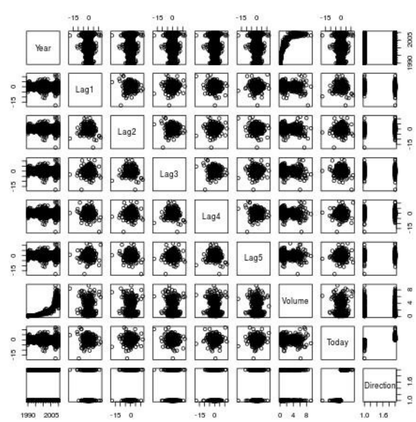
10)

a)

```
> library(ISLR)
> 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  
```

```
> jpeg("pairs-10a.jpg")
> pairs(Weekly)
> dev.off()
```




```
> cor(Weekly[, -9])

              Year         Lag1        Lag2        Lag3         Lag4
Year    1.00000000 -0.032289274 -0.03339001 -0.03000649 -0.031127923
Lag1   -0.03228927  1.000000000 -0.07485305  0.05863568 -0.071273876
Lag2   -0.03339001 -0.074853051  1.00000000 -0.07572091  0.058381535
Lag3   -0.03000649  0.058635682 -0.07572091  1.00000000 -0.075395865
Lag4   -0.03112792 -0.071273876  0.05838153 -0.07539587  1.000000000
Lag5   -0.03051910 -0.008183096 -0.07249948  0.06065717 -0.075675027
Volume  0.84194162 -0.064951313 -0.08551314 -0.06928771 -0.061074617
Today  -0.03245989 -0.075031842  0.05916672 -0.07124364 -0.007825873

               Lag5      Volume        Today
Year   -0.030519101  0.84194162 -0.032459894
Lag1   -0.008183096 -0.06495131 -0.075031842
Lag2   -0.072499482 -0.08551314  0.059166717
Lag3    0.060657175 -0.06928771 -0.071243639
Lag4   -0.075675027 -0.06107462 -0.007825873
Lag5    1.000000000 -0.05851741  0.011012698
```

```
Volume -0.058517414  1.00000000 -0.033077783
Today   0.011012698 -0.03307778  1.000000000
```

Year and Volume appear to be related. No other pattern is conceivable.

b)

```
> attach(Weekly)
The following objects are masked from Weekly (pos = 3):

    Direction, Lag1, Lag2, Lag3, Lag4, Lag5, Today, Volume, Year

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

Lag2 appears to have some statistical significance with Pr = 0.0296

c)

```
> glm.probs = predict(glm.fit, type = "response")
> glm.pred = rep("Down", length(glm.probs))
```

```
> glm.pred[glm.probs > 0.5] = "Up"
> table(glm.pred, Direction)
        Direction
glm.pred Down  Up
    Down   54   48
    Up    430  557
```

Percentage of current predictions: (54+557)/(54+557+48+430) = 0.561
Weeks the market goes up, the logistic regression is correct most of the time: 557/(557+48) = 0.921
Weeks the market goes up, the logistic regression is wrong most of the time: 48/(557+48) = 0.0793

d)

```
> train = (Year < 2009)
> Weekly.2009.2010 = Weekly[!train, ]
> glm.fit = glm(Direction ~ Lag2, data = Weekly, family = binomial, subset = train)
> glm.probs = predict(glm.fit, Weekly.2009.2010 , type = "response")
> glm.pred = rep("Down", length(glm.probs))
> glm.pred[glm.probs > 0.5] = "Up"
> Direction. 2009.2010 = Direction[!train]
> table(glm.pred, Direction.2009.2010 )
        Direction.2009.2010
glm.pred Down Up
    Down    9   5
    Up     34  56

> mean(glm.pred == Direction.2009.2010 )
[1] 0.625
```

e)

```
> library(MASS)
> lda.fit = lda(Direction ~ Lag2, data = Weekly, subset = train)
> lda.pred = predict(lda.fit, Weekly.2009.2010 )
> table(lda.pred$class, Direction.2009.2010 )
      Direction.2009.2010
       Down Up
  Down    9   5
  Up     34  56

> mean(lda.pred$class == Direction.2009.2010 )
[1] 0.625
```

f)

```
> qda.fit = qda(Direction ~ Lag2, data = Weekly, subset = train)
> qda.class = predict(qda.fit, Weekly.2009.2010)$class
> table(qda.class, Direction.2009.2010)
         Direction.2009.2010
qda.class Down Up
```

```
    Down    0   0
    Up     43  61

> mean(qda.class == Direction.2009.2010)
[1] 0.5865385
```

The correctness is 0.58, with all predictions are up

g)

```
> library(class)
> train.X = as.matrix(Lag2[train])
> test.X = as.matrix(Lag2[!train])
> train.Direction = Direction[train]
> set.seed(1)
> knn.pred = knn(train.X, test.X, train.Direction, k = 1)
> table(knn.pred, Direction.2009.2010)
        Direction.2009.2010
knn.pred Down Up
    Down   21  30
    Up     22  31

> mean(knn.pred == Direction.2009.2010)
[1] 0.5
```

h)

Logistic regression and LDA provide similar test error rates

i)

```
> # Logistic regression with Lag2:Lag1
> glm.fit = glm(Direction ~ Lag2:Lag1, data = Weekly, family = binomial, subset = train)
> glm.probs = predict(glm.fit, Weekly.2009.2010 , type = "response")
> glm.pred = rep("Down", length(glm.probs))
> glm.pred[glm.probs > 0.5] = "Up"
> Direction. 2009.2010 = Direction[!train]
> table(glm.pred, Direction.2009.2010)
        Direction.2009.2010
glm.pred Down Up
    Down    1   1
    Up     42  60

> mean(glm.pred == Direction.2009.2010)
[1] 0.5865385

> # LDA with Lag2 interaction with Lag1
> lda.fit = lda(Direction ~ Lag2:Lag1, data = Weekly, subset = train)
> lda.pred = predict(lda.fit, Weekly.2009.2010)
> mean(lda.pred$class == Direction.2009.2010)
[1] 0.5769231
```

```
> # QDA with sqrt(abs(Lag2))
> qda.fit = qda(Direction ~ Lag2 + sqrt(abs(Lag2)), data = Weekly, subset = train)
> qda.class = predict(qda.fit, Weekly.2009.2010)$class
> table(qda.class, Direction.2009.2010)
          Direction.2009.2010
qda.class Down Up
     Down   12  13
     Up     31  48

> mean(qda.class == Direction.2009.2010)
[1] 0.5769231

> # KNN k =10
> knn.pred = knn(train.X, test.X, train.Direction, k = 10)
> table(knn.pred, Direction.2009.2010)
         Direction.2009.2010
knn.pred Down Up
    Down   17  18
    Up     26  43

> mean(knn.pred == Direction.2009.2010)
[1] 0.5769231

> # KNN k = 100
> knn.pred = knn(train.X, test.X, train.Direction, k = 100)
> table(knn.pred, Direction.2009.2010)
         Direction.2009.2010
knn.pred Down Up
    Down    9 12
    Up     34 49

> mean(knn.pred == Direction.2009.2010)
[1] 0.5576923
```

Overall, the original LDA and logistic regression provide better performance in terms of test error rates

11)

a)

```
> library(ISLR)
> summary(Auto)
      mpg          cylinders      displacement     horsepower        weight
 Min.   : 9.00   Min.   :3.000   Min.   : 68.0   Min.   : 46.0   Min.   :1613
 1st Qu.:17.00   1st Qu.:4.000   1st Qu.:105.0   1st Qu.: 75.0   1st Qu.:2225
 Median :22.75   Median :4.000   Median :151.0   Median : 93.5   Median :2804
 Mean   :23.45   Mean   :5.472   Mean   :194.4   Mean   :104.5   Mean   :2978
 3rd Qu.:29.00   3rd Qu.:8.000   3rd Qu.:275.8   3rd Qu.:126.0   3rd Qu.:3615
 Max.   :46.60   Max.   :8.000   Max.   :455.0   Max.   :230.0   Max.   :5140
```

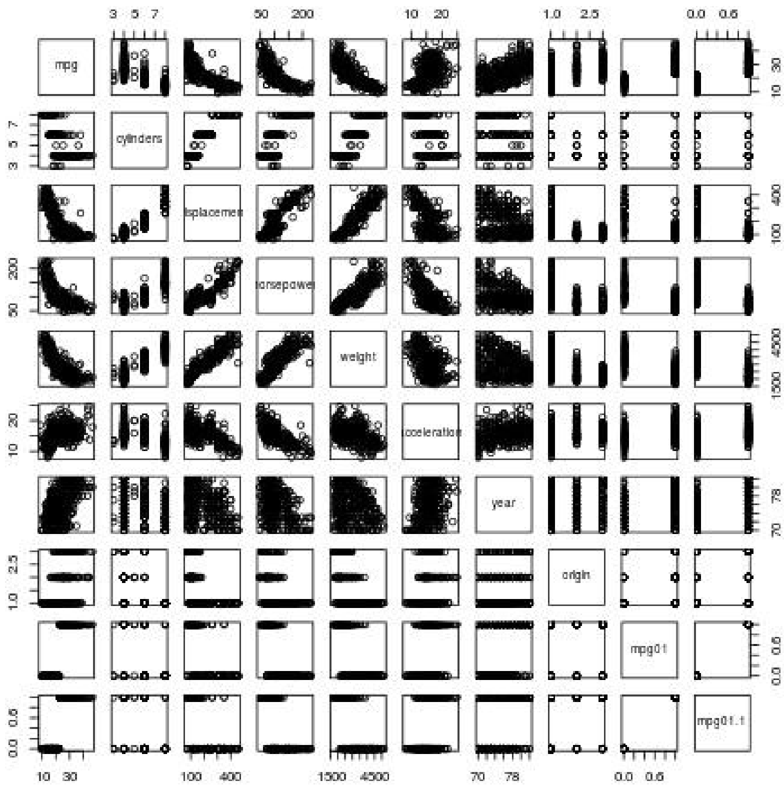
```
  acceleration        year           origin                      name    
 Min.   : 8.00   Min.   :70.00   Min.   :1.000   amc matador       :  5  
 1st Qu.:13.78   1st Qu.:73.00   1st Qu.:1.000   ford pinto        :  5  
 Median :15.50   Median :76.00   Median :1.000   toyota corolla    :  5  
 Mean   :15.54   Mean   :75.98   Mean   :1.577   amc gremlin       :  4  
 3rd Qu.:17.02   3rd Qu.:79.00   3rd Qu.:2.000   amc hornet        :  4  
 Max.   :24.80   Max.   :82.00   Max.   :3.000   chevrolet chevette:  4  
                                                 (Other)           :365  

> mpg01 = rep(0, length(mpg))
> mpg01[mpg > median(mpg)] = 1
> Auto = data.frame(Auto, mpg01)

> head(Auto)
  mpg cylinders displacement horsepower weight acceleration year origin
1  18         8          307        130   3504         12.0   70      1
2  15         8          350        165   3693         11.5   70      1
3  18         8          318        150   3436         11.0   70      1
4  16         8          304        150   3433         12.0   70      1
5  17         8          302        140   3449         10.5   70      1
6  15         8          429        198   4341         10.0   70      1
                       name mpg01 mpg01.1
1 chevrolet chevelle malibu     0       0
2         buick skylark 320     0       0
3        plymouth satellite     0       0
4             amc rebel sst     0       0
5               ford torino     0       0
6          ford galaxie 500     0       0
```

b)

```
> cor(Auto[, -9])
                    mpg  cylinders displacement horsepower     weight
mpg           1.0000000 -0.7776175   -0.8051269 -0.7784268 -0.8322442
cylinders    -0.7776175  1.0000000    0.9508233  0.8429834  0.8975273
displacement -0.8051269  0.9508233    1.0000000  0.8972570  0.9329944
horsepower   -0.7784268  0.8429834    0.8972570  1.0000000  0.8645377
weight       -0.8322442  0.8975273    0.9329944  0.8645377  1.0000000
acceleration  0.4233285 -0.5046834   -0.5438005 -0.6891955 -0.4168392
year          0.5805410 -0.3456474   -0.3698552 -0.4163615 -0.3091199
origin        0.5652088 -0.5689316   -0.6145351 -0.4551715 -0.5850054
mpg01         0.8369392 -0.7591939   -0.7534766 -0.6670526 -0.7577566
mpg01.1       0.8369392 -0.7591939   -0.7534766 -0.6670526 -0.7577566
             acceleration       year     origin      mpg01    mpg01.1
mpg             0.4233285  0.5805410  0.5652088  0.8369392  0.8369392
cylinders      -0.5046834 -0.3456474 -0.5689316 -0.7591939 -0.7591939
displacement   -0.5438005 -0.3698552 -0.6145351 -0.7534766 -0.7534766
horsepower     -0.6891955 -0.4163615 -0.4551715 -0.6670526 -0.6670526
weight         -0.4168392 -0.3091199 -0.5850054 -0.7577566 -0.7577566
```

```
acceleration    1.0000000  0.2903161  0.2127458  0.3468215  0.3468215
year            0.2903161  1.0000000  0.1815277  0.4299042  0.4299042
origin          0.2127458  0.1815277  1.0000000  0.5136984  0.5136984
mpg01           0.3468215  0.4299042  0.5136984  1.0000000  1.0000000
mpg01.1         0.3468215  0.4299042  0.5136984  1.0000000  1.0000000
```

```
> jpeg("pairs-11b")
> pairs(Auto[,-9])
> dev.off()
```



Having a negative
correlation with cylinders, weight, displacement, horsepower and mpg.

c)

```
> train = (year%%2 == 0)  # if the year is even, data is training
> test = !train
> Auto.train = Auto[train, ]
> Auto.test = Auto[test, ]
> mpg01.test = mpg01[test]
```

d)

```
> # LDA
> library(MASS)
> lda.fit = lda(mpg01 ~ cylinders + weight + displacement + horsepower, data = Auto, subset = train)
> lda.pred = predict(lda.fit, Auto.test)
> mean(lda.pred$class != mpg01.test)
[1] 0.1263736
```

Test error rate: 0.1263736

e)
```
> # QDA
> qda.fit = qda(mpg01 ~ cylinders + weight + displacement + horsepower, data = Auto, subset = train)
> qda.pred = predict(qda.fit, Auto.test)
> mean(qda.pred$class != mpg01.test)
[1] 0.1318681
```

Test error rate: 0.1318681

f)
```
> # Logistic regression
> glm.fit = glm(mpg01 ~ cylinders + weight + displacement + horsepower, data = Auto, family =
binomial, subset = train)
> glm.probs = predict(glm.fit, Auto.test, type = "response")
> glm.pred = rep(0, length(glm.probs))
> glm.pred[glm.probs > 0.5] = 1
> mean(glm.pred != mpg01.test)
[1] 0.1208791
```

Test error rate: 0.1208791

g)
```
> library(class)
> train.X = cbind(cylinders, weight, displacement, horsepower)[train, ]
> test.X = cbind(cylinders, weight, displacement, horsepower)[test, ]
> train.mpg01 = mpg01[train]
> set.seed(1)
> # KNN(k=1)
> knn.pred = knn(train.X, test.X, train.mpg01, k = 1)
> mean(knn.pred != mpg01.test)
[1] 0.1538462

> # KNN(k=10)
> knn.pred = knn(train.X, test.X, train.mpg01, k = 10)
> mean(knn.pred != mpg01.test)
[1] 0.1648352

> # KNN(k=100)
> knn.pred = knn(train.X, test.X, train.mpg01, k = 100)
> mean(knn.pred != mpg01.test)
[1] 0.1428571
```

Test error rate (k=1): 0.1538462
Test error rate (k=10): 0.1648352
Test error rate (k=100): 0.1428571
with k=100, KNN produces the most significant performance.