Homework 3 Classification

Jinfei Xue

2/6/2019

4.6

(a)

We can obtain the logistic regression:

$$logit(p) = -6 + 0.05X_1 + X_2$$

```
library(boot)
inv.logit(-6+0.05*40+3.5)
```

```
## [1] 0.3775407
```

So the estimated probability that a student who studies for 40 h and has an undergrad GPA of 3.5 gets an A in the class is 0.3775407.

(b)

```
(logit(0.5)-(-6)-3.5)/0.05
```

```
## [1] 50
```

So the student in part (a) need to study 50 hours to have a 50% chance of getting an A in the class.

4.8

In the case of KNN with K=1, we have a training error rate of 0% because in this case, we have:

$$P(Y = j|X = x_i) = I(y_i = j)$$

which is equal to 1 if $y_i = j$ and 0 if not. We do not make any error on the training data within this setting, that explains the 0% training error rate. However, we have an average error rate of 18% wich implies a test error rate of 36% for KNN which is greater than the test error rate for logistic regression of 30%. So, it is better to choose logistic regression because of its lower test error rate.

4.9

(a)

We can obtain:

$$\frac{p}{1-p} = 0.37$$

We can transform it into:

$$p = \frac{0.37}{1+0.37} \approx 0.27$$

So, on average, 27% of people with an odds of 0.37 of defaulting on their credit card payment will in fact default.

(b)

The odds that she will default is:

$$\frac{p}{1-p} = \frac{0.16}{1-0.16} \approx 0.19$$

4.10

(a)

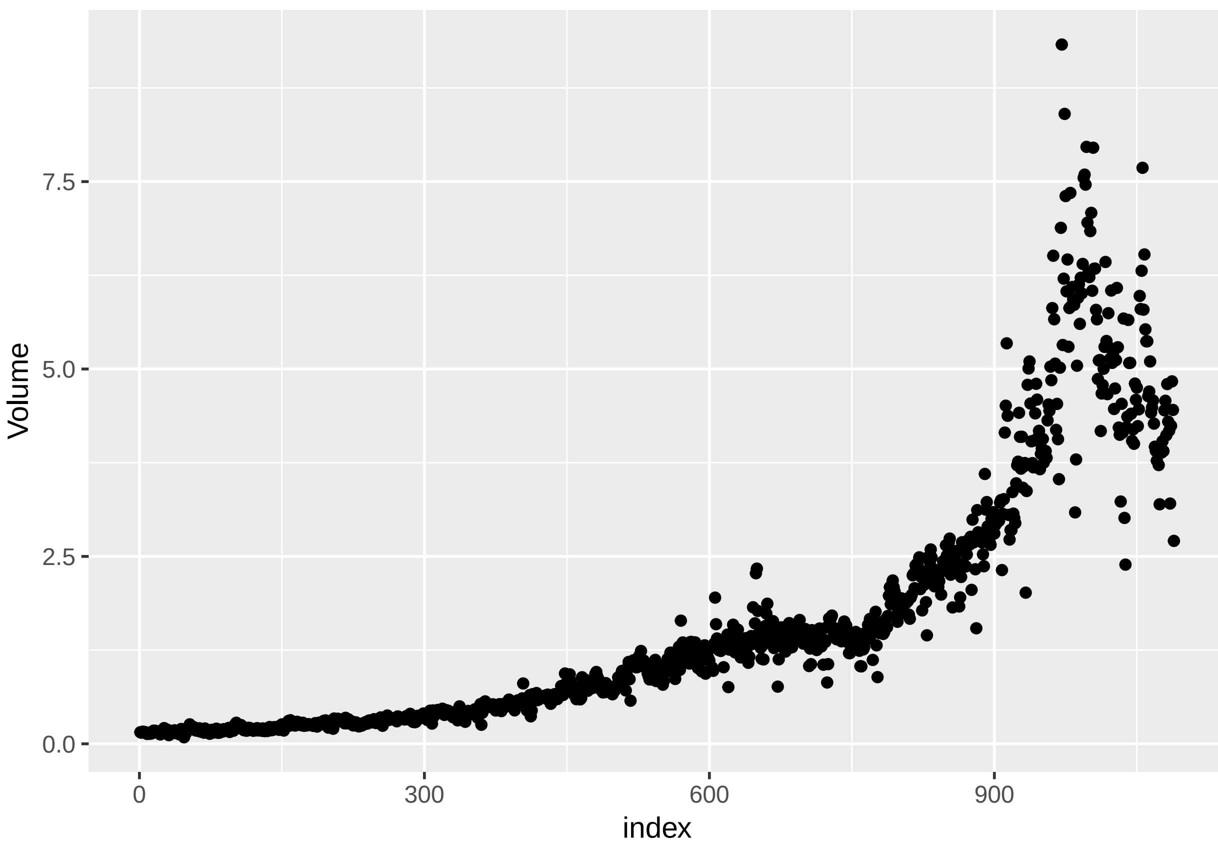
```r
library(ISLR)
data(Weekly)
Weekly = Weekly

# numerical summary
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
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
# graphical summary
library(ggplot2)
index=1:length(Weekly$Volume)
ggplot(Weekly) + geom_point(aes(x=index,y=Volume))
```



The correlations between lag variables and "Today" are close to zero. When we plot "Volume", we can see that it is increasing over time.

(b)

```r
r10_b <- glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume,
               data = Weekly, family = binomial)
summary(r10_b)
```

```
##
## 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
```

We can see that "Lag2" is the only predictor statistically significant because its p-value is less than 0.05.

(c)

```r
prob_b <- predict(r10_b, type = "response")
pred_b <- rep("Down", length(prob_b))
pred_b[prob_b > 0.5] <- "Up"
table(pred_b, Weekly$Direction)
```

```
##
## pred_b Down  Up
##   Down   54  48
##   Up    430 557
```

```r
mean(pred_b == Weekly$Direction)
```

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

We can conclude that this logistic regression correctly predicted the movement of the market 56.10652% of the *time and 43.89348% is the training error rate. We could also say that for weeks when the market goes up, the model is right 92.0661157% of the time (557/(48+557)). For weeks when the market goes down, the model is right only 11.1570248% of the time (54/(54+430)).*

(d)

```
train <- Weekly[Weekly$Year<2009, ]
r10_d <- glm(Direction ~ Lag2, data = train, family = binomial)
summary(r10_d)
```

```
##
## Call:
## glm(formula = Direction ~ Lag2, family = binomial, data = 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
```

```
# prediction
test <- Weekly[-(1:nrow(train)),]
prob_d <- predict(r10_d, test, type = "response")
pred_d <- rep("Down", length(prob_d))
pred_d[prob_d > 0.5] <- "Up"
table(pred_d, test$Direction)
```

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

```
mean(pred_d == test$Direction)
```

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

We can conclude that this new logistic regression correctly predicted the movement of the market 62.5% of the time and 37.5% is the training error rate. We could also say that for weeks when the market goes up, the model is right 91.80328% of the time (56/(56+5)). For weeks when the market goes down, the model is right only 20.93023% of the time (9/(9+34)).

(e)

```r
library(MASS)
fit.lda=lda(Direction~Lag2 ,data=train)
fit.lda
```

```
## Call:
## lda(Direction ~ Lag2, data = train)
##
## 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
pred.lda <- predict(fit.lda, test)
table(pred.lda$class, test$Direction)
```

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

```r
mean(pred.lda$class==test$Direction)
```

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

In this case, we may conclude that the percentage of correct predictions on the test data is 62.5%. In other words 37.5% is the test error rate. We could also say that for weeks when the market goes up, the model is right 91.80328% of the time. For weeks when the market goes down, the model is right only 20.93023% of the time. These results are almost same as those obtained with the logistic regression model in (d).

(f)

```r
fit.qda <- qda(Direction ~ Lag2, data = train)
fit.qda
```

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

```r
pred.qda <- predict(fit.qda, test)
table(pred.qda$class, test$Direction)
```

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

```r
mean(pred.qda$class==test$Direction)
```

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

In this case, we may conclude that the percentage of correct predictions on the test data is 58.65385%. In other words 41.34615% is the test error rate. We could also say that for weeks when the market goes up, the model is right 100% of the time. For weeks when the market goes down, the model is right only 0% of the time. We may note, that QDA achieves a correctness of 58.6538462% even though the model chooses "Up" the whole time !

(g)

```r
library(class)
train.X <- as.matrix(train$Lag2)
test.X <- as.matrix(test$Lag2)
train.Direction <- train$Direction

set.seed(1)
pred.knn <- knn(train.X, test.X, train.Direction, k = 1)
table(pred.knn, test$Direction)
```

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

```r
mean(pred.knn==test$Direction)
```

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

In this case, we may conclude that the percentage of correct predictions on the test data is 50%. Therefore, the test error rate is 50%. We could also say that for weeks when the market goes up, the model is right 50.81967% of the time. For weeks when the market goes down, the model is right only 48.83721% of the time.

(h)

If we compare the test error rates, we can see that LDA have the minimum error rates, followed by QDA and KNN.

(i)

```r
# Logistic regression with interaction Lag2:Lag1
fit.glm <- glm(Direction ~ Lag2:Lag1, data = train, family = binomial)

probs <- predict(fit.glm, test, type = "response")
```

```r
pred.glm <- rep("Down", length(probs))
pred.glm[probs > 0.5] = "Up"

table(pred.glm, test$Direction)
```

```
##
## pred.glm Down Up
##     Down    1  1
##     Up     42 60
```

```r
mean(pred.glm == test$Direction)
```

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

```r
# LDA with interaction Lag2:Lag1
fit.lda2 <- lda(Direction ~ Lag2:Lag1, data = train)
pred.lda2 <- predict(fit.lda2, test)
mean(pred.lda2$class == test$Direction)
```

```
## [1] 0.5769231
```

```r
# QDA with log transformation for Lag2
fit.qda2 <- qda(Direction ~ log(abs(Lag2)), data = train)
pred.qda2 <- predict(fit.qda2, test)
table(pred.qda2$class, test$Direction)
```

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

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

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

```r
# KNN k=100
pred.knn2 <- knn(train.X, test.X, train.Direction, k = 100)
table(pred.knn2, test$Direction)
```

```
##
## pred.knn2 Down Up
##      Down    9 12
##      Up     34 49
```

```r
mean(pred.knn2==test$Direction)
```

```
## [1] 0.5576923
```

In all these combinations, the logistic regression with interaction and LDA have the best performance in terms of test error rates.

4.11

(a)

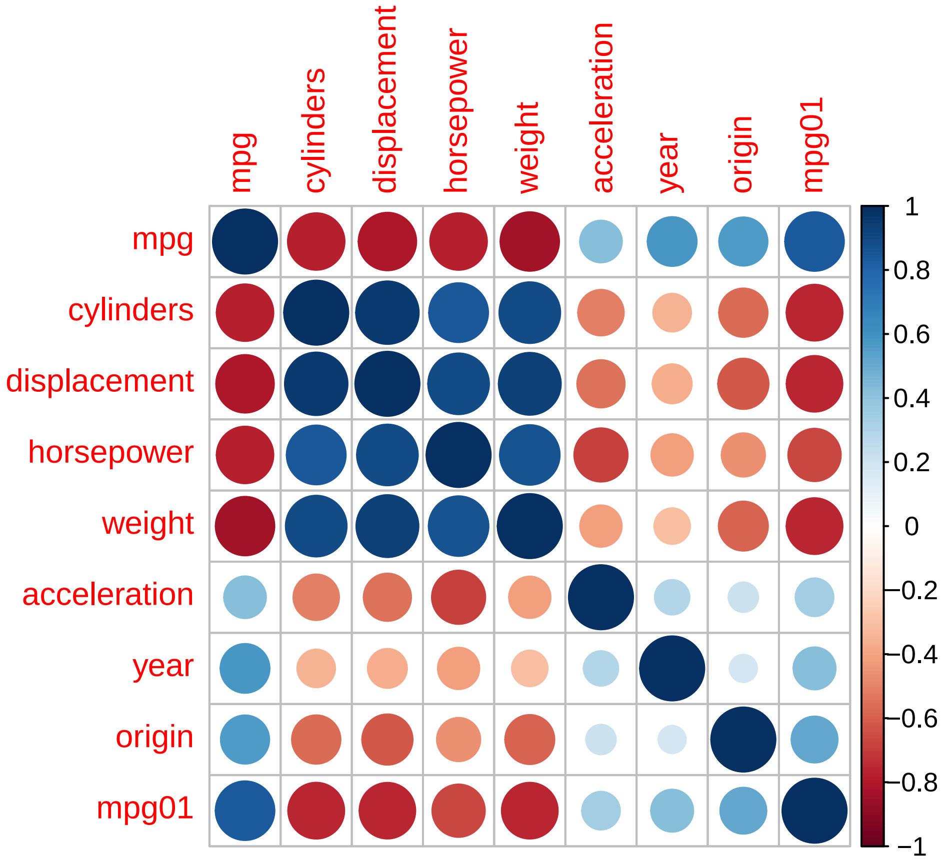
```r
attach(Auto)
mpg01 <- rep(0, length(mpg))
mpg01[mpg > median(mpg)] <- 1
Auto <- data.frame(Auto, mpg01)
```

(b)

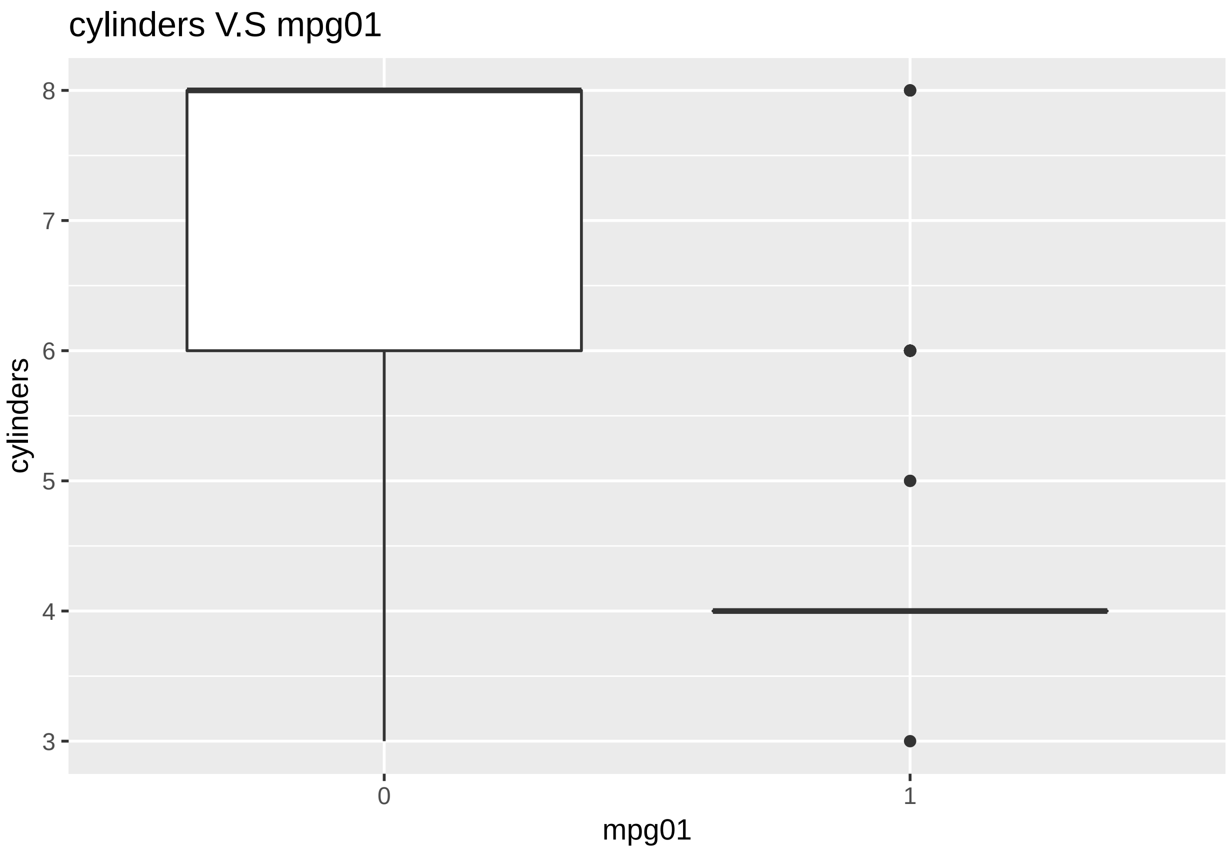
```r
# The variable in column 9 is not numeric
M=cor(Auto[,-9])
M
```

```
##                     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
```

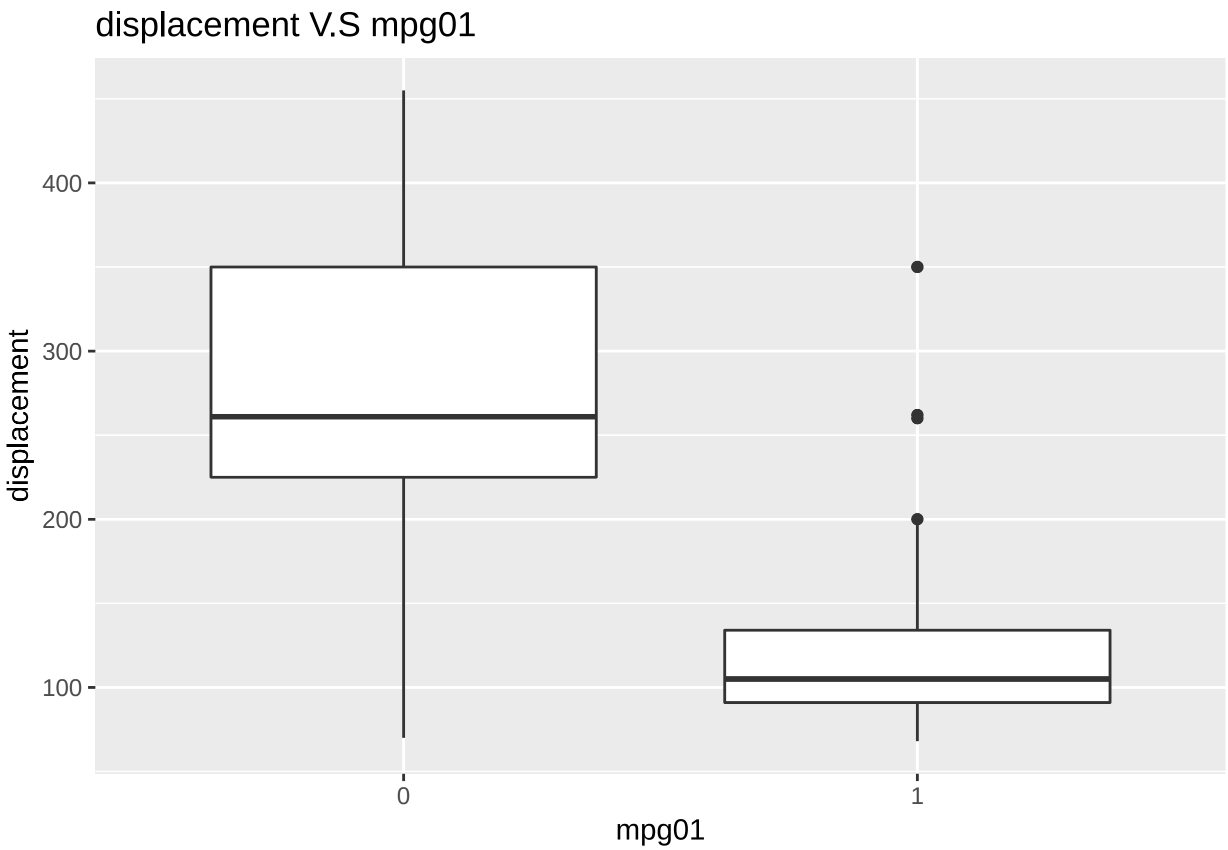
```r
library(corrplot)
corrplot(M, method = "circle")
```



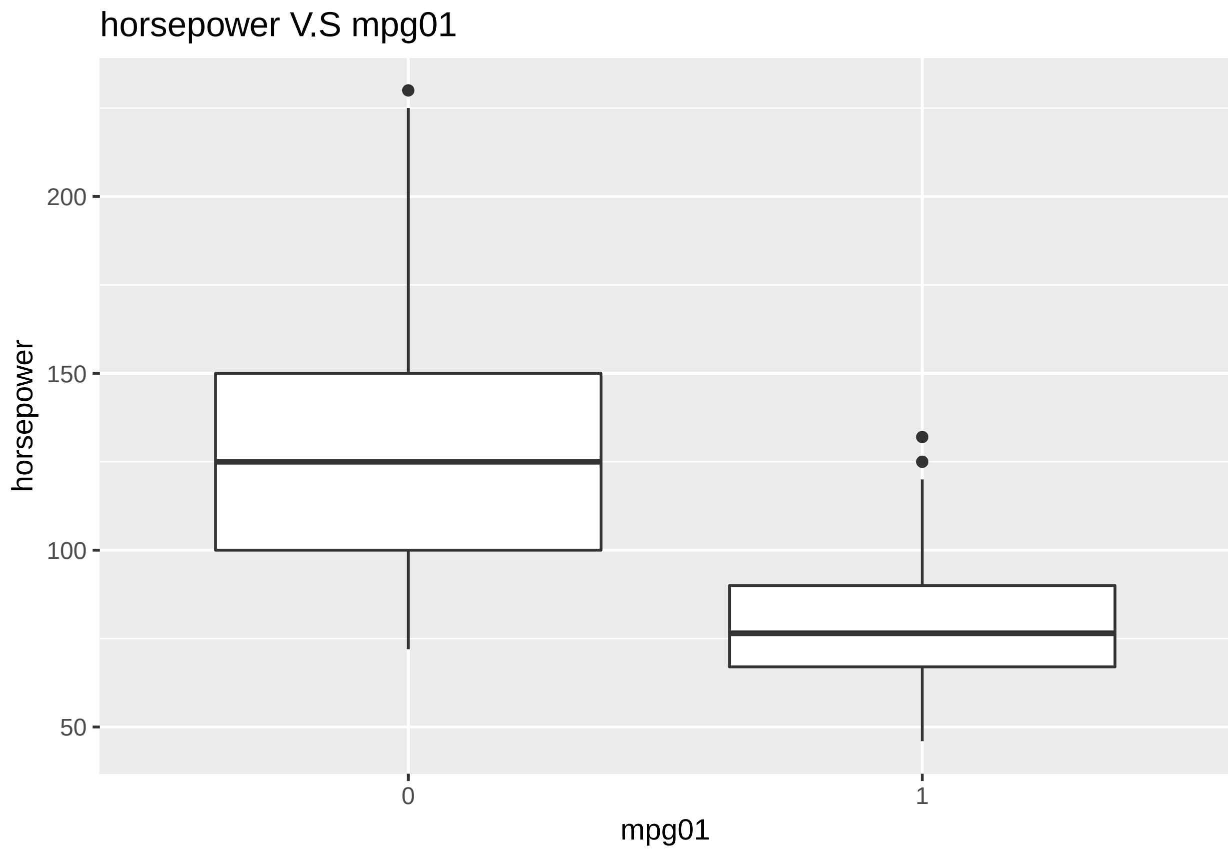

```
Auto$mpg01 <- as.factor(Auto$mpg01)
ggplot(Auto, aes(x=mpg01, y=cylinders)) +
  geom_boxplot() +
  labs(title = "cylinders V.S mpg01")
```



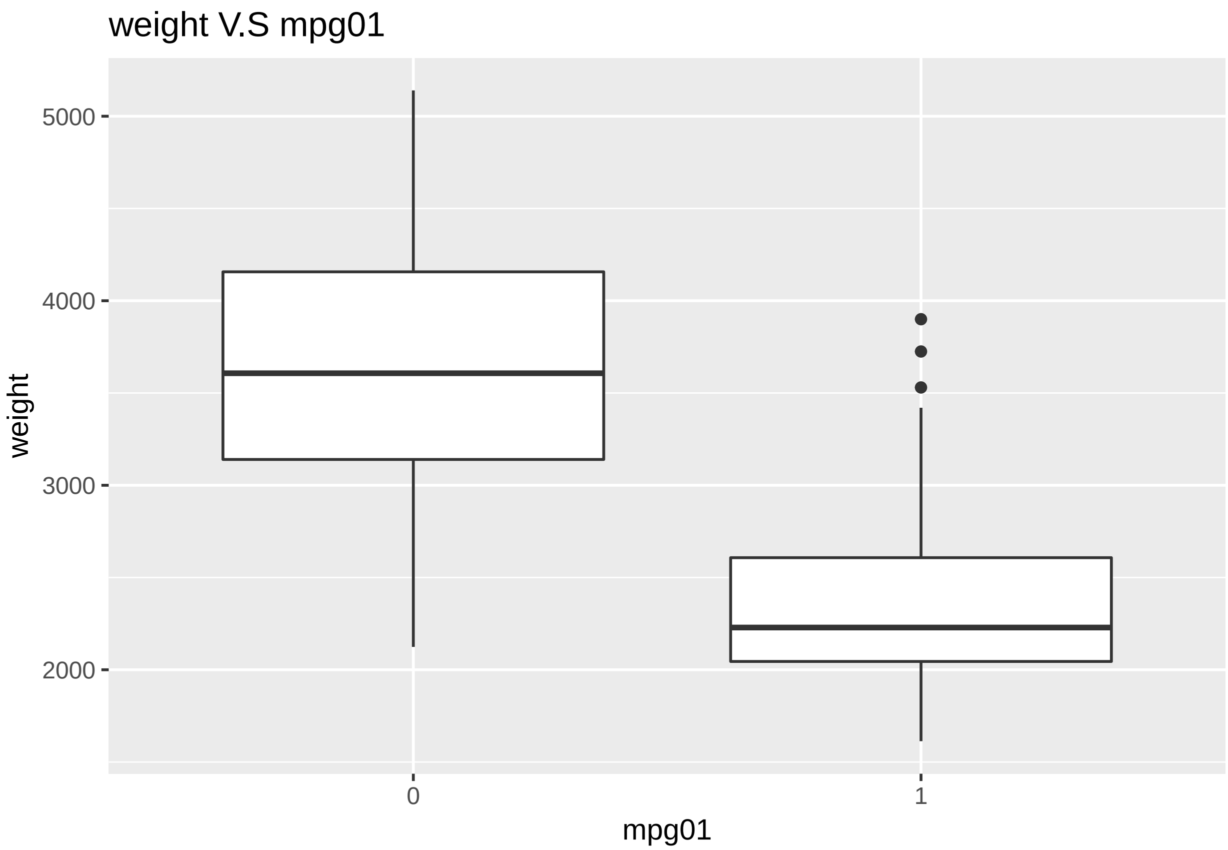

```r
ggplot(Auto, aes(x=mpg01, y=displacement)) +
  geom_boxplot() +
  labs(title = "displacement V.S mpg01")
```



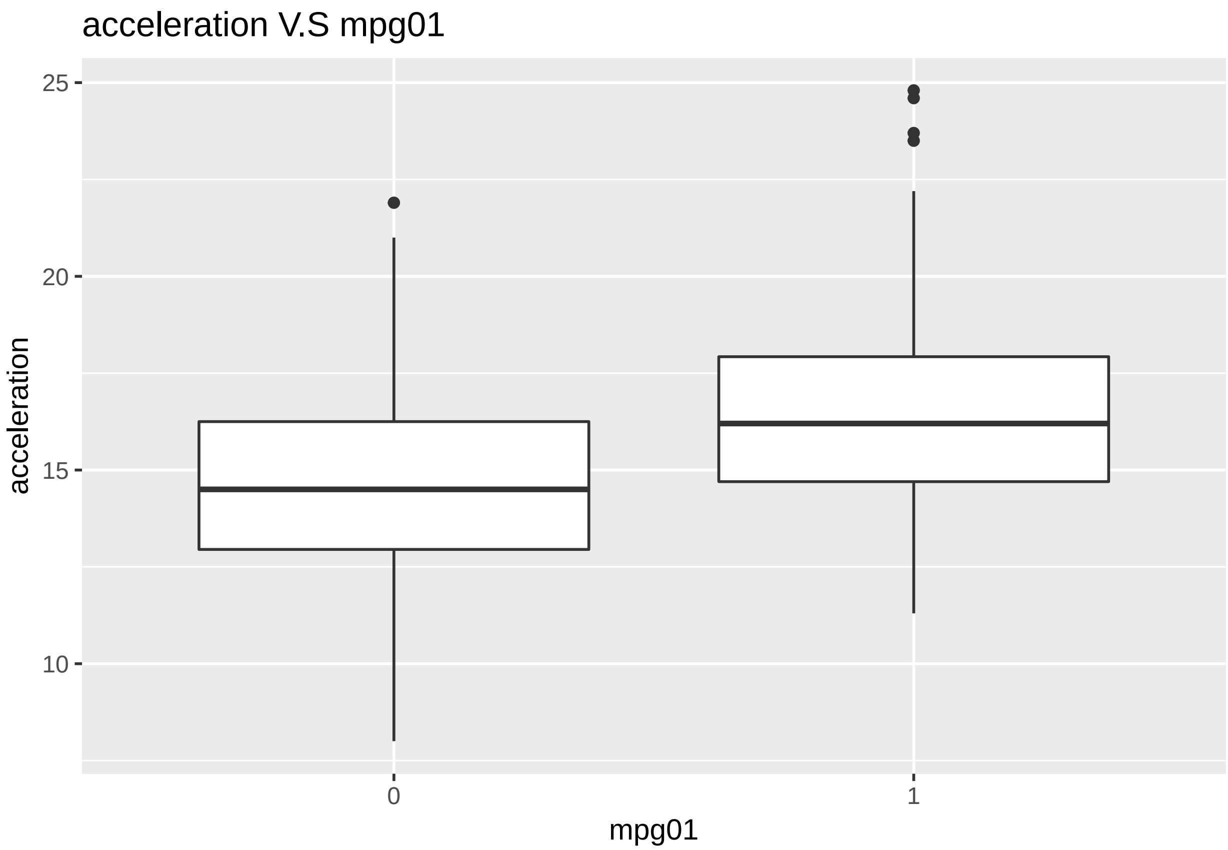

```r
ggplot(Auto, aes(x=mpg01, y=horsepower)) +
  geom_boxplot() +
  labs(title = "horsepower V.S mpg01")
```



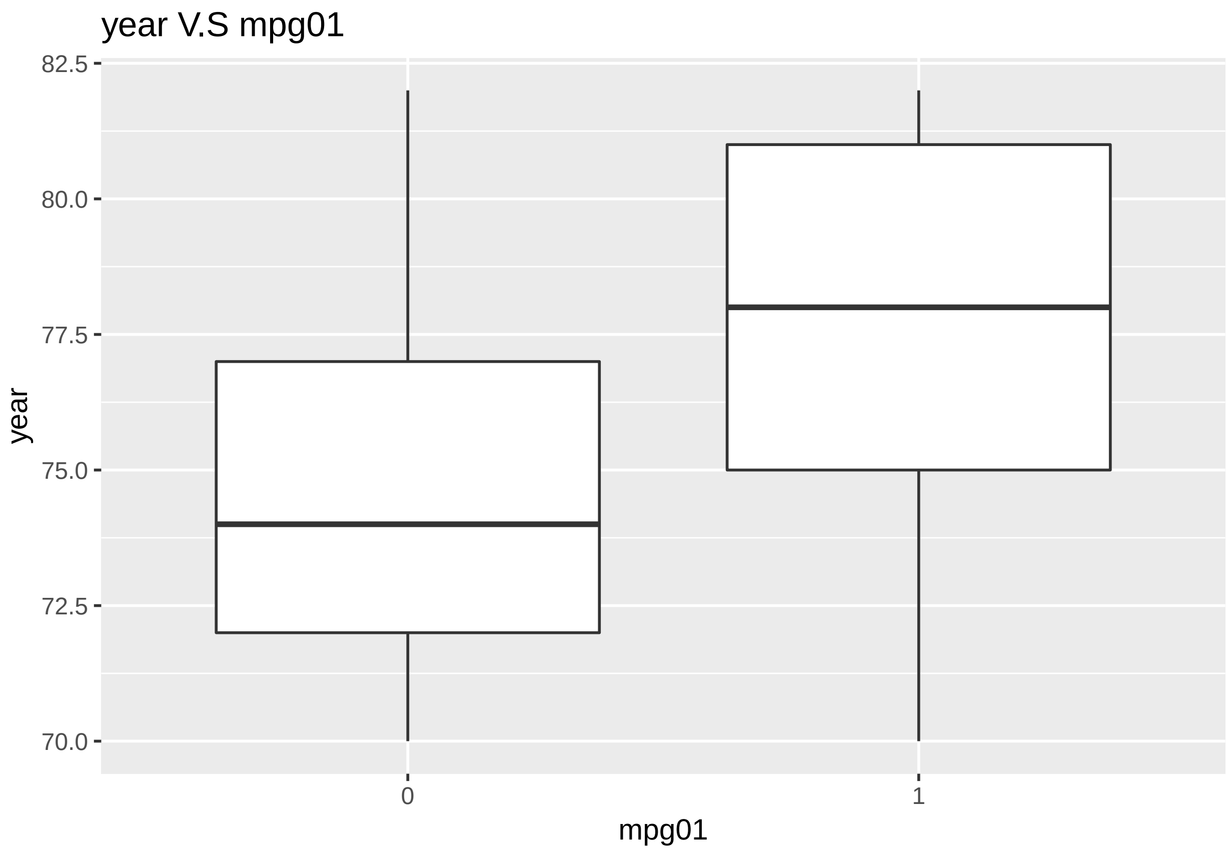

```
ggplot(Auto, aes(x=mpg01, y=weight)) +
  geom_boxplot() +
  labs(title = "weight V.S mpg01")
```




```
ggplot(Auto, aes(x=mpg01, y=acceleration)) +
  geom_boxplot() +
  labs(title = "acceleration V.S mpg01")
```




```r
ggplot(Auto, aes(x=mpg01, y=year)) +
  geom_boxplot() +
  labs(title = "year V.S mpg01")
```



We may conclude that there exists some association between "mpg01" and "cylinders", "weight", "displacement" and "horsepower".

(c)

```
train <- (year %% 2 == 0)
Auto.train <- Auto[train, ]
Auto.test <- Auto[!train, ]
mpg01.test <- mpg01[!train]
```

(d)

```
fit.lda <- lda(mpg01 ~ cylinders + weight + displacement + horsepower, data = Auto.train)
fit.lda
```

```
## Call:
## lda(mpg01 ~ cylinders + weight + displacement + horsepower, data = Auto.train)
##
## Prior probabilities of groups:
##         0         1
## 0.4571429 0.5428571
##
## Group means:
##   cylinders   weight displacement horsepower
```

```
## 0  6.812500 3604.823     271.7396  133.14583
## 1  4.070175 2314.763     111.6623   77.92105
##
## Coefficients of linear discriminants:
##                        LD1
## cylinders    -0.6741402638
## weight       -0.0011465750
## displacement  0.0004481325
## horsepower    0.0059035377
```

```
pred.lda <- predict(fit.lda, Auto.test)
table(pred.lda$class, mpg01.test)
```

```
##    mpg01.test
##      0  1
##   0 86  9
##   1 14 73
```

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

```
## [1] 0.1263736
```

The test error rate of LDA is 12.63736%.

(e)

```
fit.qda <- qda(mpg01 ~ cylinders + weight + displacement + horsepower, data = Auto.train)
fit.qda
```

```
## Call:
## qda(mpg01 ~ cylinders + weight + displacement + horsepower, data = Auto.train)
##
## Prior probabilities of groups:
##         0         1
## 0.4571429 0.5428571
##
## Group means:
##   cylinders   weight displacement horsepower
## 0  6.812500 3604.823     271.7396  133.14583
## 1  4.070175 2314.763     111.6623   77.92105
```

```
pred.qda <- predict(fit.qda, Auto.test)
table(pred.qda$class, mpg01.test)
```

```
##    mpg01.test
##      0  1
##   0 89 13
##   1 11 69
```

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

```
## [1] 0.1318681
```

The test error rate of QDA is 13.18681%.

(f)

```r
fit.glm <- glm(mpg01 ~ cylinders + weight + displacement + horsepower, data = Auto.train,
               family = binomial)
summary(fit.glm)
```

```
##
## Call:
## glm(formula = mpg01 ~ cylinders + weight + displacement + horsepower,
##     family = binomial, data = Auto.train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.48027  -0.03413   0.10583   0.29634   2.57584
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  17.658730   3.409012   5.180 2.22e-07 ***
## cylinders    -1.028032   0.653607  -1.573   0.1158
## weight       -0.002922   0.001137  -2.569   0.0102 *
## displacement  0.002462   0.015030   0.164   0.8699
## horsepower   -0.050611   0.025209  -2.008   0.0447 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 289.58  on 209  degrees of freedom
## Residual deviance:  83.24  on 205  degrees of freedom
## AIC: 93.24
##
## Number of Fisher Scoring iterations: 7
```

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

```
##         mpg01.test
## pred.glm  0  1
##        0 89 11
##        1 11 71
```

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

```
## [1] 0.1208791
```

The test error rate of logistic regression is 12.08791%.

(g)

```r
train.X <- cbind(cylinders, weight, displacement, horsepower)[train, ]
test.X <- cbind(cylinders, weight, displacement, horsepower)[!train, ]
train.mpg01 <- mpg01[train]
```

```r
set.seed(1)
# k=1
pred.knn <- knn(train.X, test.X, train.mpg01, k = 1)
table(pred.knn, mpg01.test)
```

```
##           mpg01.test
## pred.knn   0  1
##          0 83 11
##          1 17 71
```

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

```
## [1] 0.1538462
```

```r
# k=30
pred.knn <- knn(train.X, test.X, train.mpg01, k = 30)
table(pred.knn, mpg01.test)
```

```
##           mpg01.test
## pred.knn   0  1
##          0 83  8
##          1 17 74
```

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

```
## [1] 0.1373626
```

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

```
##           mpg01.test
## pred.knn   0  1
##          0 81  7
##          1 19 75
```

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

```
## [1] 0.1428571
```

We can see when k=30, the error rate is the smallest.

4.12

(a)

```r
Power <- function() {
    2^3
}

Power()
```

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

(b)

```
Power2 <- function(x, a) {
    x^a
}

Power2(3, 8)
```

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

(c)

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

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

```
Power2(8, 17)
```

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

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

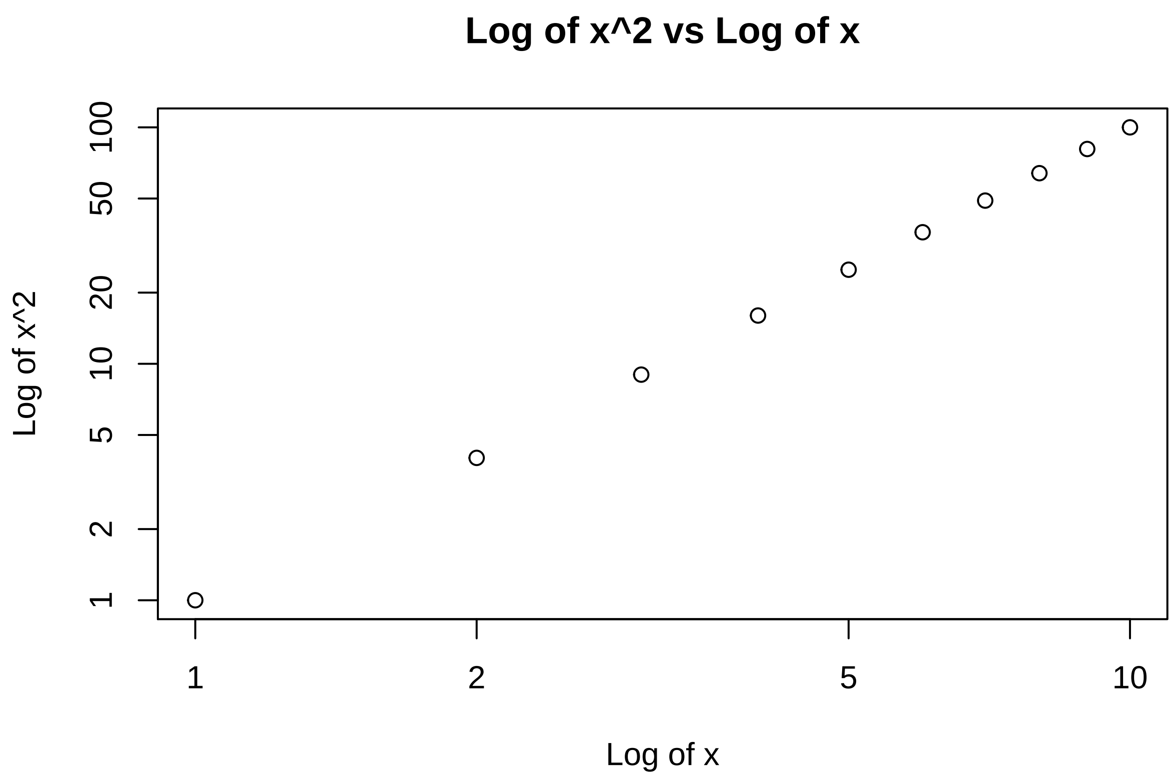
```
## [1] 2248091
```

(d)

```
Power3 <- function(x, a) {
    result <- x^a
    return(result)
}
```

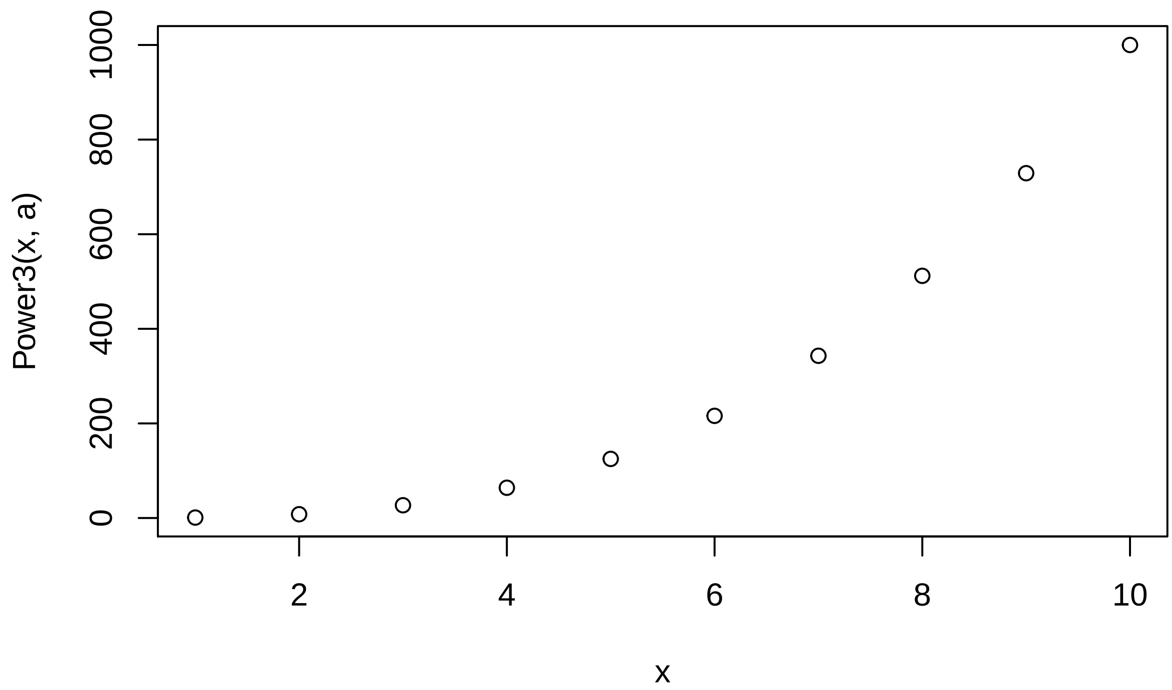
(e)

```
x <- 1:10
plot(x, Power3(x, 2), log = "xy", xlab = "Log of x", ylab = "Log of x^2", main = "Log of x^2 vs Log of :
```



(f)

```
PlotPower <- function(x, a) {
    plot(x, Power3(x, a))
}

PlotPower(1:10, 3)
```



4.13

```
library(MASS)
attach(Boston)
crim01 <- rep(0, length(crim))
crim01[crim > median(crim)] <- 1
Boston <- data.frame(Boston, crim01)

train <- 1:(length(crim) / 2)
test <- (length(crim) / 2 + 1):length(crim)
Boston.train <- Boston[train, ]
Boston.test <- Boston[test, ]
crim01.test <- crim01[test]

# logistic regression
fit.glm <- glm(crim01 ~ . - crim01 - crim - chas - nox,
               data = Boston, family = binomial, subset = train)
# prediction
probs <- predict(fit.glm, Boston.test, type = "response")
pred.glm <- rep(0, length(probs))
pred.glm[probs > 0.5] <- 1
table(pred.glm, crim01.test)
```

```
##          crim01.test
## pred.glm   0   1
##        0  78  28
##        1  12 135
```

```
mean(pred.glm != crim01.test)
```

```
## [1] 0.1581028
```

The error rate of logistic regression is 15.81028%.

```r
# LDA
fit.lda <- lda(crim01 ~ . - crim01 - crim, data = Boston, subset = train)
pred.lda <- predict(fit.lda, Boston.test)
table(pred.lda$class, crim01.test)
```

```
##    crim01.test
##       0   1
##   0  80  24
##   1  10 139
```

```r
mean(pred.lda$class != crim01.test)
```

```
## [1] 0.1343874
```

The error rate of LDA is 13.43874%.

```r
train.X <- cbind(zn, indus, chas, nox, rm, age, dis, rad, tax, ptratio, black, lstat, medv)[train, ]
test.X <- cbind(zn, indus, chas, nox, rm, age, dis, rad, tax, ptratio, black, lstat, medv)[test, ]
train.crim01 <- crim01[train]
# KNN
set.seed(1)
pred.knn <- knn(train.X, test.X, train.crim01, k = 10)
table(pred.knn, crim01.test)
```

```
##         crim01.test
## pred.knn   0   1
##        0  83  21
##        1   7 142
```

```r
mean(pred.knn != crim01.test)
```

```
## [1] 0.1106719
```

The error rate of KNN (k=10) is 11.06719%.