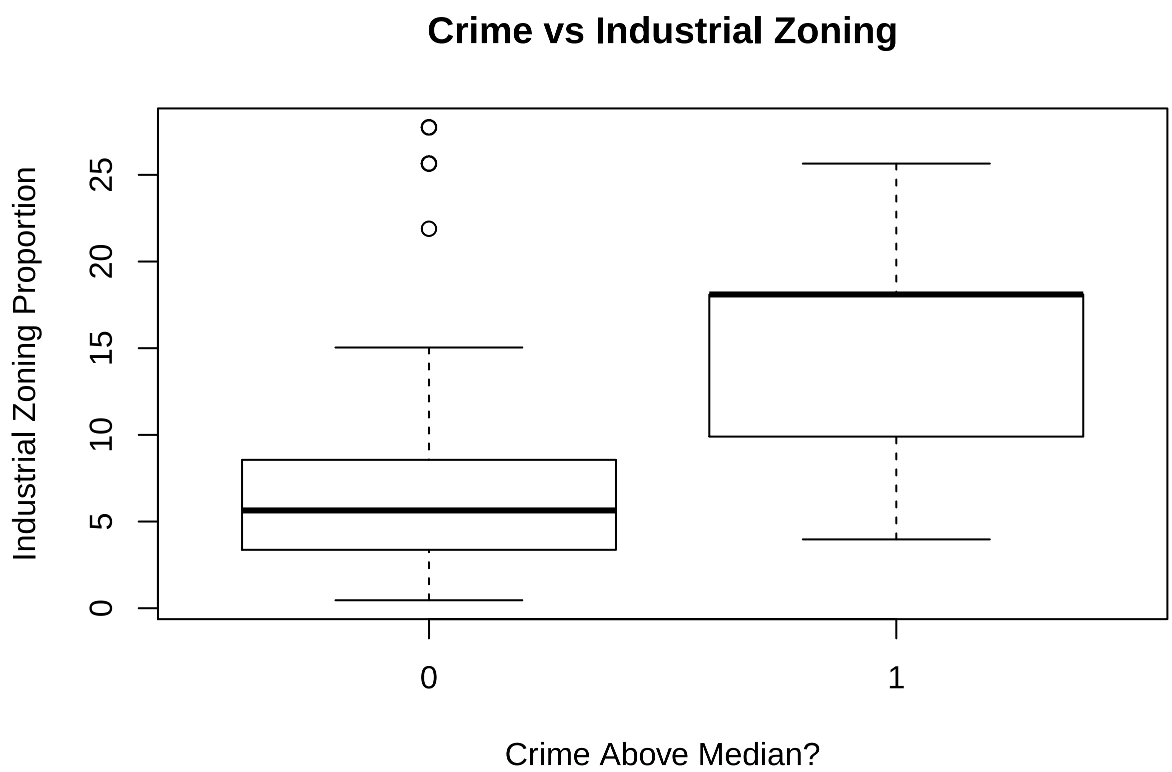
```r
#Import libraries
  library(MASS)
#Create binary outcome variable
  Boston$crim01 <- ifelse(Boston$crim > median(Boston$crim),1,0)
#Determine relevant predictors
  boxplot(crim~chas, data=Boston, main="Crime vs Charles River", xlab="Close to Charles", ylab="Crime Ra
```



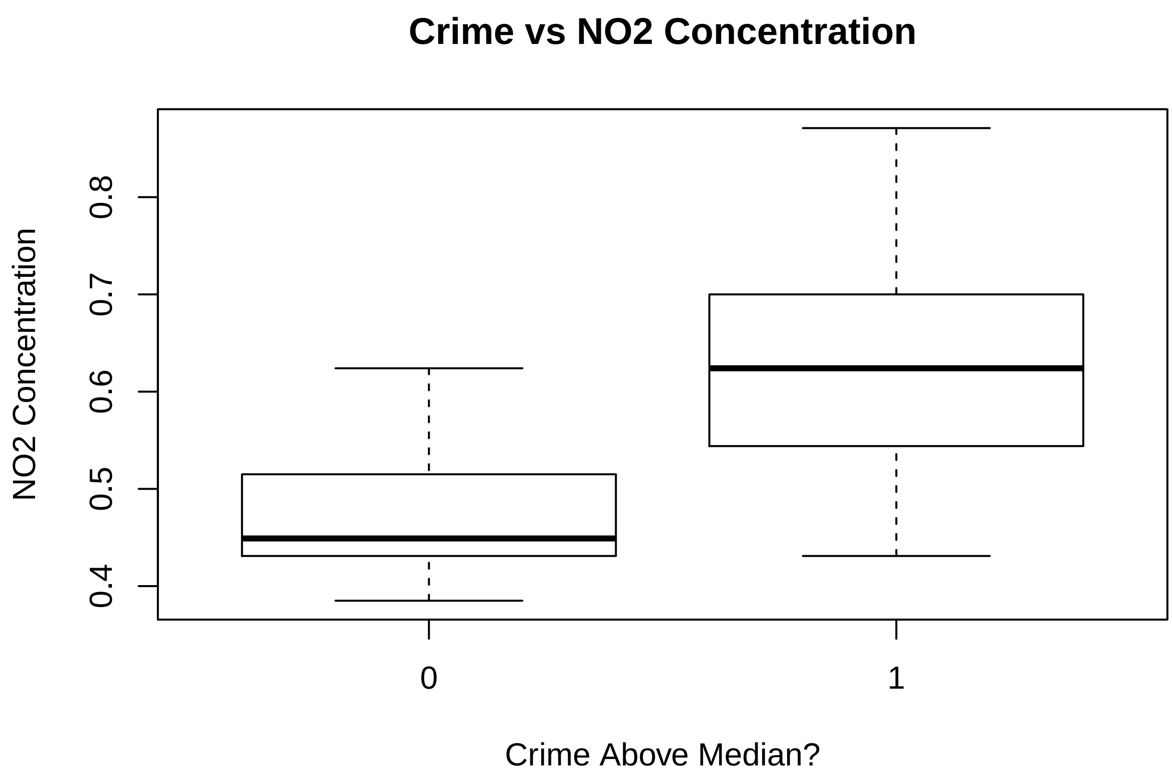

```
boxplot(zn~crim01, data=Boston, main="Crime vs Residential Zoning", xlab="Crime Above Median?", ylab=
```



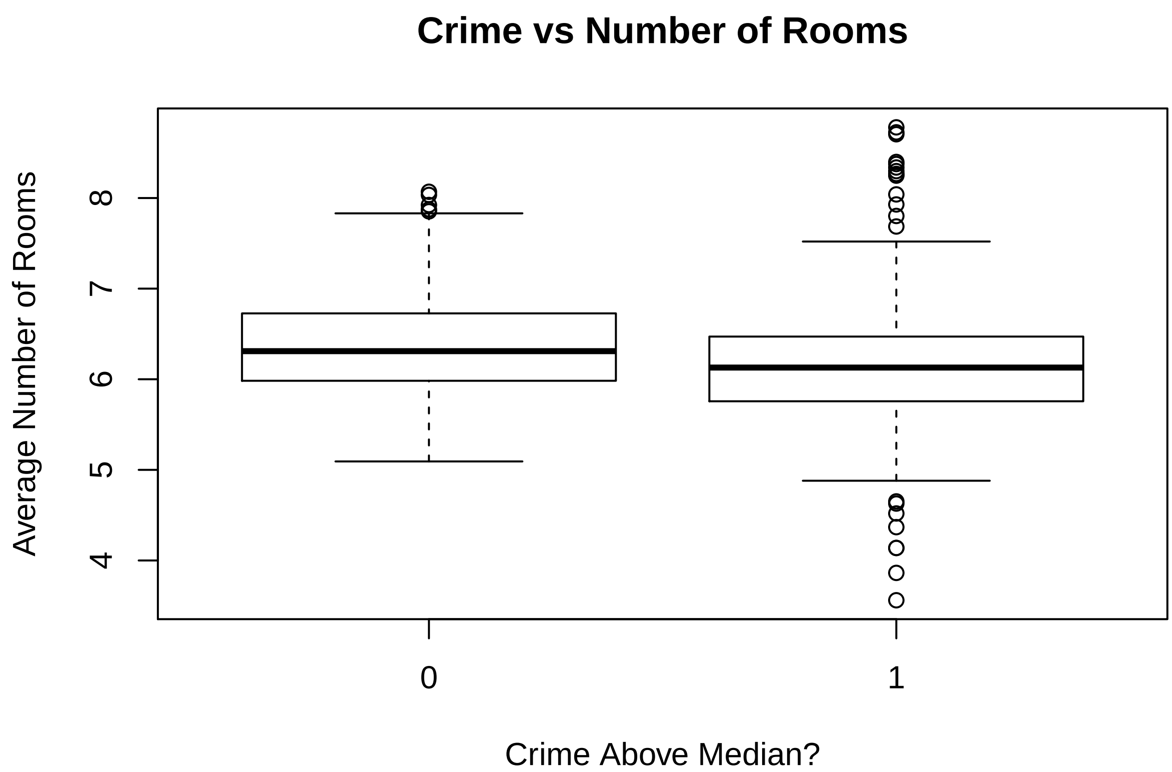

```
boxplot(indus~crim01, data=Boston, main="Crime vs Industrial Zoning",  xlab="Crime Above Median?", yla
```



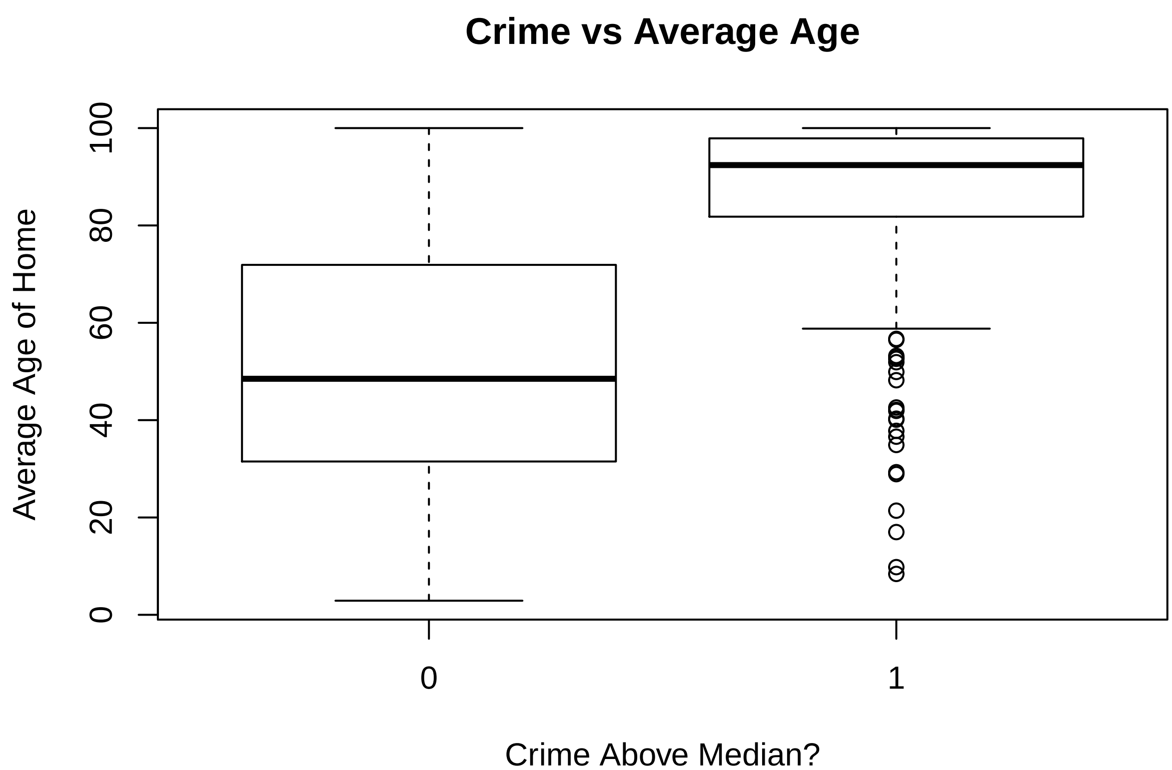

```
boxplot(nox~crim01, data=Boston, main="Crime vs NO2 Concentration", xlab="Crime Above Median?", ylab=
```



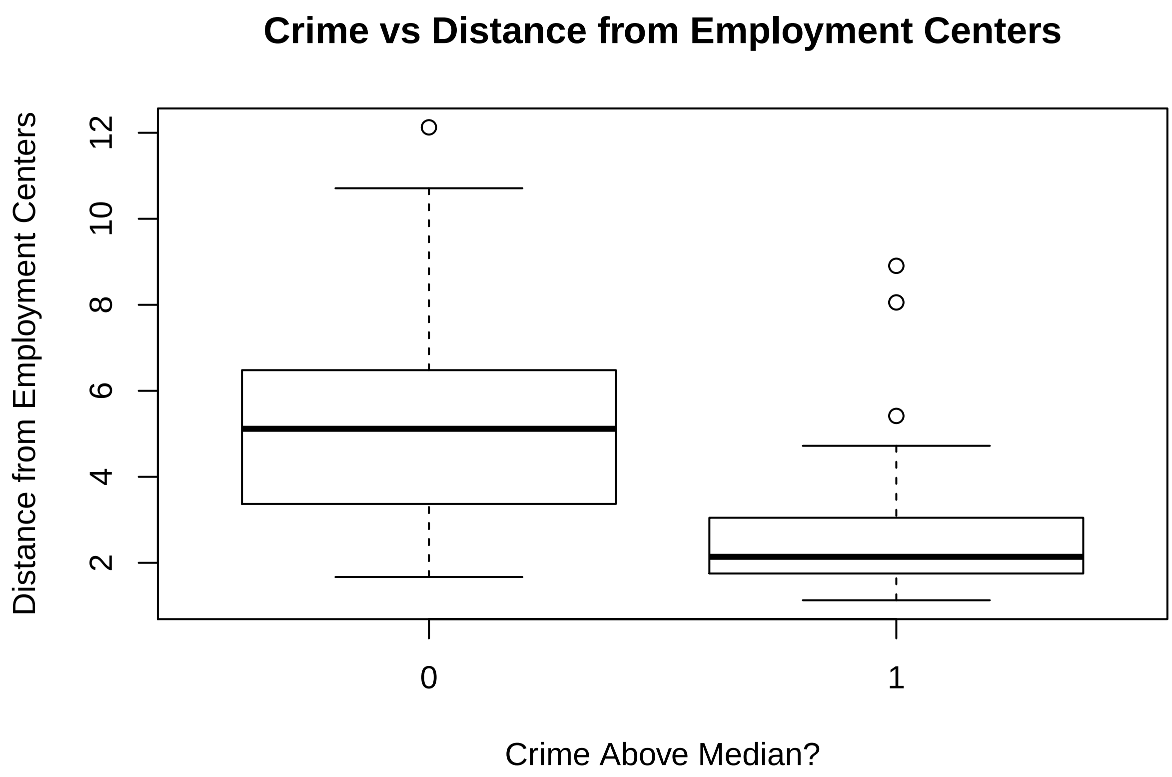

```
boxplot(rm~crim01, data=Boston, main="Crime vs Number of Rooms", xlab="Crime Above Median?", ylab="Av
```



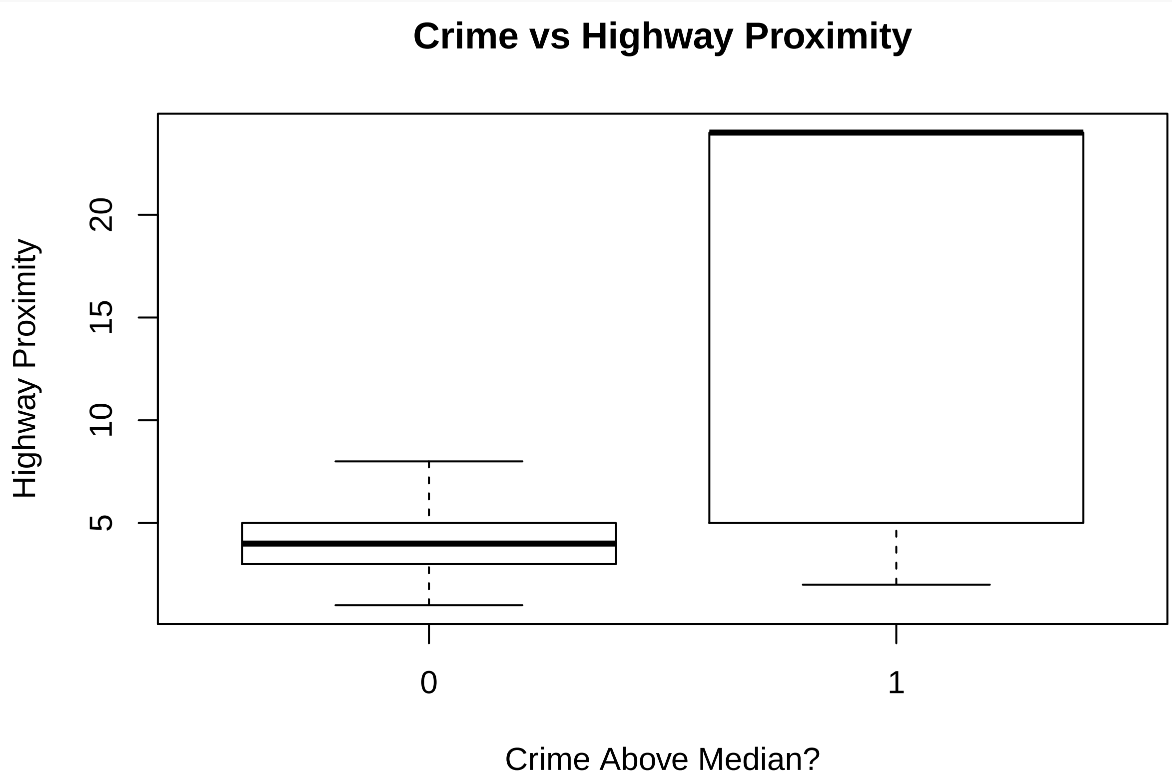

```
boxplot(age~crim01, data=Boston, main="Crime vs Average Age", xlab="Crime Above Median?", ylab="Averag
```



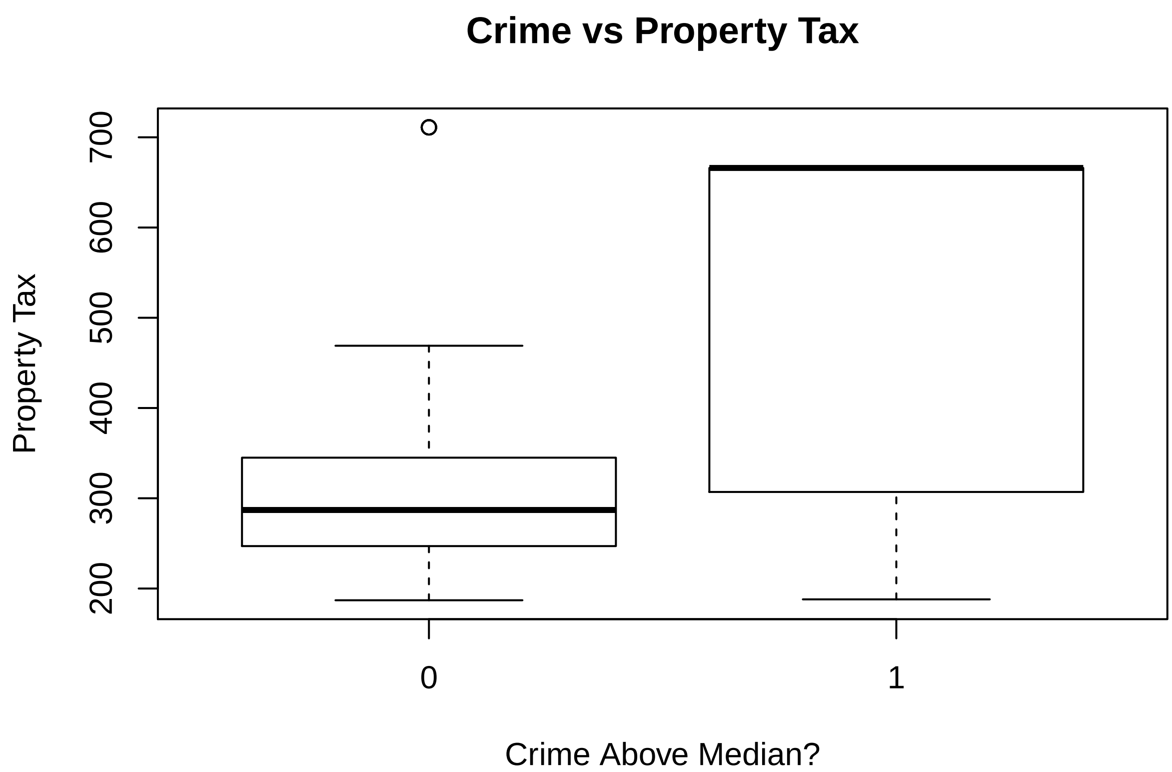

```
boxplot(dis~crim01, data=Boston, main="Crime vs Distance from Employment Centers", xlab="Crime Above M
```



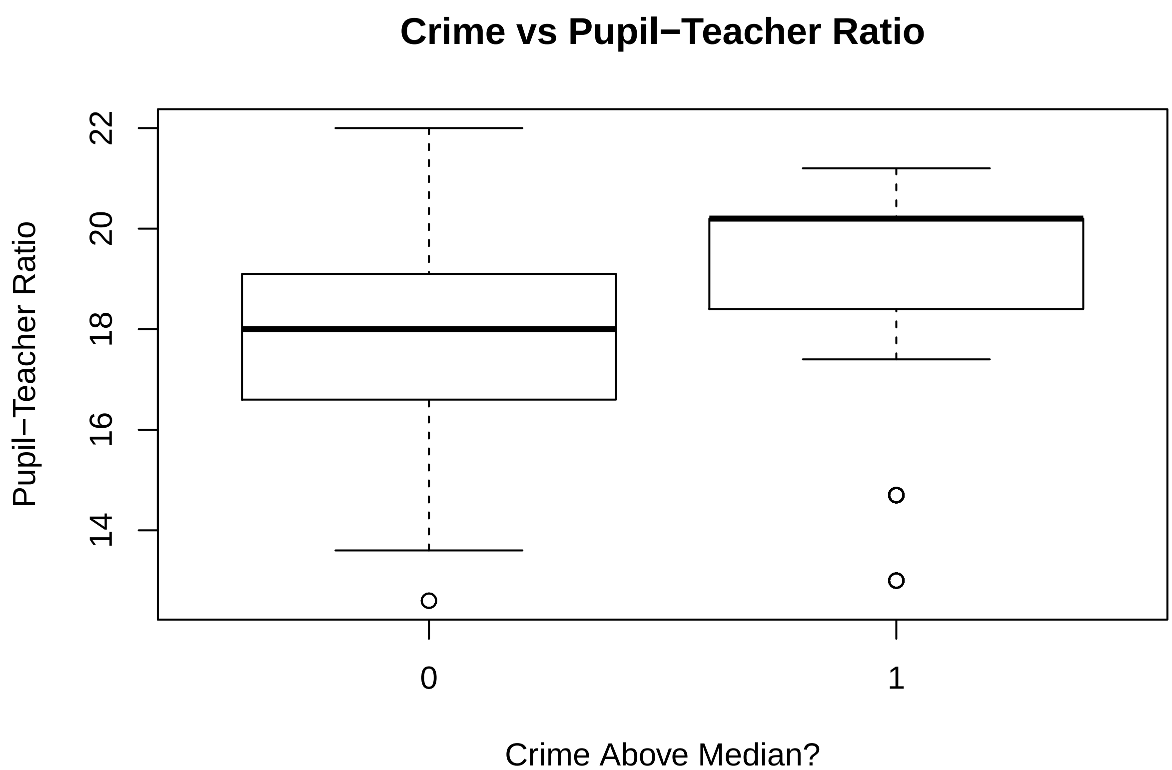

```
boxplot(rad~crim01, data=Boston, main="Crime vs Highway Proximity", xlab="Crime Above Median?", ylab=
```



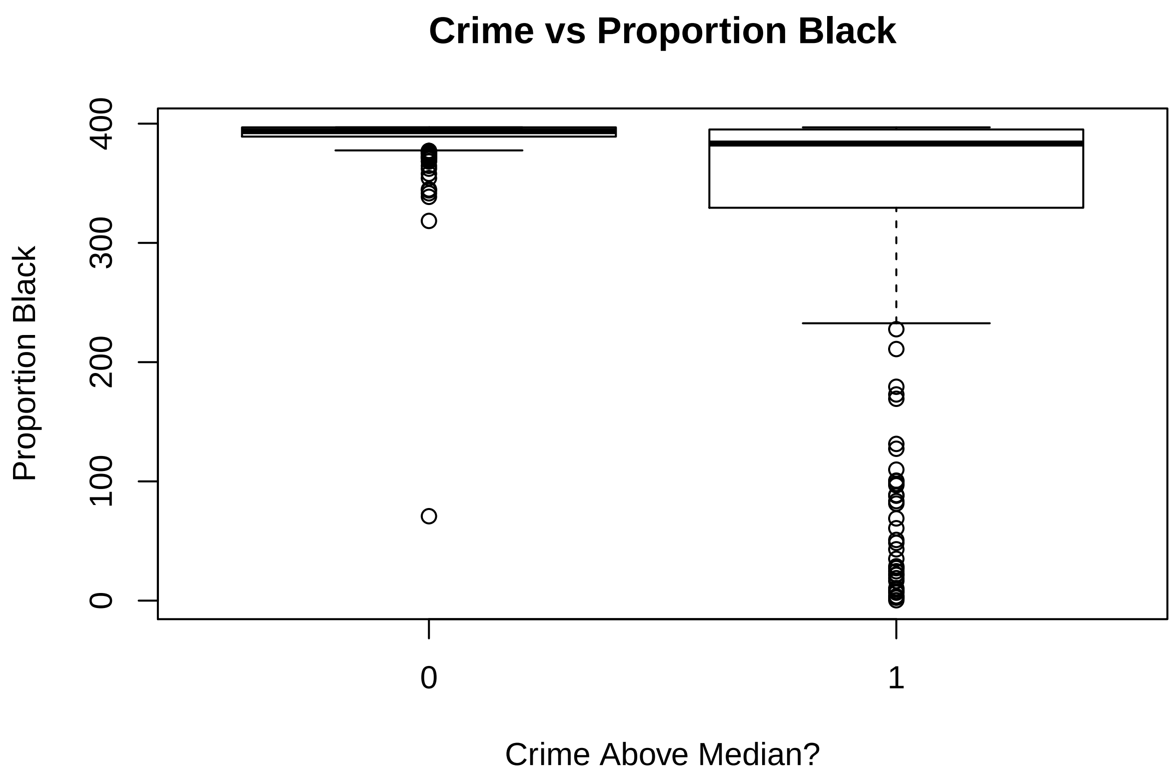

```
boxplot(tax~crim01, data=Boston, main="Crime vs Property Tax", xlab="Crime Above Median?", ylab="Prope
```



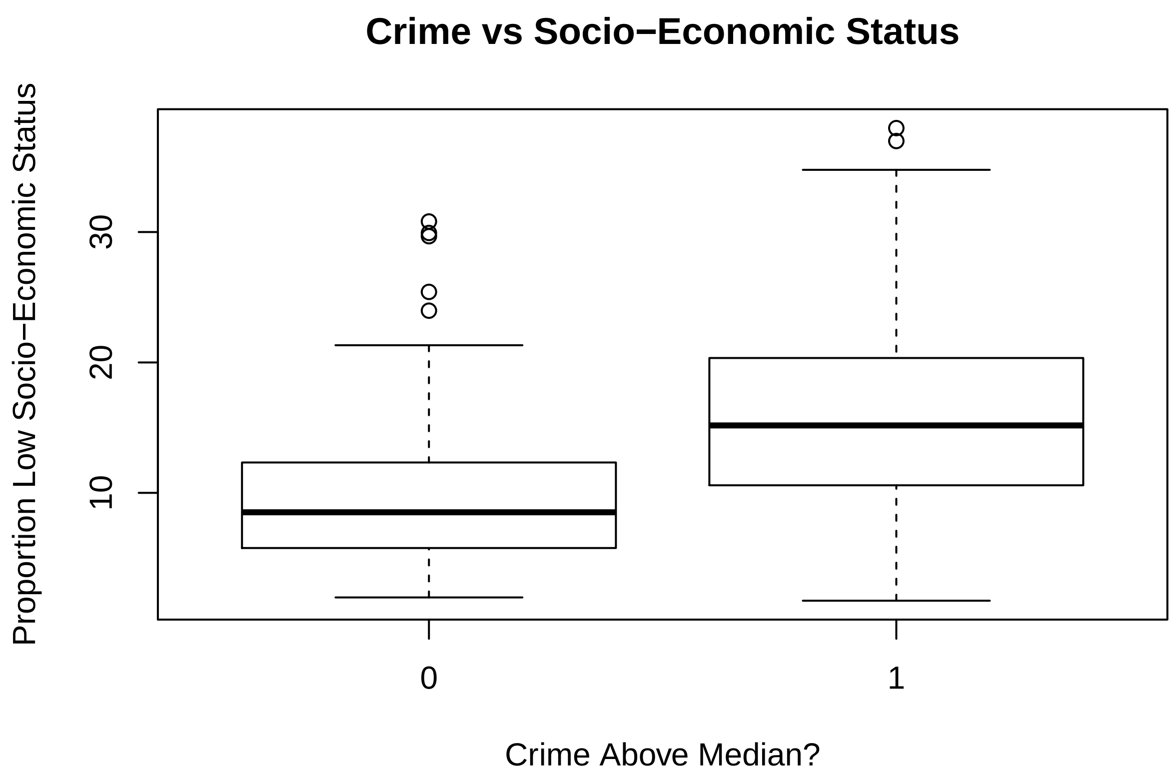

```
boxplot(ptratio~crim01,  data=Boston, main="Crime vs Pupil-Teacher Ratio", xlab="Crime Above Median?"
```



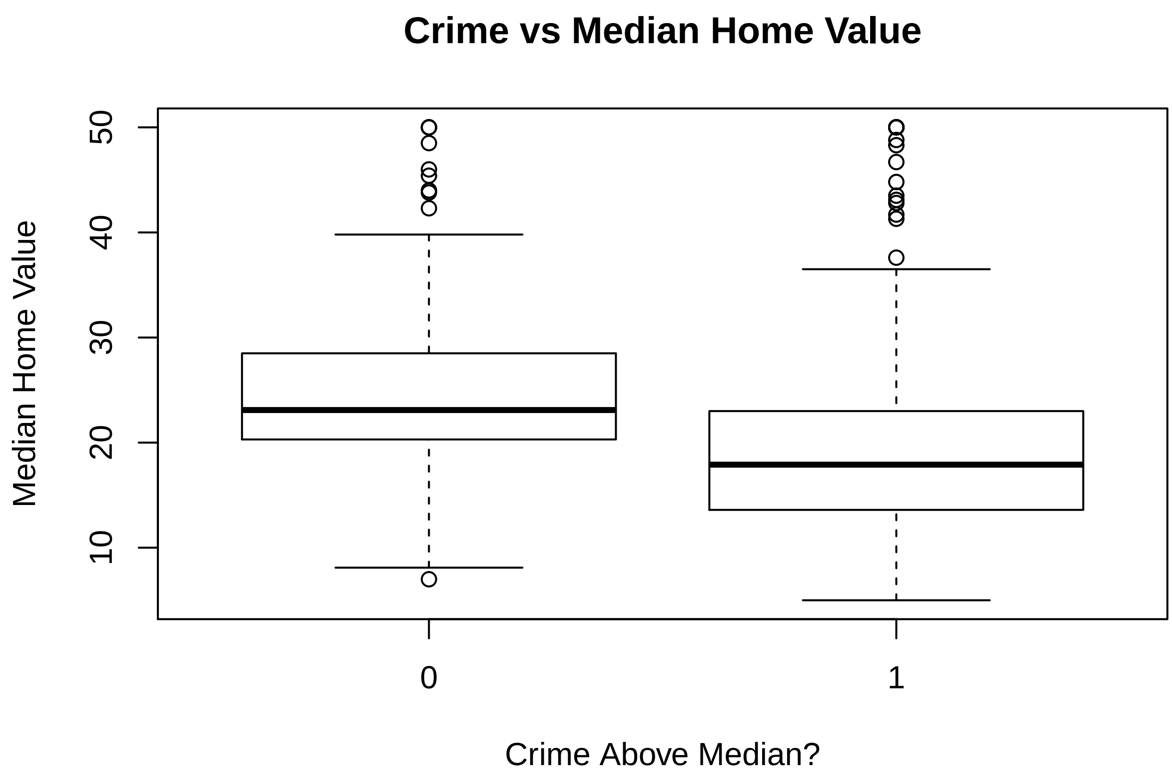

```
boxplot(black~crim01, data=Boston, main="Crime vs Proportion Black", xlab="Crime Above Median?", ylab
```




```
boxplot(lstat~crim01, data=Boston, main="Crime vs Socio-Economic Status", xlab="Crime Above Median?",
```




```
boxplot(medv~crim01, data=Boston, main="Crime vs Median Home Value", xlab="Crime Above Median?", ylab=
```




```
#Sort by variable's correlation with the crim variable
  sort(cor(Boston)[1,])
```

```
##        medv       black         dis          rm          zn        chas
## -0.38830461 -0.38506394 -0.37967009 -0.21924670 -0.20046922 -0.05589158
##     ptratio         age       indus      crim01         nox       lstat
##  0.28994558  0.35273425  0.40658341  0.40939545  0.42097171  0.45562148
##         tax         rad        crim
##  0.58276431  0.62550515  1.00000000
```

Going to fit two models each. The first will use the predictors medv, lstat, indus, nox, age, dis, rad, tax, and ptratio. The second will use just rad and tax, the only two with a correlation coefficient greater that 0.5.

```
#Partition the data
  set.seed(479)
  sample_size <- floor(0.80 * nrow(Boston))
  train_ind <- sample(seq_len(nrow(Boston)), size=sample_size, replace=FALSE)
  train <- Boston[train_ind,]
  test <- Boston[-train_ind,]
#Fit first logit model
    logit.fit1 <- glm(crim01~medv+lstat+indus+nox+age+dis+rad+tax+ptratio, data=train, family=binomial)
    logit.probs1 <- predict(logit.fit1, test, type="response")
    logit.class1 <- ifelse(logit.probs1>0.5,1,0)
    #Generate the confusion matrix
    table(logit.class1, test$crim01)
```

```
##
## logit.class1  0  1
##            0 48  9
##            1  3 42
```

```r
    #Compute the test error
    mean(logit.class1 != test$crim01)
```

```
## [1] 0.1176471
```

```r
#Fit second logit model
    logit.fit2 <- glm(crim01~rad+tax, data=train, family=binomial)
    logit.probs2 <- predict(logit.fit2, test, type="response")
    logit.class2 <- ifelse(logit.probs2>0.5,1,0)
    #Generate the confusion matrix
    table(logit.class2, test$crim01)
```

```
##
## logit.class2  0  1
##            0 47 14
##            1  4 37
```

```r
    #Compute the test error
    mean(logit.class2 != test$crim01)
```

```
## [1] 0.1764706
```

```r
#Fit first LDA model
    lda.fit1 <- lda(crim01~medv+lstat+indus+nox+age+dis+rad+tax+ptratio, data=train)
    lda.pred1 <- predict(lda.fit1, test)
    lda.class1 <- lda.pred1$class
    #Generate confusion matrix
    table(lda.class1, test$crim01)
```

```
##
## lda.class1  0  1
##          0 47 15
##          1  4 36
```

```r
    #Compute the test error
    mean(lda.class1 != test$crim01)
```

```
## [1] 0.1862745
```

```r
#Fit second LDA model
  lda.fit2 <- lda(crim01~rad+tax, data=train)
  lda.pred2 <- predict(lda.fit2, test)
  lda.class2 <- lda.pred2$class
  #Generate the confusion matrix
  table(lda.class2, test$crim01)
```

```
##
## lda.class2  0  1
##          0 49 21
##          1  2 30
```

```r
  #Compute the test error
  mean(lda.class2 != test$crim01)
```

```
## [1] 0.2254902
```

```r
#Fit first QDA model
    qda.fit1 <- qda(crim01~medv+lstat+indus+nox+age+dis+rad+tax+ptratio, data=train)
    qda.pred1 <- predict(qda.fit1, test)
```

```r
    qda.class1 <- qda.pred1$class
    #Generate confusion matrix
    table(qda.class1, test$crim01)
```

```
##
## qda.class1  0  1
##          0 51 11
##          1  0 40
```

```r
    #Compute the test error
    mean(qda.class1 != test$crim01)
```

```
## [1] 0.1078431
```

```r
#Fit second QDA model
    qda.fit2 <- qda(crim01~rad+tax, data=train)
    qda.pred2 <- predict(qda.fit2, test)
    qda.class2 <- qda.pred2$class
    #Generate confusion matrix
    table(qda.class2, test$crim01)
```

```
##
## qda.class2  0  1
##          0 51 21
##          1  0 30
```

```r
    #compute the test error
    mean(qda.class2 != test$crim01)
```

```
## [1] 0.2058824
```

```r
#Fit some KNN models
    library(class)
    train.X1 <- cbind(train$medv, train$lstat, train$indus, train$nox, train$age, train$dis, train$rad,
    test.X1 <- cbind(test$medv, test$lstat, test$indus, test$nox, test$age, test$dis, test$rad, test$ta
    train.X2 <- cbind(train$rad, train$tax)
    test.X2 <- cbind(test$rad, test$tax)

    knn.pred1.1 <- knn(train.X1, test.X1, train$crim01, k=1)
    table(knn.pred1.1, test$crim01)
```

```
##
## knn.pred1.1  0  1
##           0 49  7
##           1  2 44
```

```r
    mean(knn.pred1.1 != test$crim01)
```

```
## [1] 0.08823529
```

```r
    knn.pred5.1 <- knn(train.X1, test.X1, train$crim01, k=5)
    table(knn.pred5.1, test$crim01)
```

```
##
## knn.pred5.1  0  1
##           0 48  7
##           1  3 44
```

```r
mean(knn.pred5.1 != test$crim01)
```

```
## [1] 0.09803922
```

```r
knn.pred10.1 <- knn(train.X1, test.X1, train$crim01, k=10)
table(knn.pred10.1, test$crim01)
```

```
##
## knn.pred10.1  0  1
##            0 48  7
##            1  3 44
```

```r
mean(knn.pred10.1 != test$crim01)
```

```
## [1] 0.09803922
```

```r
knn.pred25.1 <- knn(train.X1, test.X1, train$crim01, k=25)
table(knn.pred25.1, test$crim01)
```

```
##
## knn.pred25.1  0  1
##            0 44  9
##            1  7 42
```

```r
mean(knn.pred25.1 != test$crim01)
```

```
## [1] 0.1568627
```

```r
knn.pred50.1 <- knn(train.X1, test.X1, train$crim01, k=50)
table(knn.pred50.1, test$crim01)
```

```
##
## knn.pred50.1  0  1
##            0 42  9
##            1  9 42
```

```r
mean(knn.pred50.1 != test$crim01)
```

```
## [1] 0.1764706
```

```r
knn.pred100.1 <- knn(train.X1, test.X1, train$crim01, k=100)
table(knn.pred100.1, test$crim01)
```

```
##
## knn.pred100.1  0  1
##             0 48 20
##             1  3 31
```

```r
mean(knn.pred100.1 != test$crim01)
```

```
## [1] 0.2254902
```

```r
knn.pred200.1 <- knn(train.X1, test.X1, train$crim01, k=200)
table(knn.pred200.1, test$crim01)
```

```
##
## knn.pred200.1  0  1
##             0 49 21
##             1  2 30
```

```r
mean(knn.pred200.1 != test$crim01)
```

```
## [1] 0.2254902
```

```r
knn.pred1.2 <- knn(train.X2, test.X2, train$crim01, k=1)
table(knn.pred1.2, test$crim01)
```

```
##
## knn.pred1.2  0  1
##           0 51  5
##           1  0 46
```

```r
mean(knn.pred1.2 != test$crim01)
```

```
## [1] 0.04901961
```

```r
knn.pred5.2 <- knn(train.X2, test.X2, train$crim01, k=5)
table(knn.pred5.2, test$crim01)
```

```
##
## knn.pred5.2  0  1
##           0 46  5
##           1  5 46
```

```r
mean(knn.pred5.2 != test$crim01)
```

```
## [1] 0.09803922
```

```r
knn.pred10.2 <- knn(train.X2, test.X2, train$crim01, k=10)
table(knn.pred10.2, test$crim01)
```

```
##
## knn.pred10.2  0  1
##            0 45  5
##            1  6 46
```

```r
mean(knn.pred10.2 != test$crim01)
```

```
## [1] 0.1078431
```

```r
knn.pred25.2 <- knn(train.X2, test.X2, train$crim01, k=25)
table(knn.pred25.2, test$crim01)
```

```
##
## knn.pred25.2  0  1
##            0 42  7
##            1  9 44
```

```r
mean(knn.pred25.2 != test$crim01)
```

```
## [1] 0.1568627
```

```r
knn.pred50.2 <- knn(train.X2, test.X2, train$crim01, k=50)
table(knn.pred50.2, test$crim01)
```

```
##
## knn.pred50.2  0  1
##            0 36  7
##            1 15 44
```

```r
mean(knn.pred50.2 != test$crim01)
```

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

```r
knn.pred100.2 <- knn(train.X2, test.X2, train$crim01, k=100)
table(knn.pred100.2, test$crim01)
```

```
##
## knn.pred100.2  0  1
##             0 49 21
##             1  2 30
```

```r
mean(knn.pred100.2 != test$crim01)
```

```
## [1] 0.2254902
```

```r
knn.pred200.2 <- knn(train.X2, test.X2, train$crim01, k=200)
table(knn.pred200.2, test$crim01)
```

```
##
## knn.pred200.2  0  1
##             0 49 21
##             1  2 30
```

```r
mean(knn.pred200.2 != test$crim01)
```

```
## [1] 0.2254902
```

The model with the lowest test error is the knn model with two predictors and k=1 at 4.90% followed by the knn model with many predictors and k=1 at 8.82%. The logit model with many predictors and the QDA model with many predictors also with did pretty well with test errors of 11.76% and 10.78%, respectively.