ECON 573 PS3

Part 1

Ex 2, 3, 10 from Chapter 6 of ISL.

2) For parts (a) through (c), indicate which of i. through iv. is correct. Justify your answer.

i) More flexible and hence will give improved prediction accuracy when its increase in bias is less than its decrease in variance.
ii) More flexible and hence will give improved prediction accuracy when its increase in variance is less than its decrease in bias.
iii) Less flexible and hence will give improved prediction accuracy when its increase in bias is less than its decrease in variance.
iv) Less flexible and hence will give improved prediction accuracy when its increase in variance is less than its decrease in bias.

2a) The lasso, relative to least squares, is: iii is the right answer, as decreasing the coefficient will yield more tight results. This will essentially remove any variables that are not relevant, which reduces variance, and leads to an increase in bias

2b) The ridge regression, relative to least squares, is: iii is the right answer, compared to lasso it will have a lower bias but will still be high. The ridge regression also decreases the coefficients again yielding lower variance

2c) Non-linear methods, relative to least squares: ii is the right answer, due to higher flexibility there is less bias

3) Suppose we estimate the regression coefficients in a linear regression model by minimizing

$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2 \quad \text{subject to} \quad \sum_{j=1}^{p}|\beta_j|\le s$$

for a particular value of s. For parts (a) through (e), indicate which of i. through v. is correct. Justify your answer.

i) Increase initially, and then eventually start decreasing in an inverted U shape.
ii) Decrease initially, and then eventually start increasing in a U shape.
iii) Steadily increase.
iv) Steadily decrease.
v) Remain constant.

3a) As we increase lambda from 0, the training RSS will: iv 3b) As we increase lambda from 0, the test RSS will: ii 3c) As we increase lambda from 0, the variance will: iii 3d) As we increase lambda from 0, the bias will: iv 3e) As we increase lambda from 0, the irreducible error will: v

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
11.

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

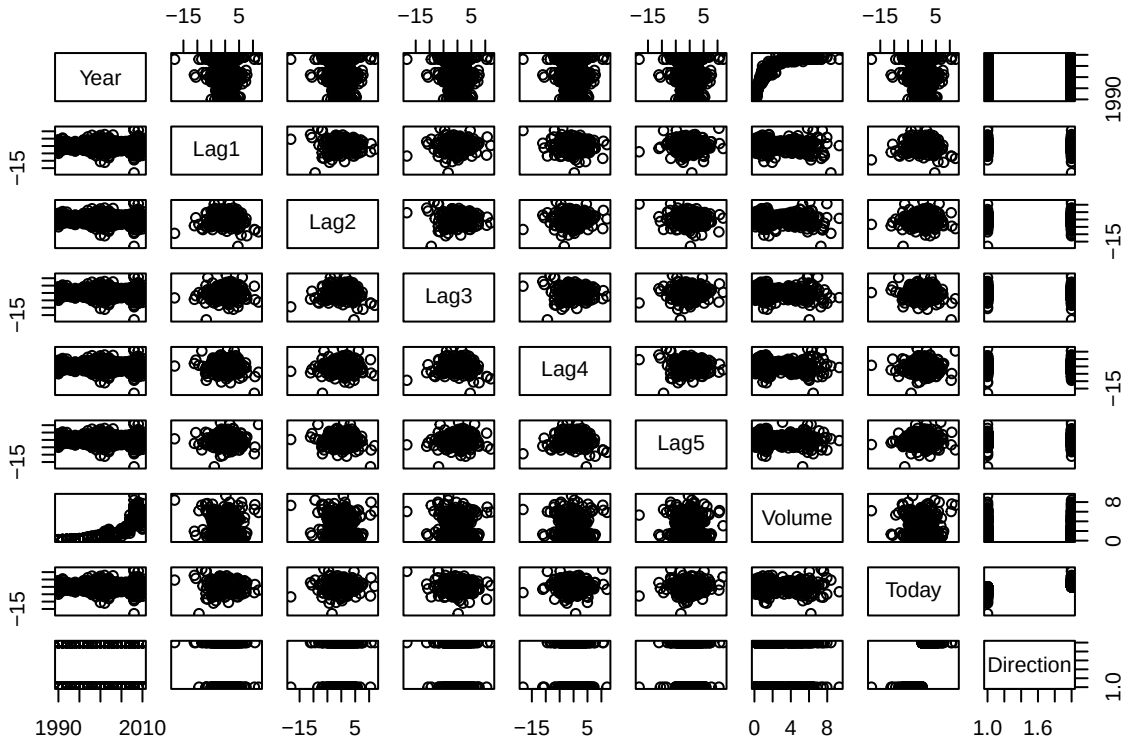
```
require(ISLR)
```

```
## Loading required package: ISLR
```

```
attach(Weekly)
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  
##            
##            
##            
##
```

```
pairs(Weekly)
```



Based on the data and the plot its seems as thought the number of shares traded over time has gone up alot

10b) 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?

```
weeklyLog = glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume, "binomial", Weekly)
summary(weeklyLog)
```

```
##
## 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
```

Of the 6 variables Lag2 is significant at the 0.05 level

10c) 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.

```
pVals = predict(weeklyLog, type= "response")
matrix = rep("Down", length(pVals))
matrix[pVals > 0.5] = "Up"
table(matrix, Direction)
```

```
##       Direction
## matrix Down  Up
##   Down   54  48
##   Up    430 557
```

This model seems to have some type of heavy bias towards being up. While we have 556 true ups, we also have 430 false positives

10d) 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).

```
training = (Year < 2009)
test = Weekly[!training,]
weeklyLog = glm(Direction ~ Lag2, "binomial", data = Weekly, subset = training)
pVals = predict(weeklyLog, test, type= "response")
matrix = rep("Down", length(pVals))
matrix[pVals > 0.5] = "Up"
testDirection = Direction[!training]
table(matrix, testDirection)
```

```
##       testDirection
## matrix Down Up
##   Down    9  5
##   Up     34 56
```

```
mean(matrix == testDirection)
```

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

This model predicted weekly trends 62.5% of the time

10e) Repeat (d) using LDA.

```
library(MASS)
weeklyLDA = lda(Direction~Lag2, data = Weekly, family="binomial", subset = training)
pVals = predict(weeklyLDA, test)$class
table(pVals, testDirection)
```

```
##       testDirection
## pVals  Down Up
##   Down    9  5
##   Up     34 56
```

```
mean(pVals == testDirection)
```

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

10f) Repeat (d) using QDA

```
weeklyQDA = qda(Direction ~ Lag2, data = Weekly, subset = training)
pVals = predict(weeklyQDA, test)$class
table(pVals, testDirection)
```

```
##       testDirection
## pVals  Down Up
##   Down    0  0
##   Up     43 61
```

```
mean(pVals == testDirection)
```

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

10h) Which of these methods appears to provide the best results on this data? logistic regression and LDA seem to be most effective at 62.5%

10i) 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.

Part 2

Ex. 4, 8, 11, 12, 13 from Chapter 4 of ISL.

4) When the number of features p is large, there tends to be a deterioration in the performance of KNN and other local approaches that perform prediction using only observations that are near the test observation for which a prediction must be made. This phenomenon is known as the curse of dimensionality, and it ties into the fact that curse of dinon- parametric approaches often perform poorly when p is large. We mensionality will now investigate this curse.

4a) Suppose that we have a set of observations, each with measurements on p = 1 feature, X. We assume that X is uniformly (evenly) distributed on [0, 1]. Associated with each observation is a response value. Suppose that we wish to predict a test observation's response using only observations that are within 10% of the range of X closest to that test observation. For instance, in order to predict the response for a test observation with X = 0.6, 4.7 Exercises 169 we will use observations in the range [0.55, 0.65]. On average, what fraction of the available observations will we use to make the prediction?

There are three cases that we need to account for: if x is between 0.5 and 0.95 observation are in between that 10% window $\int_{0.5}^{0.95}$ 10 dx If x is below 0.5 then the observations are between 0 and x + 0.05 $\int_{0}^{0.05}$ (100x + 5) dx If x is greater than 0.5 then the observation are between 0.95 and 1 $\int_{0.95}^{1}$ (-100x + 105) dx Which becomes: $\int_{0.5}^{0.95}$ 10 dx + $\int_{0}^{0.05}$ (100x + 5) dx + $\int_{0.95}^{1}$ (-100x + 105) dx = 9.75%

4b) Now suppose that we have a set of observations, each with measurements on p = 2 features, X1 and X2. We assume that (X1,X2) are uniformly distributed on [0, 1] × [0, 1]. We wish to predict a test observation's response using only observations that are within 10% of the range of X1 and within 10% of the range of X2 closest to that test observation. For instance, in order to predict the response for a test observation with X1 = 0.6 and X2 = 0.35, we will use observations in the range [0.55, 0.65] for X1 and in the range [0.3, 0.4] for X2. On average, what fraction of the available observations will we use to make the prediction?

If both X1 and X2 are mutually exclusive then we will use then it will be similar to the calculation provided in part a.

9.75 * 9.75 = 9.506%

4c) Now suppose that we have a set of observations on p = 100 features. Again the observations are uniformly distributed on each feature, and again each feature ranges in value from 0 to 1. We wish to predict a test observation's response using observations within the 10% of each feature's range that is closest to that test observation. What fraction of the available observations will we use to make the prediction?

In the last problem we saw that for 2 features it was 9.75ˆ2, similarly, for a 100 features the answer will be 9.75ˆ100

4d) Using your answers to parts (a)–(c), argue that a drawback of KNN when p is large is that there are very few training observations "near" any given test observation.

As the number of features get larger and larger we see that the resulting fraction becomes smaller and smaller. As the features get larger and larger this number will go towards 0

4e) Now suppose that we wish to make a prediction for a test observation by creating a p-dimensional hypercube centered around the test observation that contains, on average, 10% of the training observations. For p = 1, 2, and 100, what is the length of each side of the hypercube? Comment on your answer.

The length of the hypercube is given by L(p) = (1/10)ˆ(1/p) Thus: L(1): (1/10)ˆ(1/1) L(2): (1/10)ˆ(1/2) L(100): (1/10)ˆ(1/100)

8. Suppose that we take a data set, divide it into equally-sized training and test sets, and then try out two different classification procedures. First we use logistic regression and get an error rate of 20% on the training data and 30% on the test data. Next we use 1-nearest neighbors (i.e. K = 1) and get an average error rate (averaged over both test and training data sets) of 18%. Based on these results, which method should we prefer to use for classification of new observations? Why?

For logistic regression: - error rate 20% - average error rate 25% - test error 30%

For knn - training error of 0% as P(Y=j|X=xi)=I(Yi=j) - average error 18% - test error 36%

We would prefer to use the logistic regression due to lower test error

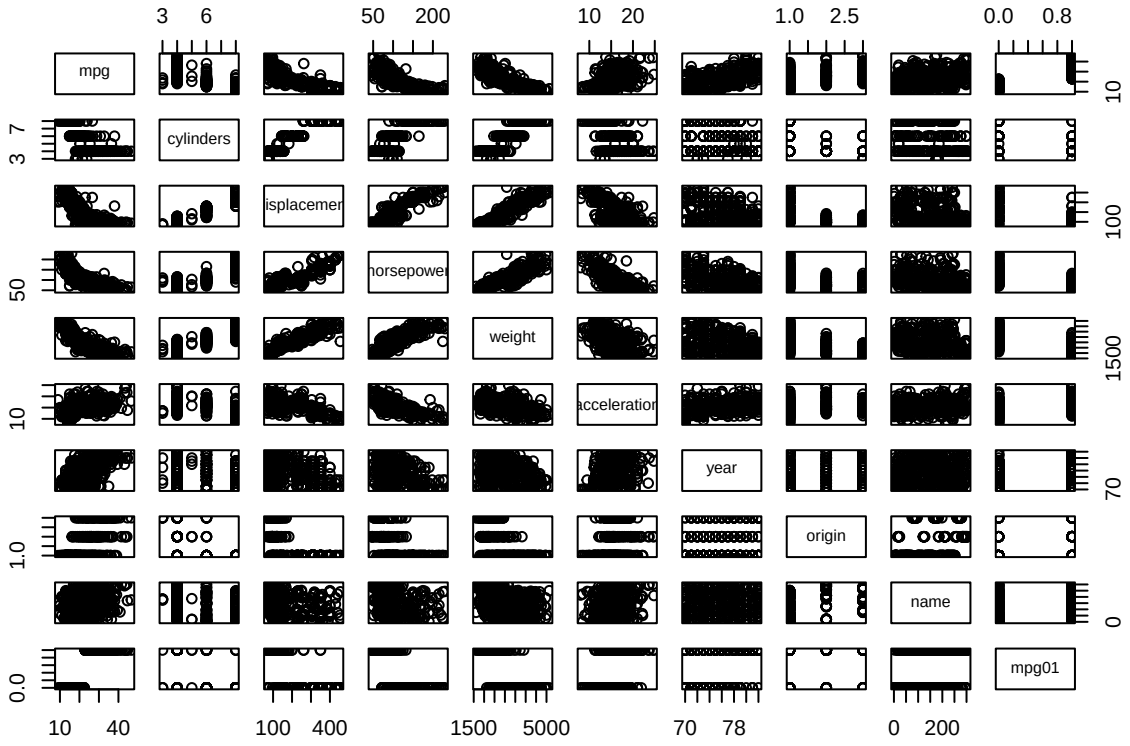
11) In this problem, you will develop a model to predict whether a given car gets high or low gas mileage based on the Auto data set.

11a) Create a binary variable, mpg01, that contains a 1 if mpg contains a value above its median, and a 0 if mpg contains a value below its median. You can compute the median using the median() function. Note you may find it helpful to use the data.frame() function to create a single data set containing both mpg01 and the other Auto variables.

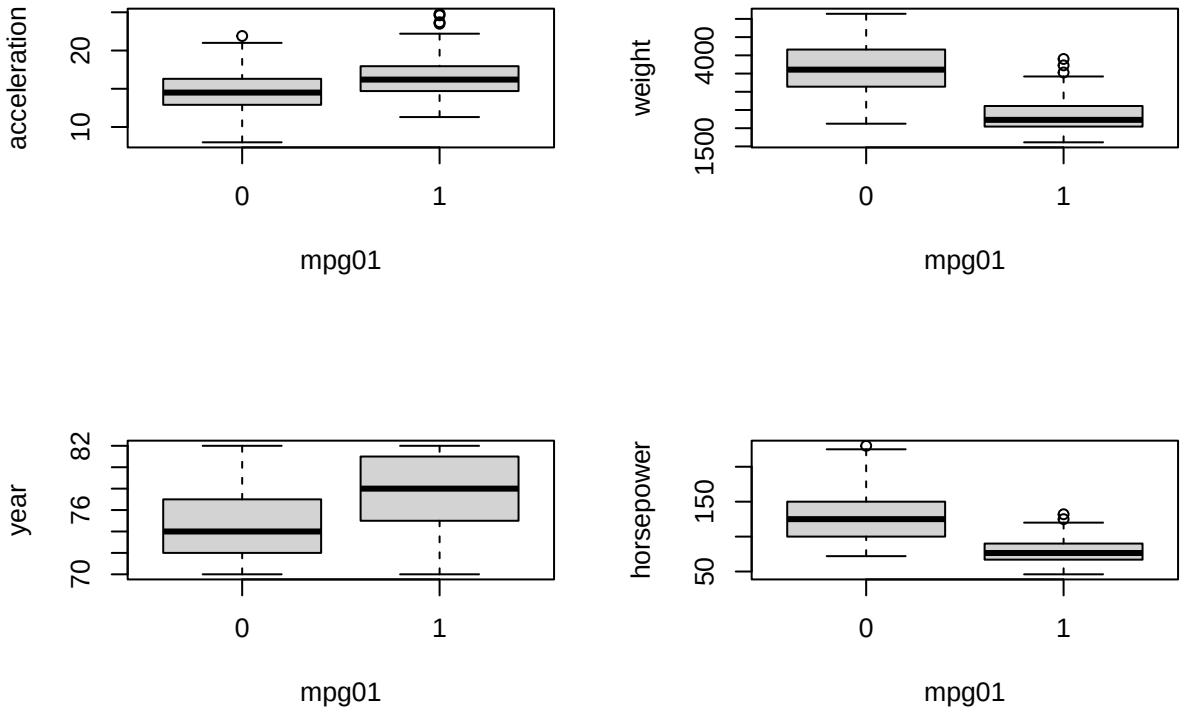
```
attach(Auto)
#detach(Weekly)
mpg01 = rep(0, length(Auto$mpg))
mpg01[Auto$mpg > median(Auto$mpg)] = 1
Auto = data.frame(Auto, mpg01)
```

11b) Explore the data graphically in order to investigate the association between mpg01 and the other features. Which of the other features seem most likely to be useful in predicting mpg01? Scatterplots and boxplots may be useful tools to answer this question. Describe your findings.

```
pairs(Auto)
```




```
par(mfrow=c(2,2))
boxplot(acceleration ~ mpg01, data = Auto )
boxplot(weight ~ mpg01, data = Auto )
boxplot(year ~ mpg01, data = Auto )
boxplot(horsepower ~ mpg01, data = Auto )
```



11c) Split the data into a training set and a test set.

```
set.seed(1)
trainData = sample(nrow(Auto), size = nrow(Auto) * 0.6)
training = Auto[trainData,]
testing = Auto[!trainData,]
```

11d) Perform LDA on the training data in order to predict mpg01 using the variables that seemed most associated with mpg01 in (b). What is the test error of the model obtained?

```
autoLDA = lda(mpg01 ~ weight + displacement + horsepower + year + cylinders, data = training)
prediction = predict(autoLDA, data = testing)
error = 1 - mean(prediction$class == testing$mpg01)
error
```

```
## [1] NaN
```

prediction error is 0.494

11e) Perform QDA on the training data in order to predict mpg01 using the variables that seemed most associated with mpg01 in (b). What is the test error of the model obtained?

```
autoQDA = qda(mpg01 ~ weight + displacement + horsepower + year + cylinders, data = training)
prediction = predict(autoQDA, data = testing)
mean(prediction$class == testing$mpg01)
```

```
## [1] NaN
```

prediction error is 0.477

11f) Perform logistic regression on the training data in order to predict mpg01 using the variables that seemed most associated with mpg01 in (b). What is the test error of the model obtained?

```
autoGLM = glm(mpg01 ~ weight + displacement + horsepower + year + cylinders, data = training, family = b
#prediction = predict(autoGLM, testing, type="response")
#error = 1 - mean(prediction == testing$mpg01)
#error
```

11g) Perform KNN on the training data, with several values of K, in order to predict mpg01. Use only the variables that seemed most associated with mpg01 in (b). What test errors do you obtain? Which value of K seems to perform the best on this data set?

12) This problem involves writing functions. 12a) Write a function, Power(), that prints out the result of raising 2 to the 3rd power. In other words, your function should compute 23 and print out the results.

```
Power =  function(a = 2, b = 3) {print(a^b)};
Power()
```

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

12b) 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) {print(x^a)}
```

12c) Using the Power2() function that you just wrote, compute 103, 817, and 1313.

```
Power2(10, 3)
```

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

```
Power2(81, 7)
```

```
## [1] 2.287679e+13
```

```
Power2(131, 3)
```

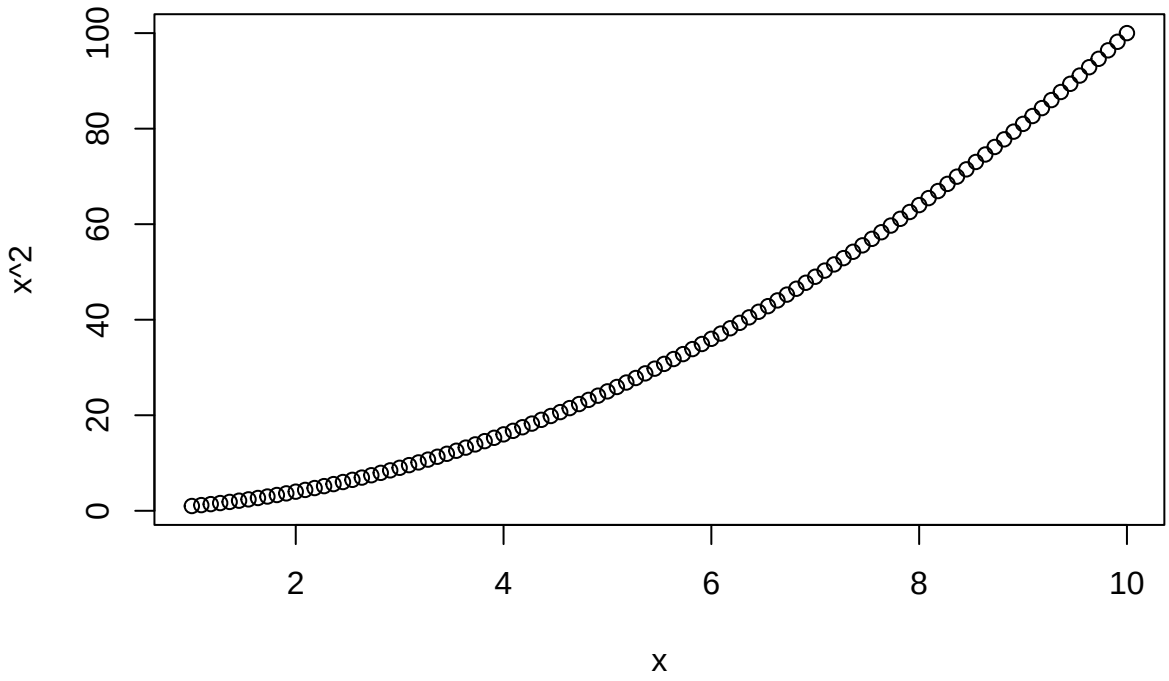
```
## [1] 2248091
```

12d) 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 return() result, using the following line:

```
Power3 = function(x, a) {return(x^a)}
```

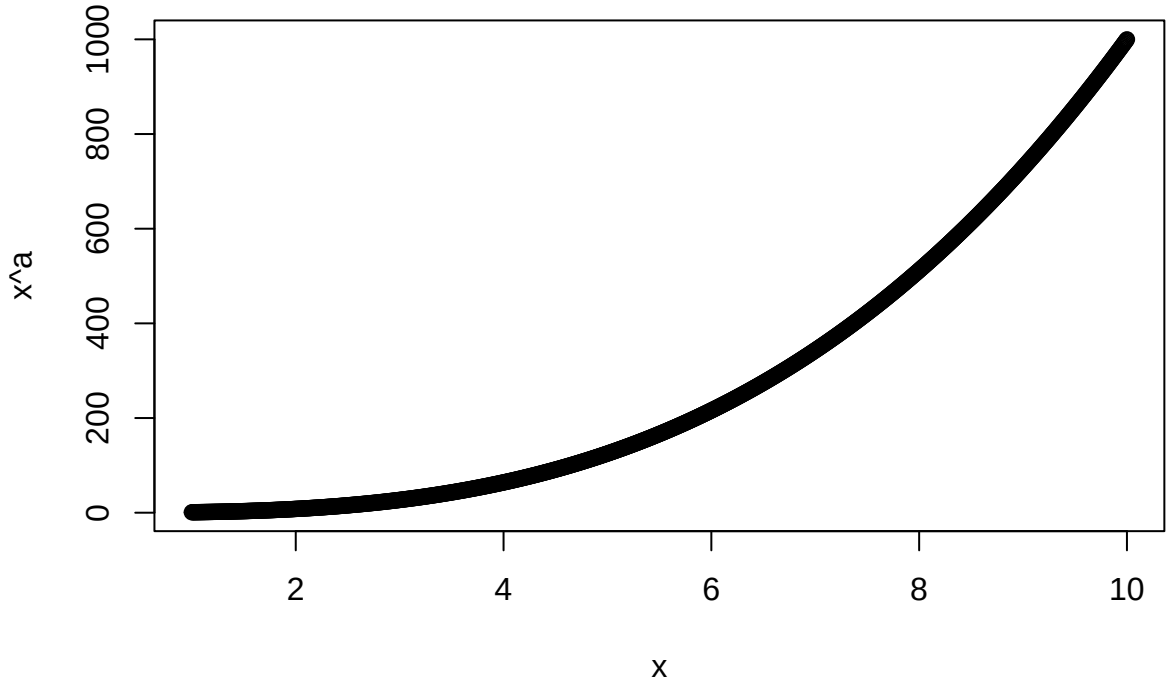
12e) Now using the Power3() function, create a plot of f(x) = x2. The x-axis should display a range of integers from 1 to 10, and the y-axis should display x2. Label the axes appropriately, and use an appropriate title for the figure. Consider displaying either the x-axis, the y-axis, or both on the log-scale. You can do this by using log="x", log="y", or log="xy" as arguments to the plot() function.

```
plot(seq(from = 1, to = 10, length.out = 100), Power3(seq(from = 1, to = 10, length.out = 100), 2), xla
```



12f) Create a function, PlotPower(), that allows you to create a plot of x against xˆa for a fixed a and for a range of values of x. For instance, if you call

```
PlotPower = function(x, a) {
  return(plot(seq(from = min(x), to = max(x), length.out = 1000), seq(from = min(x), to = max(x), lengt
}
PlotPower(1:10, 3)
```



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.

```
library(MASS)
attach(Boston)
crimeAboveMed = rep(0, length(crim))
crimeAboveMed[crim > median(crim)] = 1
Bostonlog = data.frame(Boston, crimeAboveMed)

#setup
trainData = 1:(length(crim) / 2)
testData = (length(crim) / 2 + 1):length(crim)
training = Bostonlog[trainData, ]
testing = Bostonlog[testData, ]
testCrime = crimeAboveMed[testData]
```

Logistic

```
bostonLOG = glm(crimeAboveMed ~ . - crimeAboveMed - crim, data = Bostonlog, family = binomial, subset =
```

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

```
summary(bostonLOG)
```

```
##
```

```
## Call:
## glm(formula = crimeAboveMed ~ . - crimeAboveMed - crim, family = binomial,
##     data = Bostonlog, subset = trainData)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.83229  -0.06593   0.00000   0.06181   2.61513
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -91.319906  19.490273  -4.685 2.79e-06 ***
## zn           -0.815573   0.193373  -4.218 2.47e-05 ***
## indus         0.354172   0.173862   2.037  0.04164 *
## chas          0.167396   0.991922   0.169  0.86599
## nox          93.706326  21.202008   4.420 9.88e-06 ***
## rm           -4.719108   1.788765  -2.638  0.00833 **
## age           0.048634   0.024199   2.010  0.04446 *
## dis           4.301493   0.979996   4.389 1.14e-05 ***
## rad           3.039983   0.719592   4.225 2.39e-05 ***
## tax          -0.006546   0.007855  -0.833  0.40461
## ptratio       1.430877   0.359572   3.979 6.91e-05 ***
## black        -0.017552   0.006734  -2.606  0.00915 **
## lstat         0.190439   0.086722   2.196  0.02809 *
## medv          0.598533   0.185514   3.226  0.00125 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 329.367  on 252  degrees of freedom
## Residual deviance:  69.568  on 239  degrees of freedom
## AIC: 97.568
##
## Number of Fisher Scoring iterations: 10
```

```
pvals = predict(bostonLOG, testing, type = "response")
prediction = rep(0, length(pvals))
prediction[pvals > 0.5] = 1
table(prediction, testCrime)
```

```
##           testCrime
## prediction   0   1
##          0  68  24
##          1  22 139
```

```
1 - mean(prediction == testCrime)
```

```
## [1] 0.1818182
```

Test error: 18.1%

LDA

```
#setup
attach(Boston)
```

```
## The following objects are masked from Boston (pos = 3):
##
##     age, black, chas, crim, dis, indus, lstat, medv, nox, ptratio, rad,
##     rm, tax, zn
```

```
crimeAboveMed = rep(0, length(crim))
crimeAboveMed[crim > median(crim)] = 1
Bostlda = data.frame(Boston, crimeAboveMed)
trainData = 1:(length(crim) / 2)
testData = (length(crim) / 2 + 1):length(crim)
training = Bostlda[trainData, ]
testing = Bostlda[testData, ]
testCrime = crimeAboveMed[testData]

bostonLDA = lda(crimeAboveMed ~ . - crimeAboveMed - crim - chas - nox - tax, data = Bostlda, subset = t
prediction = predict(bostonLDA, testing)
table(prediction$class, testCrime)
```

```
##    testCrime
##       0   1
##   0  83  28
##   1   7 135
```

```
1 - mean(prediction$class == testCrime)
```

```
## [1] 0.1383399
```

Test error: 13.8%

KNN

```
#setup
attach(Boston)
```

```
## The following objects are masked from Boston (pos = 3):
##
##     age, black, chas, crim, dis, indus, lstat, medv, nox, ptratio, rad,
##     rm, tax, zn

## The following objects are masked from Boston (pos = 4):
##
##     age, black, chas, crim, dis, indus, lstat, medv, nox, ptratio, rad,
##     rm, tax, zn
```

```
crimeAboveMed = rep(0, length(crim))
crimeAboveMed[crim > median(crim)] = 1
Bostonknn = data.frame(Boston, crimeAboveMed)
trainData = 1:(length(crim) / 2)
```

```
testData = (length(crim) / 2 + 1):length(crim)
training = Bostonknn[trainData, ]
testing = Bostonknn[testData, ]
testCrime = crimeAboveMed[testData]

library(class)
knnTrain = cbind(zn, indus, chas, nox, rm, age, dis, rad, tax, ptratio, black, lstat, medv)[trainData,]
knnTest = cbind(zn, indus, chas, nox, rm, age, dis, rad, tax, ptratio, black, lstat, medv)[testData,]
crimeAboveMedTrain = crimeAboveMed[trainData]
prediction = knn(knnTrain, knnTest, crimeAboveMedTrain, k = 1)
table(prediction, testCrime)
```

```
##           testCrime
## prediction   0   1
##          0  85 111
##          1   5  52
```