Big Data Analytics with R - Solution to Coursework 2

1. Logistic regression (Textbook 4.10)

1.a. First we load and summarize the data using the main measures of centrality and spread and graphically by plotting scatterplots of all variables.

```
require(ISLR)
```

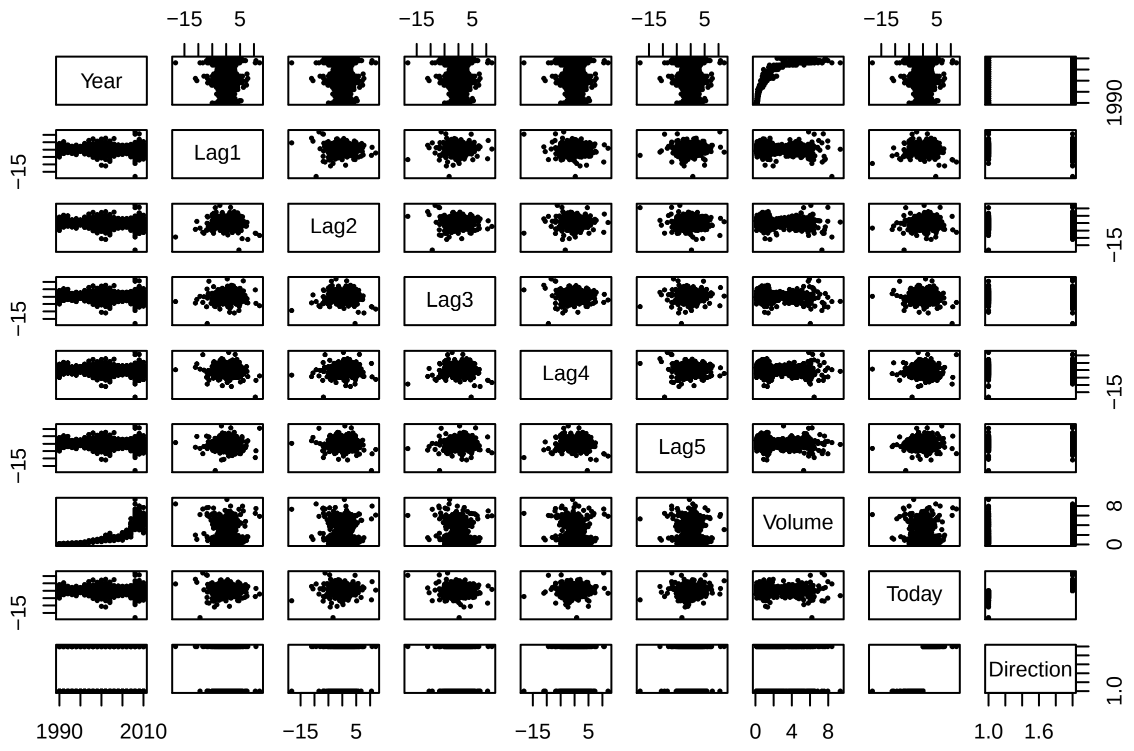
```
## Loading required package: ISLR

## Warning: package 'ISLR' was built under R version 3.1.1
```

```
summary(Weekly)
```

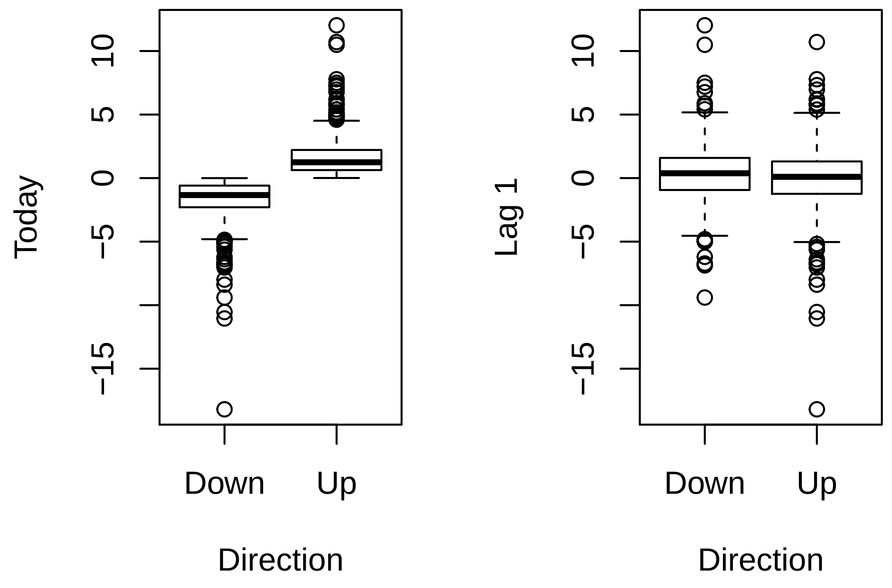
```
##       Year           Lag1               Lag2               Lag3
##  Min.   :1990   Min.   :-18.1950   Min.   :-18.1950   Min.   :-18.1950
##  1st Qu.:1995   1st Qu.: -1.1540   1st Qu.: -1.1540   1st Qu.: -1.1580
##  Median :2000   Median :  0.2410   Median :  0.2410   Median :  0.2410
##  Mean   :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
```

```
plot(Weekly, pch=20, cex=0.5)
```



From the matrix of scatterplots we see that there do not appear to be any relationships between `Today` and any of the `Lag` variables. There is a relationship between `Year` and `Volume`, with the latter increasing over time, showing that the market is growing.

```
par(mfrow=c(1,2))
plot(Weekly$Direction, Weekly$Today, xlab="Direction", ylab="Today")
plot(Weekly$Direction, Weekly$Lag1, xlab="Direction", ylab="Lag 1")
```



```
par(mfrow=c(1,1))
```

There is the expected relationship between `Direction` and `Today` but there does not appear to be a relationship between `Direction` and any of the `Lag` variables (only the boxplot for `Lag1` is shown).

1.b. We use the `glm` function to perform logistic regression of `Direction` on the five `Lag` variables and `Volume`.

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

From the regression output we see that only the coefficient on `Lag2` is statistically different from zero.

1.c. Next we first predict the probability of the market moving up based on the logistic regression model, using the training data.

```
proba <- predict(fit, type="response")
predict <- rep("Down", length(proba))
predict[proba > 0.5] <- "Up"
actual <- Weekly$Direction
table(predict, actual)
```

```
##        actual
## predict Down  Up
##    Down   54  48
##    Up    430 557
```

```
mean(predict == actual)
```

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

Note that we assume the predicted direction is `Up` if the predicted probability is greater than 0.5.

From the confusion matrix we see that the model has a prediction accuracy of 0.5610652 on the training data. The main error that the model is making is that it appears to be misclassifying `Down` days, i.e. false positives.

1.d. We start by creating an index for the training set of data, up to and including 2008, with the remaining data from 2009 onwards to be used for the test set.

```
train <- (Weekly$Year <= 2008)
dim(Weekly[train, ])
```

```
## [1] 985   9
```

```
dim(Weekly[!train, ])
```

```
## [1] 104   9
```

We now fit the model as before

```r
fit <- glm(Direction ~ Lag2, data=Weekly, subset=train, family="binomial")
summary(fit)
```

```
##
## Call:
## glm(formula = Direction ~ Lag2, family = "binomial", data = Weekly,
##     subset = train)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.536  -1.264   1.021   1.091   1.368
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.20326    0.06428   3.162  0.00157 **
## Lag2         0.05810    0.02870   2.024  0.04298 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1354.7  on 984  degrees of freedom
## Residual deviance: 1350.5  on 983  degrees of freedom
## AIC: 1354.5
##
## Number of Fisher Scoring iterations: 4
```

From the output we see that `Lag2` is significant. We now make predictions for the test data using this model, create the confusion matrix and determine the test accuracy.

```r
proba <- predict(fit, newdata=Weekly[!train, ], type="response")
predict <- rep("Down", nrow(Weekly[!train, ]))
predict[proba > 0.5] <- "Up"
actual <- Weekly[!train, ]$Direction
table(predict, actual)
```

```
##        actual
## predict Down Up
##    Down    9  5
##    Up     34 56
```

```r
mean(predict == actual)
```

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

Using just `Lag2` as the predictor, our logistic regression model trained on data up to 2008 has a prediction accurary of 0.625 on the test data from 2009 onwards.

```r
# clean up before moving on to next question
rm(list = ls())
```

2. Logistic regression (Textbook 4.11)

2.a. First load the `ISLR` package to get the `Auto` dataset

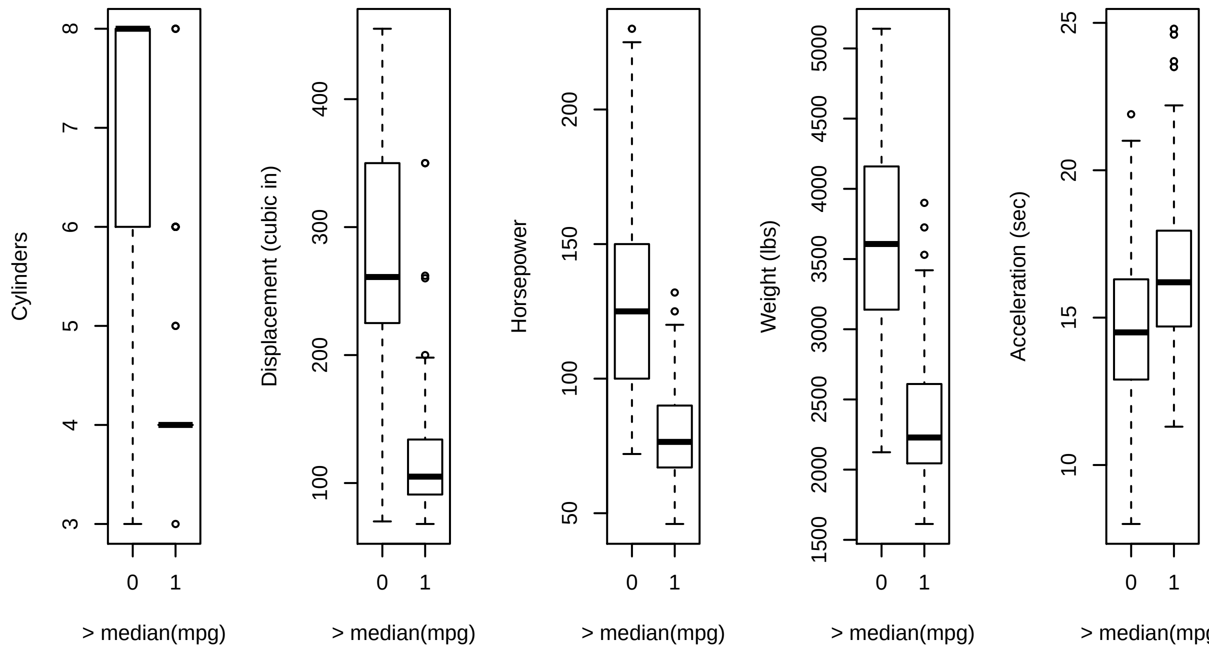
```
require(ISLR)
```

Create a new indicator variable `mpg01` indicating whether a specific car's fuel economy is lower or higher than the median vehicle in the dataset, 22.75 miles per gallon.

```
mpg01 <- rep(0, nrow(Auto))
mpg01[Auto$mpg > median(Auto$mpg)] <- 1
Auto <- data.frame(Auto, mpg01)
```

2.b. We first plot boxplots for each of the numeric variables against `mpg01`.

```
par(mfrow=c(1,5))
boxplot(Auto$cylinders ~ Auto$mpg01, xlab="> median(mpg)", ylab="Cylinders")
boxplot(Auto$displacement ~ Auto$mpg01, xlab="> median(mpg)", ylab="Displacement (cubic in)")
boxplot(Auto$horsepower ~ Auto$mpg01, xlab="> median(mpg)", ylab="Horsepower")
boxplot(Auto$weight ~ Auto$mpg01, xlab="> median(mpg)", ylab="Weight (lbs)")
boxplot(Auto$acceleration ~ Auto$mpg01, xlab="> median(mpg)", ylab="Acceleration (sec)")
```



```
par(mfrow=c(1,1))
```

From these boxplots it appears that there exist relationships between `mpg01` and `cylinders`, `displacement`, `horsepower` and `weight`. There is only a weak relationship with `acceleration`, which will be excluded from the logistic regression model.

Next we show a table of `mpg01` to show the relationship with the categorical variable `origin`, which has value 1 for American cars, 2 for European cars and 3 for Japanese cars.

```r
table(Auto$mpg01, Auto$origin)
```

```
##
##       1   2   3
##   0 173  14   9
##   1  72  54  70
```

There appears to be a relationship here, with relatively more American cars having `mpg01 = 0`, so we will include `origin` in our model.

2.c. We first create a random sample of indices to create our training set.

```r
set.seed(1)
train <- sample(nrow(Auto), nrow(Auto)/2)
length(train)
```

```
## [1] 196
```

Next we train a logistic regression model using the variables identified in (b).

```r
fit <- glm(mpg01 ~ cylinders + displacement + horsepower + weight + as.factor(origin), data=Auto, subse
summary(fit)
```

```
##
## Call:
## glm(formula = mpg01 ~ cylinders + displacement + horsepower +
##     weight + as.factor(origin), family = "binomial", data = Auto,
##     subset = train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.28128  -0.20527  -0.00631   0.43109   2.97174
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)        12.839617   2.470426   5.197 2.02e-07 ***
## cylinders          -0.311671   0.484135  -0.644   0.5197
## displacement        0.004062   0.013164   0.309   0.7576
## horsepower         -0.064435   0.020030  -3.217   0.0013 **
## weight             -0.002081   0.001061  -1.961   0.0499 *
## as.factor(origin)2  0.911354   0.823521   1.107   0.2684
## as.factor(origin)3  0.416375   0.822655   0.506   0.6128
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 271.20  on 195  degrees of freedom
## Residual deviance: 112.82  on 189  degrees of freedom
## AIC: 126.82
##
## Number of Fisher Scoring iterations: 7
```

From this we see that `horsepower` and `weight` are significant at $\alpha = 0.05$, despite the strong separation found in (b) for some of the other variables. This may be explained by correlation between the various predictors.

Using the trained model, we can now make predictions for the test set.

```
probs <- predict(fit, newdata=Auto[-train, ], type="response")
preds <- rep(0, length(probs))
preds[probs > 0.5] <- 1
actual <- Auto[-train,]$mpg01
mean(preds == actual)
```

```
## [1] 0.8877551
```

Our model has a prediction accuary of 0.8877551 and so has test error of 0.1122449

```
# clean up before moving on to next question
rm(list = ls())
```

3. Validation set approach (Textbook 5.5)

3.a. First load the `ISLR` package.

```
require(ISLR)
```

Next we fit a logistic regression model of `default` against `income` and `balance`.

```
fit <- glm(default ~ income + balance, data=Default, family="binomial")
summary(fit)
```

```
##
## Call:
## glm(formula = default ~ income + balance, family = "binomial",
##     data = Default)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4725  -0.1444  -0.0574  -0.0211   3.7245
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.154e+01  4.348e-01 -26.545  < 2e-16 ***
## income       2.081e-05  4.985e-06   4.174 2.99e-05 ***
## balance      5.647e-03  2.274e-04  24.836  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
```

```
##
##     Null deviance: 2920.6  on 9999  degrees of freedom
## Residual deviance: 1579.0  on 9997  degrees of freedom
## AIC: 1585
##
## Number of Fisher Scoring iterations: 8
```

The results show that both `income` and `balance` are significant.

3.b.i. To estimate the test error, we will train the model on half of the dataset and estimate the test error on the remaining half. First we create a training set of indices

```
set.seed(1)
train <- sample(nrow(Default), nrow(Default)/2)
```

3.b.ii. Next we fit a logistic regression model using the same variables as in (a).

```
fit <- glm(default ~ income + balance, data=Default, subset=train, family="binomial")
```

3.b.iii. Predict `default` for the validation set

```
probs <- predict(fit, newdata=Default[-train, ], type="response")
preds <- rep("No", length(probs))
preds[probs > 0.5] <- "Yes"
actual <- Default[-train, ]$default
```

3.b.iv. Next we generatet the confusion matrix for the predicted versus actual outcome.

```
table(preds, actual)
```

```
##      actual
## preds   No  Yes
##   No  4805  115
##   Yes   28   52
```

The prediction accurary is given by

```
mean(preds == actual)
```

```
## [1] 0.9714
```

and the test error is given by

```
1 - mean(preds == actual)
```

```
## [1] 0.0286
```

3.c. To estimate the test error, we will repeat part (b) three times. To generate new estimates of the test error we use different seeds for creating the test sets. Since this is the only variable we create a function that generates a test error for a given seed value.

```
get_test_error <- function(seed) {
  set.seed(seed)
  train <- sample(nrow(Default), nrow(Default)/2)
  fit <- glm(default ~ income + balance, data=Default, subset=train, family="binomial")
  probs <- predict(fit, newdata=Default[-train, ], type="response")
  preds <- rep("No", length(probs))
  preds[probs > 0.5] <- "Yes"
  actual <- Default[-train, ]$default
  return(1 - mean(preds == actual))
}
```

We can test the function by repeating part (b) with seed = 1.

```
get_test_error(1)
```

```
## [1] 0.0286
```

which gives an identical result to part (b).

To generate three new test error estimates, we can now apply our function to a vector of seed values

```
seeds <- 2:4
errs <- sapply(seeds, get_test_error)
errs
```

```
## [1] 0.0276 0.0248 0.0262
```

The test error estimates range between 0.0248 and 0.0276, with an average test error of 0.0262.

3.d. We now alter our model to include `student` and then repeat the procedure from part b.

Generate a sample

```
set.seed(1)
train <- sample(nrow(Default), nrow(Default)/2)
```

Perform logistic regression on the model including `student`, using the training set.

```
fit <- glm(default ~ income + balance + student, data=Default, subset=train, family="binomial")
summary(fit)
```

```
##
## Call:
## glm(formula = default ~ income + balance + student, family = "binomial",
##     data = Default, subset = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2905  -0.1260  -0.0465  -0.0161   3.7715
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.147e+01  7.562e-01 -15.164   <2e-16 ***
## income       2.433e-06  1.256e-05   0.194    0.846
## balance      6.124e-03  3.525e-04  17.373   <2e-16 ***
## studentYes  -5.608e-01  3.473e-01  -1.615    0.106
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1456.95  on 4999  degrees of freedom
## Residual deviance:  731.81  on 4996  degrees of freedom
## AIC: 739.81
##
## Number of Fisher Scoring iterations: 8
```

We see that the dummy variable for `student` is not statistically significant. However, we also see that the `income` is no longer significant either.

Next we make predictions for the test set and calculate the test error

```r
probs <- predict(fit, newdata=Default[-train, ], type="response")
preds <- rep("No", length(probs))
preds[probs > 0.5] <- "Yes"
actual <- Default[-train, ]$default
mean(preds == actual)
```

```
## [1] 0.9712
```

```r
1 - mean(preds == actual)
```

```
## [1] 0.0288
```

Including `student` leads to a test error estimate of 0.0288, which is slightly higher than the average test error of 0.0262 obtained in part (c).

Therefore including `student` in the model does not substantially reduce the test error.

```r
# clean up before moving on to next question
rm(list = ls())
```

4. LOOCV and Loop (Textbook 5.7)

4.a. First load the `ISLR` package.

```r
require(ISLR)
```

Now we fit a logistic regression model of `Direction` on `Lag1` and `Lag2`.

```r
fit <- glm(Direction ~ Lag1 + Lag2, data=Weekly, family="binomial")
summary(fit)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2, family = "binomial", data = Weekly)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.623  -1.261   1.001   1.083   1.506
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.22122    0.06147   3.599 0.000319 ***
## Lag1        -0.03872    0.02622  -1.477 0.139672
## Lag2         0.06025    0.02655   2.270 0.023232 *
## ---
## 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: 1488.2  on 1086  degrees of freedom
## AIC: 1494.2
##
## Number of Fisher Scoring iterations: 4
```

4.b. Now we do as in part (a) except we exclude the first observation from the data set.

```
fit <- glm(Direction ~ Lag1 + Lag2, data=Weekly[-1,], family="binomial")
summary(fit)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2, family = "binomial", data = Weekly[-1,
##     ])
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6258  -1.2617   0.9999   1.0819   1.5071
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.22324    0.06150   3.630 0.000283 ***
## Lag1        -0.03843    0.02622  -1.466 0.142683
## Lag2         0.06085    0.02656   2.291 0.021971 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1494.6  on 1087  degrees of freedom
## Residual deviance: 1486.5  on 1085  degrees of freedom
## AIC: 1492.5
##
## Number of Fisher Scoring iterations: 4
```

4.c. We make a prediction for the first observation using the model from part (b).

```
prob <- predict.glm(fit, newdata=Weekly[1,], type="response")
actual <- Weekly[1, "Direction"]
pred <- ifelse(prob > 0.5, "Up", "Down")
pred == actual
```

```
##     1
## FALSE
```

4.d. Now we use parts (b) and (c) as the basis for manually determing the leave on out cross validation error estimate. We do this by looping through the rows of the dataset and for each loop we determine whether we have made an accurate prediction or not

```
errs <- vector()
for (i in 1:nrow(Weekly)) {
  fit <- glm(Direction ~ Lag1 + Lag2, data=Weekly[-i, ], family="binomial") #i
  prob <- predict.glm(fit, newdata=Weekly[i, ], type="response") #ii
  pred <- ifelse(prob > 0.5, "Up", "Down") #iii
  actual <- Weekly[i, "Direction"]
  errs[i] <- ifelse(pred == actual, 0, 1) #iv
}
```

4.e. Using the results from part (d) the leave one out cross validation error is given by the mean of the vector.

```
mean(errs)
```

```
## [1] 0.4499541
```

To compare, we can use the `cv.glm()` function from the `boot` package to compute the LOOCV error directly.

```
require(boot)
```

```
## Loading required package: boot
```

```
fit <- glm(Direction ~ Lag1 + Lag2, data=Weekly, family="binomial")
cost <- function(r, pi = 0) mean(abs(r-pi) > 0.5)
loocv_err <- cv.glm(Weekly, fit, cost)
loocv_err$delta[1]
```

```
## [1] 0.4499541
```

We can see that the results obtained are identical. Note that for `cv.glm()`, we have used the cost function recommended in the package documentation.

```
# clean up before moving on to next question
rm(list = ls())
```

5. LOOCV (Textbook 5.8)

5.a. We start by generating the simulated data as directed.

```r
set.seed(1)
y <- rnorm(100)
x <- rnorm(100)
y <- x - 2*x^2 + rnorm(100)
```

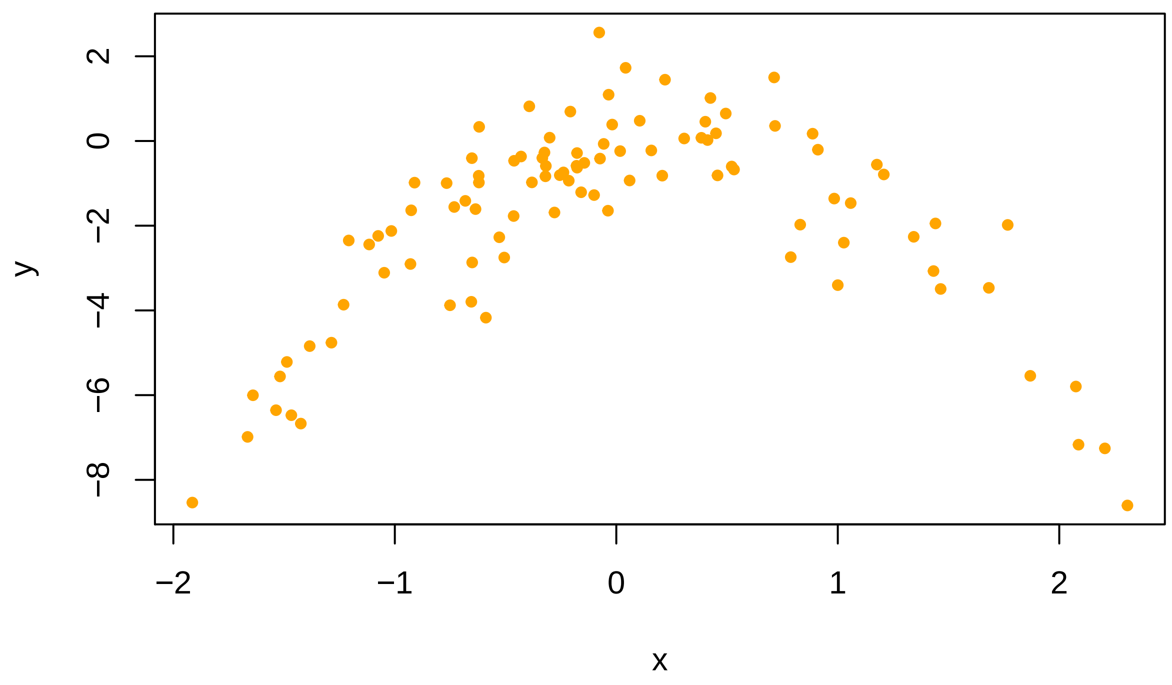
Here we have `n = 100` and `p = 2`. In mathematical notation, the model is:

$$Y = f(X) + \epsilon = X - 2X^2 + \epsilon$$

where $\epsilon \sim N(0, 1)$.

5.b. Next we create a new dataframe and then plot a scatterplot of `y` against `x`.

```r
df <- data.frame(y=y, x=x)
plot(df$y ~ df$x, xlab="x", ylab="y", pch=20, col="orange")
```



The plot shows the expected quadratic relationship between `x` and `y`.

5.c. Since we are going to be computing leave one out cross validation errors for models of varying degree, we start by creating a function that will easily allow us to generate LOOCV error estimates for a given degree.

```r
require(boot)
loocv_of_degree <- function(degree) {
  fit <- glm(y ~ poly(x, degree, raw=TRUE), data=df)
  loocv <- cv.glm(df, fit)
  return(loocv$delta[1])
}
```

To check that the function is working as expected we can compare the LOOCV error obtained by the usual method

```
fit <- glm(y ~ poly(x, 1, raw=TRUE), data=df)
loocv <- cv.glm(df, fit)
loocv$delta[1]
```

```
## [1] 5.890979
```

with that generated by our function.

```
loocv_of_degree(1)
```

```
## [1] 5.890979
```

The results are identical. Now we set a random seed and generate LOOCV estimates for the four models

```
set.seed(2)
degrees = 1:4
loocv_errs1 <- sapply(degrees, loocv_of_degree)
loocv_errs1
```

```
## [1] 5.890979 1.086596 1.102585 1.114772
```

5.d. Using a second random seed, the LOOCV estimates are as follows

```
set.seed(3)
degrees = 1:4
loocv_errs2 <- sapply(degrees, loocv_of_degree)
loocv_errs2
```

```
## [1] 5.890979 1.086596 1.102585 1.114772
```

By inspection we see that the LOOCV estimates are the same in each case. The reason for this is because there is no random sampling in leave one out cross validation - every observation is left out once to serve as the test set.

5.e. From the results in part (d) we see that the model with the lowest LOOCV estimate is the model with polynomial of degree 2. This is exactly what we would expect given that the function from which the data were simulated is a polynomial of degree 2.

5.f. The fit summaries for the four models in part (c) are as follows:

```
summary(glm(y ~ poly(x, 1, raw=TRUE), data=df))
```

```
##
## Call:
## glm(formula = y ~ poly(x, 1, raw = TRUE), data = df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -7.3469  -0.9275   0.8028   1.5608   4.3974
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)             -1.8185     0.2364  -7.692 1.14e-11 ***
## poly(x, 1, raw = TRUE)   0.2430     0.2479   0.981    0.329
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 5.580018)
##
##     Null deviance: 552.21  on 99  degrees of freedom
## Residual deviance: 546.84  on 98  degrees of freedom
## AIC: 459.69
##
## Number of Fisher Scoring iterations: 2
```

```
summary(glm(y ~ poly(x, 2, raw=TRUE), data=df))
```

```
##
## Call:
## glm(formula = y ~ poly(x, 2, raw = TRUE), data = df)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.89884  -0.53765   0.04135   0.61490   2.73607
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)             -0.09544    0.13345  -0.715    0.476
## poly(x, 2, raw = TRUE)1  0.89961    0.11300   7.961 3.24e-12 ***
## poly(x, 2, raw = TRUE)2 -1.86665    0.09151 -20.399  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 1.06575)
##
##     Null deviance: 552.21  on 99  degrees of freedom
## Residual deviance: 103.38  on 97  degrees of freedom
## AIC: 295.11
##
## Number of Fisher Scoring iterations: 2
```

```
summary(glm(y ~ poly(x, 3, raw=TRUE), data=df))
```

```
##
## Call:
## glm(formula = y ~ poly(x, 3, raw = TRUE), data = df)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.87250  -0.53881   0.02862   0.59383   2.74350
##
```

```
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)              -0.09865    0.13453  -0.733    0.465
## poly(x, 3, raw = TRUE)1  0.95551    0.22150   4.314  3.9e-05 ***
## poly(x, 3, raw = TRUE)2 -1.85303    0.10296 -17.998  < 2e-16 ***
## poly(x, 3, raw = TRUE)3 -0.02479    0.08435  -0.294    0.769
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 1.075883)
##
##     Null deviance: 552.21  on 99  degrees of freedom
## Residual deviance: 103.28  on 96  degrees of freedom
## AIC: 297.02
##
## Number of Fisher Scoring iterations: 2
```

```
summary(glm(y ~ poly(x, 4, raw=TRUE), data=df))
```

```
##
## Call:
## glm(formula = y ~ poly(x, 4, raw = TRUE), data = df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8914  -0.5244   0.0749   0.5932   2.7796
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)              -0.13897    0.15973  -0.870 0.386455
## poly(x, 4, raw = TRUE)1  0.90980    0.24249   3.752 0.000302 ***
## poly(x, 4, raw = TRUE)2 -1.72802    0.28379  -6.089  2.4e-08 ***
## poly(x, 4, raw = TRUE)3  0.00715    0.10832   0.066 0.947510
## poly(x, 4, raw = TRUE)4 -0.03807    0.08049  -0.473 0.637291
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 1.084654)
##
##     Null deviance: 552.21  on 99  degrees of freedom
## Residual deviance: 103.04  on 95  degrees of freedom
## AIC: 298.78
##
## Number of Fisher Scoring iterations: 2
```

The summaries show that for those models of degree 2 or higher, only the coefficients on the linear and the quadratic terms are significant. This is as expected given that the simulated data are drawn from a model that involves a linear and a quadratic term.

In addition, we see that for the model of degree 2 we have $\hat{\beta}_0$ = 0.8996 with confidence interval [0.6736, 1.1256], which includes 1, and for $\hat{\beta}_1$ = -1.8666 with confidence interval [-2.0496, -1.6836], which includes -2.