DSO 530 - Homework 2

Xueyan Gu

2/24/2019

ISLR Chapter 4

4.7 Exercises

7. Answer: If we put the parameters and given number into the formula $p_k(x) = P(Y = k|X = x)$:

$$p_1(4) = \frac{0.8e^{(-1/72)(4-10)^2}}{0.8e^{(-1/72)(4-10)^2} + 0.2e^{(-1/72)(4-0)^2}} = 0.752$$

So, we can get that the probability that a company will issue a dividend this year given that its percentage return was X=4 last year is 0.752.

10. Answer:

(a)

```
library(ISLR)
summary(Weekly)
```

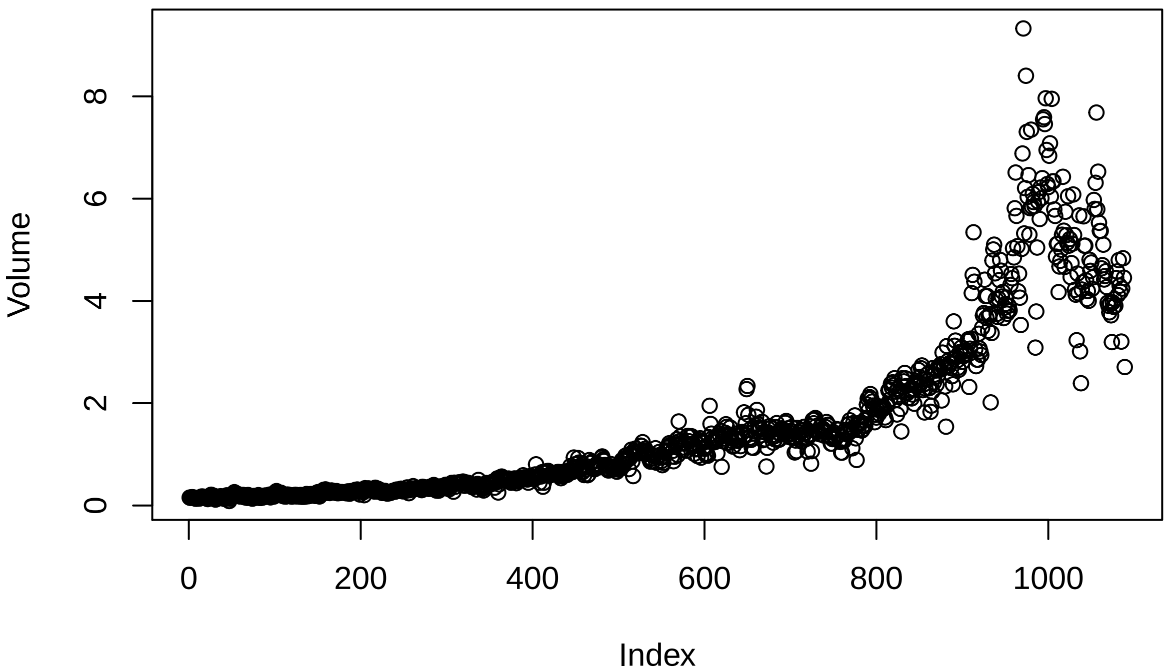
```
##       Year          Lag1               Lag2               Lag3
##  Min.   :1990   Min.   :-18.1950   Min.   :-18.1950   Min.   :-18.1950
##  1st Qu.:1995   1st Qu.: -1.1540   1st Qu.: -1.1540   1st Qu.: -1.1580
##  Median :2000   Median :  0.2410   Median :  0.2410   Median :  0.2410
##  Mean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

```
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
```

```r
attach(Weekly)
plot(Volume)
```



As we can see from the graph, the correlations between the “lag1” to “lag5” variables and “today’s returns” variables are close to zero. The correlation between “Year” and “Volume”, which is 0.84, is the most substantial. If we plot “Volume” variable, we can see that “Volume” is increasing as time went by.

(b) Logistic regression:

```r
fit.glm1 = glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume, data = Weekly, family = binomial)
summary(fit.glm1)
```

```
##
## 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
```

As we can see from the result, "Lag2" variable is the statistically significant predictor because its p-value is less than 0.05.

(c) Confusion matrix:

```r
probs1 = predict(fit.glm1, type = "response")
pred.glm1 = rep("Down", length(probs1))
pred.glm1[probs1 > 0.5] = "Up"
table(pred.glm1, Direction)
```

```
##          Direction
## pred.glm1 Down  Up
##      Down   54  48
##      Up    430 557
```

As we can see from the result, the percentage of correct predictions on the data set is (54+557)/1089, which is equal to 56.11%. That is to say, the training error rate is (1-56.11%), which is 43.89%. Further, we can conclude that when the market goes up, the model gives correct predictions 557/(48+557), which is 92.07% of the time. Also, when the market goes down, the model gives correct predictions 54/(54+430), which is 11.16% of the time.

(d) Fit the model:

```r
# Filter the data

training = (Year < 2009)
Weekly_20092010 = Weekly[!training, ]
Direction_20092010 = Direction[!training]
fit.glm2 = glm(Direction ~ Lag2, data = Weekly, family = binomial, subset = training)
summary(fit.glm2)
```

```
##
## Call:
## glm(formula = Direction ~ Lag2, family = binomial, data = Weekly,
##     subset = training)
##
## 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
```

```r
# Build up a confusion matrix

probs2 = predict(fit.glm2, Weekly_20092010, type = "response")
pred.glm2 = rep("Down", length(probs2))
pred.glm2[probs2 > 0.5] = "Up"
table(pred.glm2, Direction_20092010)
```

```
##          Direction_20092010
## pred.glm2 Down Up
##      Down    9  5
##      Up     34 56
```

As we can see from the result, the percentage of correct predictions on the test data is (9+56)/104, which is equal to 62.5%. That is to say, the test error rate is (1-62.5%), which is 37.5%. Further, we can conclude that when the market goes up, the model gives correct predictions 56/(56+5), which is 91.80% of the time. Also, when the market goes down, the model gives correct predictions 9/(9+34), which is 20.93% of the time.

(e) LDA:

```r
# Build a LDA model

library(MASS)
fit.lda = lda(Direction ~ Lag2, data = Weekly, subset = training)
fit.lda
```

```
## Call:
## lda(Direction ~ Lag2, data = Weekly, subset = training)
##
## Prior probabilities of groups:
##      Down        Up
## 0.4477157 0.5522843
##
## Group means:
##             Lag2
## Down -0.03568254
## Up    0.26036581
##
## Coefficients of linear discriminants:
##            LD1
## Lag2 0.4414162
```

```r
# Build up a confusion matrix

pred.lda = predict(fit.lda, Weekly_20092010)
table(pred.lda$class, Direction_20092010)
```

```
##       Direction_20092010
##        Down Up
##   Down    9  5
##   Up     34 56
```

As we can see from the table, the result is almost the same as the result from logistic regression model. We can see that the percentage of correct predictions on the test data is (9+56)/104, which is equal to 62.5%. That is to say, the test error rate is (1-62.5%), which is 37.5%. Further, we can conclude that when the market goes up, the model gives correct predictions 56/(56+5), which is 91.80% of the time. Also, when the market goes down, the model gives correct predictions 9/(9+34), which is 20.93% of the time.

(f) QDA:

```r
# Build a QDA model

fit.qda = qda(Direction ~ Lag2, data = Weekly, subset = training)
fit.qda
```

```
## Call:
## qda(Direction ~ Lag2, data = Weekly, subset = training)
##
## Prior probabilities of groups:
##      Down        Up
## 0.4477157 0.5522843
##
## Group means:
##             Lag2
## Down -0.03568254
## Up    0.26036581
```

```r
#Build up a confusion matrix

pred.qda = predict(fit.qda, Weekly_20092010)
table(pred.qda$class, Direction_20092010)
```

```
##       Direction_20092010
##        Down Up
##   Down    0  0
##   Up     43 61
```

As we can see from the result, the percentage of correct predictions on the test data is (61+0)/104, which is equal to 58.65%. That is to say, the test error rate is (1-58.6538462%), which is 41.35%. Further, we can conclude that when the market goes up, the model gives correct predictions 100% of the time. However, when the market goes down, the model gives correct predictions only 0% of the time.

(g) KNN:

```r
# Build up a confusion matrix

library(class)
train.X = as.matrix(Lag2[training])
test.X = as.matrix(Lag2[!training])
train_Direction = Direction[training]
```

```r
set.seed(1)
pred.knn = knn(train.X, test.X, train_Direction, k = 1)
table(pred.knn, Direction_20092010)
```

```
##         Direction_20092010
## pred.knn Down Up
##     Down   21 30
##     Up     22 31
```

As we can see from the result, the percentage of correct predictions on the test data is (21+31)/104, which is equal to 50%. That is to say, the test error rate is 50%. Further, we can conclude that when the market goes up, the model gives correct predictions 31/(30+31), which is 50.82% of the time. Also, when the market goes down, the model gives correct predictions 21/(21+22), which is 48.84% of the time.

(h) Based on the results above, if we compare the test error rates of different methods, we can see that logistic regression and LDA have the smallest test error rates. Thus, logistic regression and LDA provide the best results on this data.

11. Answer:

(a)

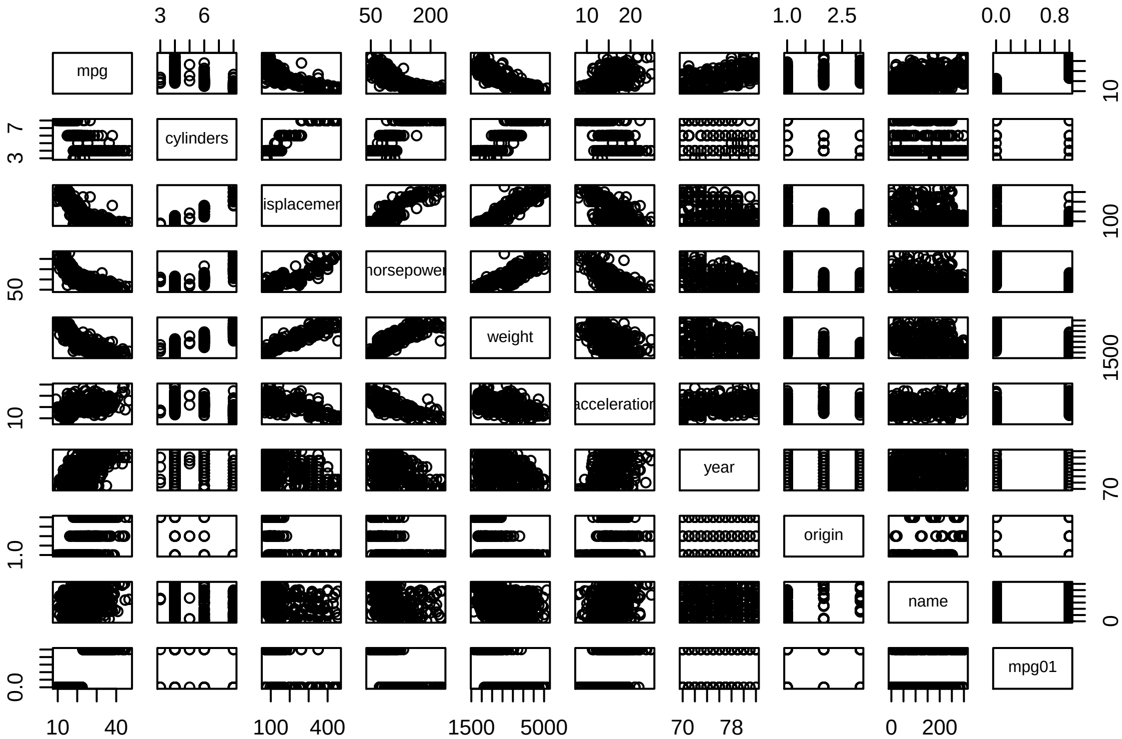
```r
# Create a dataset

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

(b)

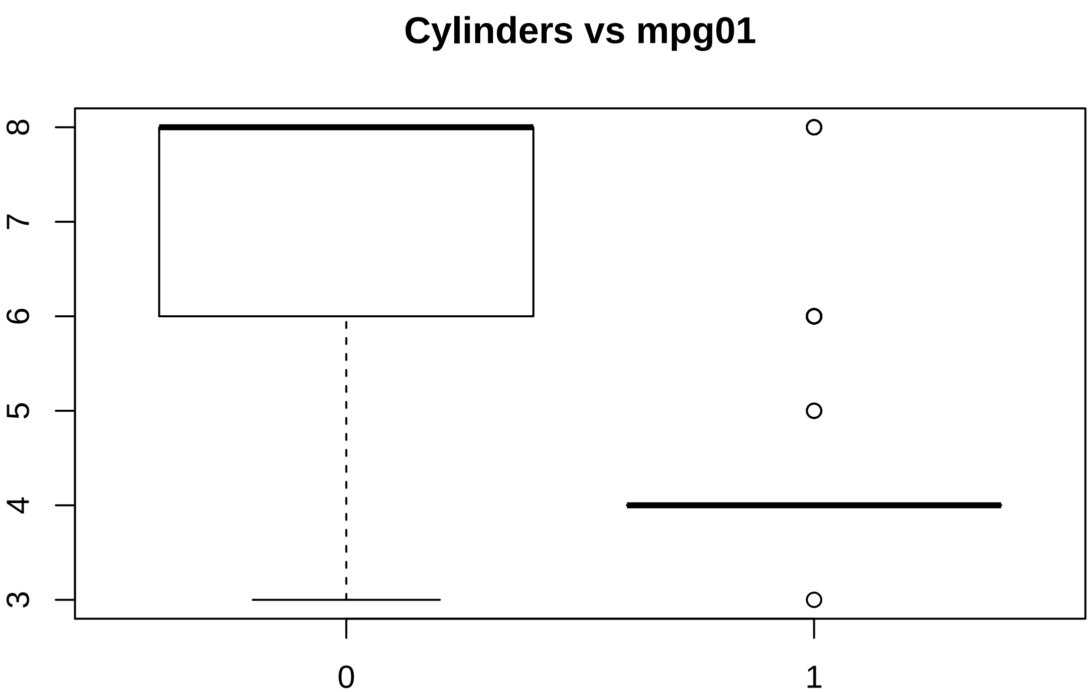
```r
cor(Auto[, -9])
```

```
##                     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
##              acceleration       year     origin      mpg01
## mpg             0.4233285  0.5805410  0.5652088  0.8369392
## cylinders      -0.5046834 -0.3456474 -0.5689316 -0.7591939
## displacement   -0.5438005 -0.3698552 -0.6145351 -0.7534766
## horsepower     -0.6891955 -0.4163615 -0.4551715 -0.6670526
## weight         -0.4168392 -0.3091199 -0.5850054 -0.7577566
## acceleration    1.0000000  0.2903161  0.2127458  0.3468215
## year            0.2903161  1.0000000  0.1815277  0.4299042
## origin          0.2127458  0.1815277  1.0000000  0.5136984
## mpg01           0.3468215  0.4299042  0.5136984  1.0000000
```

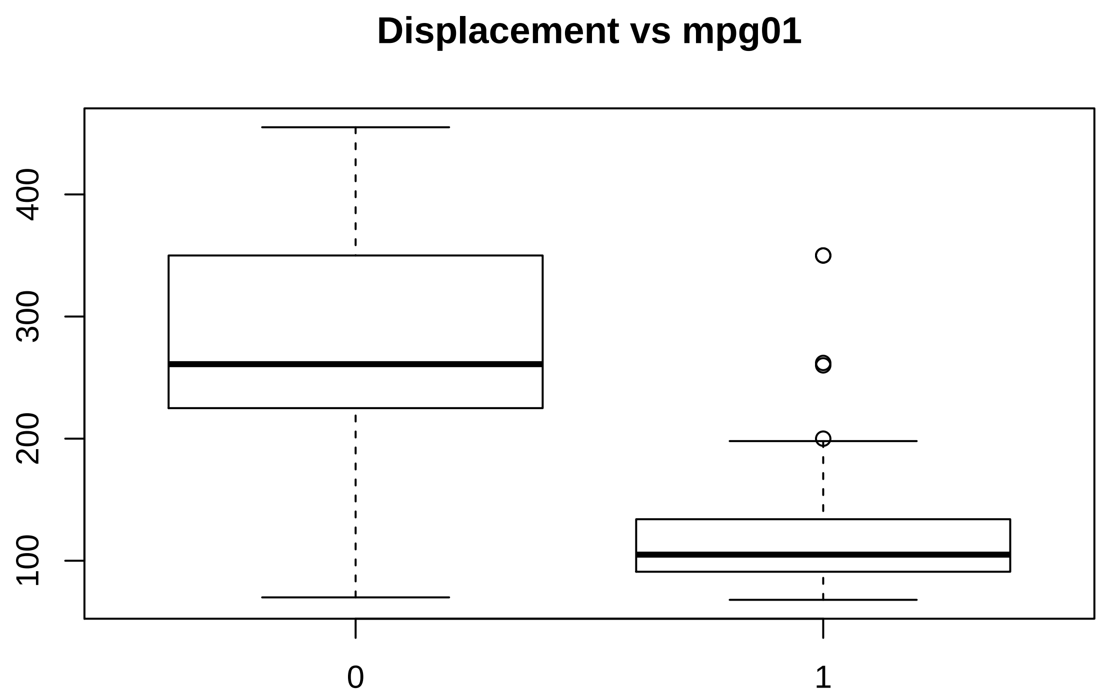
```r
pairs(Auto)
```



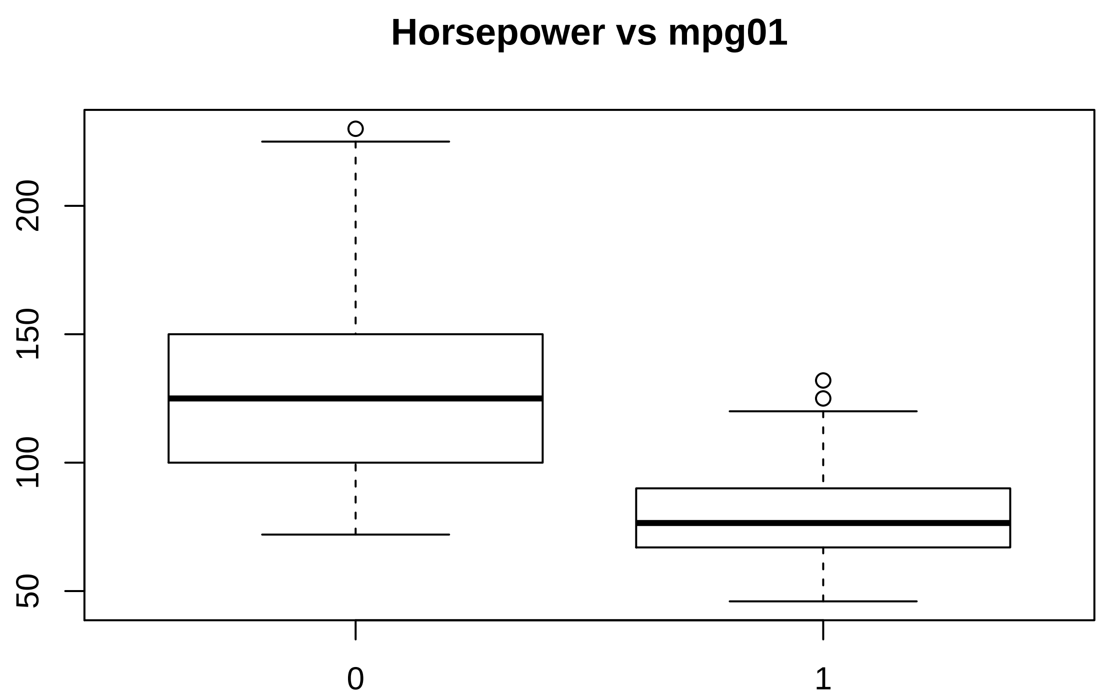

```r
boxplot(cylinders ~ mpg01, data = Auto, main = "Cylinders vs mpg01")
```



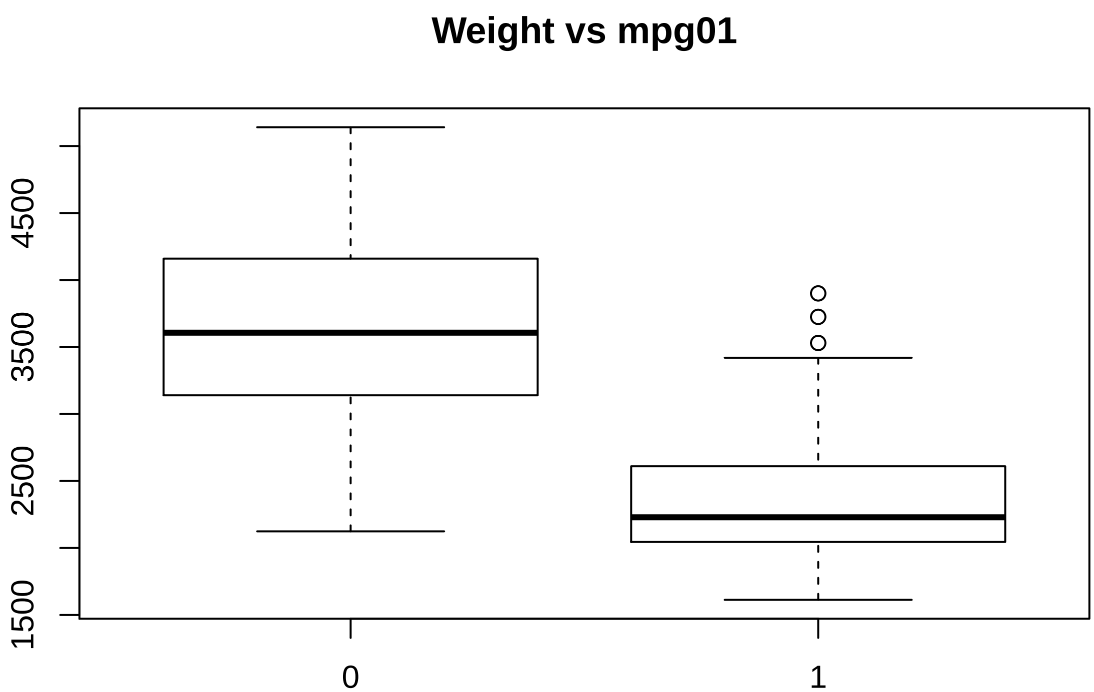

```r
boxplot(displacement ~ mpg01, data = Auto, main = "Displacement vs mpg01")
```



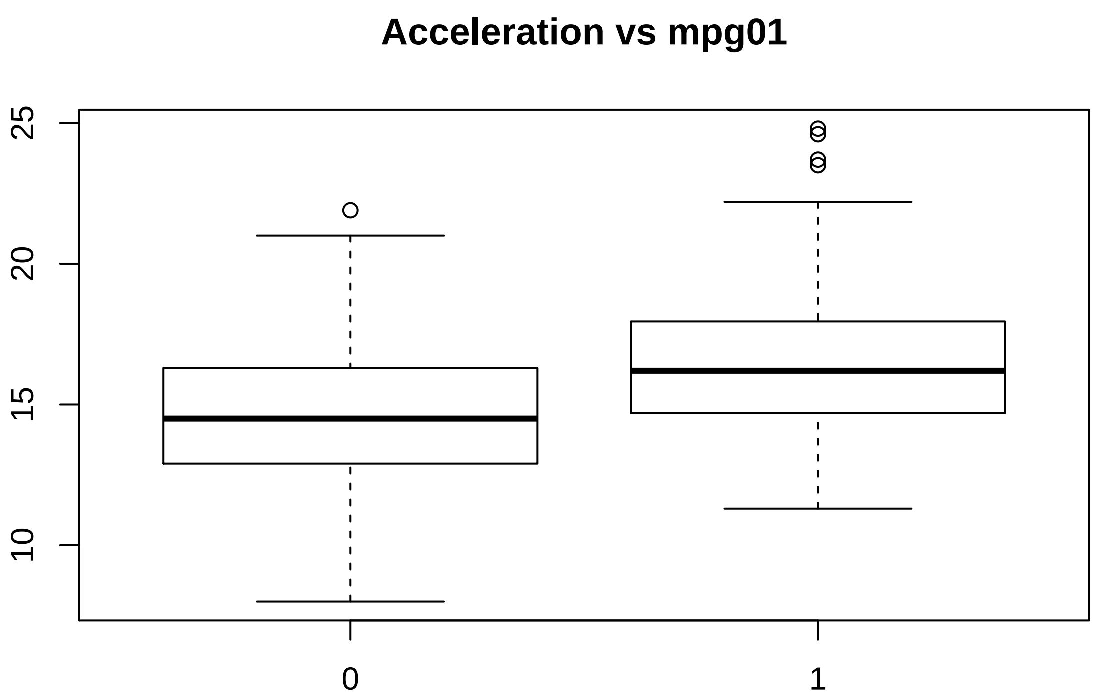

```
boxplot(horsepower ~ mpg01, data = Auto, main = "Horsepower vs mpg01")
```



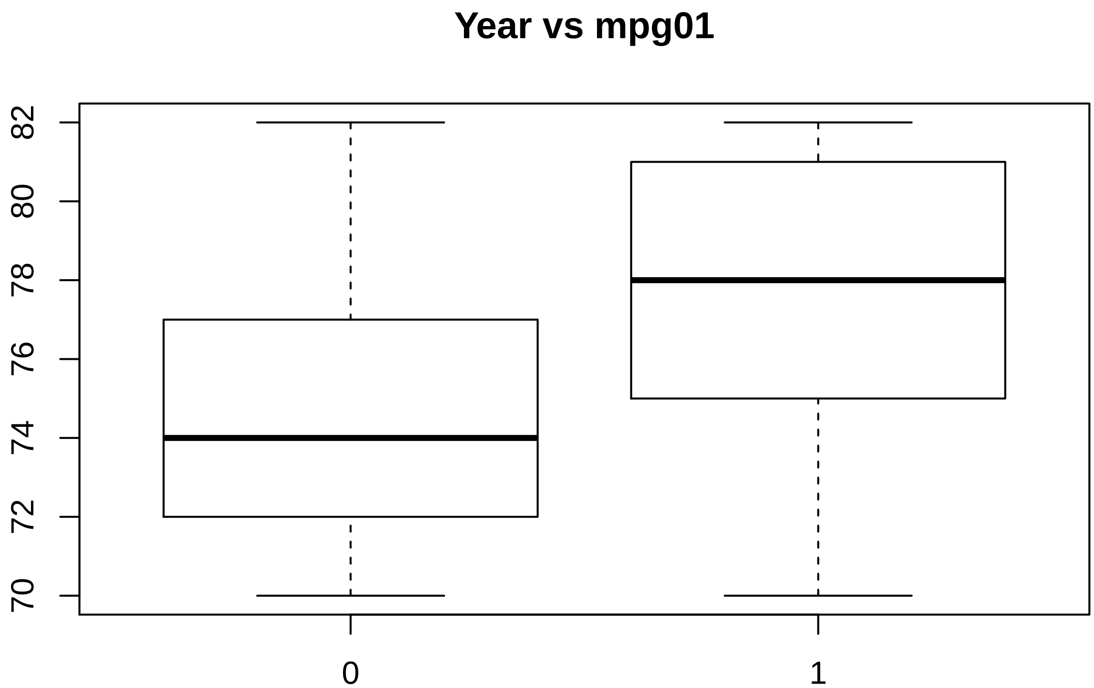

```
boxplot(weight ~ mpg01, data = Auto, main = "Weight vs mpg01")
```



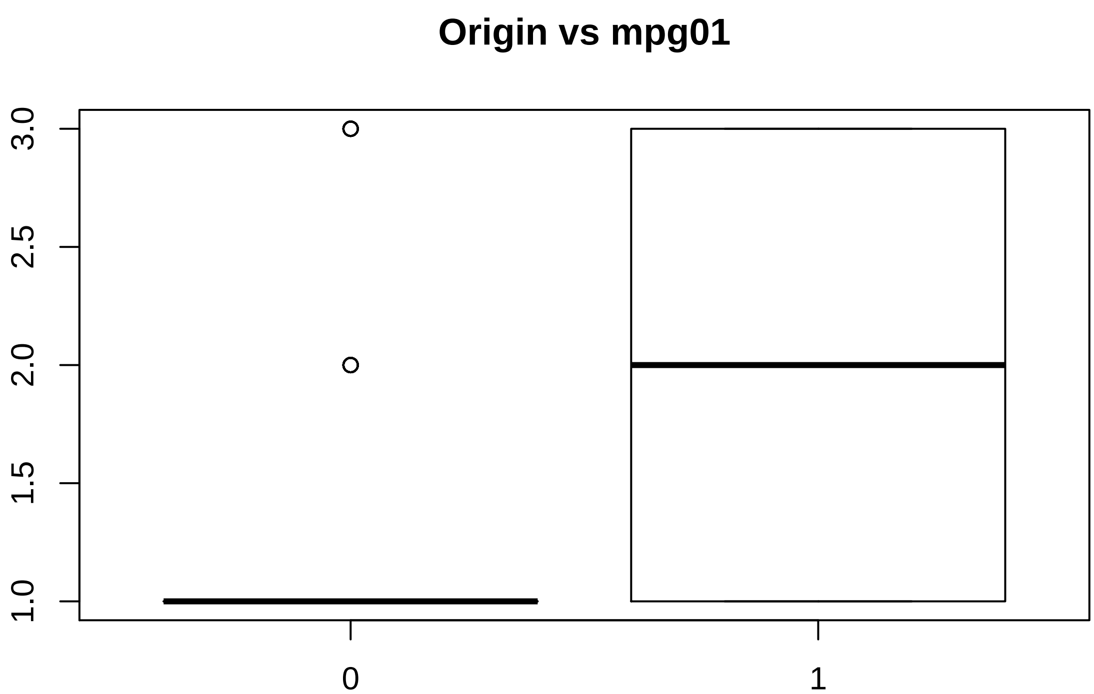

```
boxplot(acceleration ~ mpg01, data = Auto, main = "Acceleration vs mpg01")
```




```
boxplot(year ~ mpg01, data = Auto, main = "Year vs mpg01")
```




```r
boxplot(origin ~ mpg01, data = Auto, main = "Origin vs mpg01")
```



As we can see from the graphs above, it's possible that there are some associations between "mpg01" and "cylinders","displacement", "horsepower", and "weight" because of the higher correlations.

(c)

```r
# Split the data into a traning set and a test set:

train = (c(1:200))
Auto.train = Auto[train, ]
Auto.test = Auto[-c(1:200), ]
```

```r
mpg01.test = mpg01[-c(1:200)]
```

(d) LDA:

```r
fit.lda2 = lda(mpg01 ~ cylinders + weight + displacement + horsepower, data = Auto.train, subset=train)
fit.lda2
```

```
## Call:
## lda(mpg01 ~ cylinders + weight + displacement + horsepower, data = Auto.train,
##     subset = train)
##
## Prior probabilities of groups:
##    0    1
## 0.66 0.34
##
## Group means:
##   cylinders   weight displacement horsepower
## 0  6.848485 3674.818     284.7348  134.57576
## 1  4.058824 2233.676     105.5809   78.83824
##
## Coefficients of linear discriminants:
##                        LD1
## cylinders    -0.3457980629
## weight       -0.0007078441
## displacement -0.0079996328
## horsepower    0.0156182930
```

```r
pred.lda2 = predict(fit.lda2, Auto.test)
table(pred.lda2$class, mpg01.test)
```

```
##    mpg01.test
##       0   1
##   0  56  12
##   1   8 116
```

```r
mean(pred.lda2$class != mpg01.test)
```

```
## [1] 0.1041667
```

As we can see from the result, the test error rate of the model is (8+12)/192*100% = 10.42%.

(e) QDA:

```r
fit.qda2 = qda(mpg01 ~ cylinders + weight + displacement + horsepower, data = Auto.train, subset=train)
fit.qda2
```

```
## Call:
## qda(mpg01 ~ cylinders + weight + displacement + horsepower, data = Auto.train,
##     subset = train)
##
## Prior probabilities of groups:
##    0    1
## 0.66 0.34
##
## Group means:
##   cylinders   weight displacement horsepower
## 0  6.848485 3674.818     284.7348  134.57576
## 1  4.058824 2233.676     105.5809   78.83824
```

```r
pred.qda2 = predict(fit.qda2, Auto.test)
table(pred.qda2$class, mpg01.test)
```

```
##    mpg01.test
##       0   1
##   0  60  22
##   1   4 106
```

```r
mean(pred.qda2$class != mpg01.test)
```

```
## [1] 0.1354167
```

As we can see from the result, the test error rate of the model is (4+22)/192*100% = 13.54%.

(f) Losgistic:

```r
fit.glm2 = glm(mpg01 ~ cylinders + weight + displacement + horsepower, data = Auto.train, subset=train)
summary(fit.glm2)
```

```
##
## Call:
## glm(formula = mpg01 ~ cylinders + weight + displacement + horsepower,
##     data = Auto.train, subset = train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -0.94385  -0.17478   0.08549   0.22473   0.72133
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.340e+00  1.467e-01   9.137  < 2e-16 ***
## cylinders    -8.081e-02  4.269e-02  -1.893  0.05986 .
## weight       -1.654e-04  6.565e-05  -2.520  0.01255 *
## displacement -1.869e-03  8.446e-04  -2.213  0.02803 *
## horsepower    3.650e-03  1.162e-03   3.140  0.00195 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.09314573)
##
##     Null deviance: 44.880  on 199  degrees of freedom
## Residual deviance: 18.163  on 195  degrees of freedom
## AIC: 99.794
##
## Number of Fisher Scoring iterations: 2
```

```r
probs2 = predict(fit.glm2, Auto.test, type = "response")
pred.glm2 = rep(0, length(probs2))
pred.glm2[probs2 > 0.5] = 1
table(pred.glm2, mpg01.test)
```

```
##          mpg01.test
## pred.glm2   0   1
##         0  56  12
##         1   8 116
```

```r
mean(pred.glm2 != mpg01.test)
```

```
## [1] 0.1041667
```

As we can see from the result, the test error rate of the model is (8+12)/192*100% = 10.42%.

(g) KNN:

```r
library(class)
train.X = cbind(cylinders, weight, displacement, horsepower)[c(1:200), ]
test.X = cbind(cylinders, weight, displacement, horsepower)[-c(1:200), ]
train.mpg01 = mpg01[1:200]
set.seed(1)
pred.knn = knn(train.X, test.X, train.mpg01, k = 1)
table(pred.knn, mpg01.test)
```

```
##          mpg01.test
## pred.knn   0   1
##         0  61  28
##         1   3 100
```

```r
mean(pred.knn != mpg01.test)
```

```
## [1] 0.1614583
```

As we can see from the result, the test error rate of the model is 16.15% for k = 1.

```r
pred.knn = knn(train.X, test.X, train.mpg01, k = 10)
table(pred.knn, mpg01.test)
```

```
##          mpg01.test
## pred.knn   0   1
##         0  60  26
##         1   4 102
```

```r
mean(pred.knn != mpg01.test)
```

```
## [1] 0.15625
```

As we can see from the result, the test error rate of the model is 15.63% for k = 10.

```r
pred.knn = knn(train.X, test.X, train.mpg01, k = 100)
table(pred.knn, mpg01.test)
```

```
##          mpg01.test
## pred.knn   0   1
##         0  61  23
##         1   3 105
```

```r
mean(pred.knn != mpg01.test)
```

```
## [1] 0.1354167
```

As we can see from the result, the test error rate of the model is 13.54% for k = 100. Thus, k = 100 seems to perform the best on this data set.