LAB 4

Group 5

2022-09-22

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com (http://rmarkdown.rstudio.com).

#4.7 Exercises; Problem 11

```
library(ISLR)
library(MASS)
names(Auto)
```

```
## [1] "mpg"          "cylinders"    "displacement" "horsepower"   "weight"
## [6] "acceleration" "year"         "origin"       "name"
```

```
dim(Auto)
```

```
## [1] 392   9
```

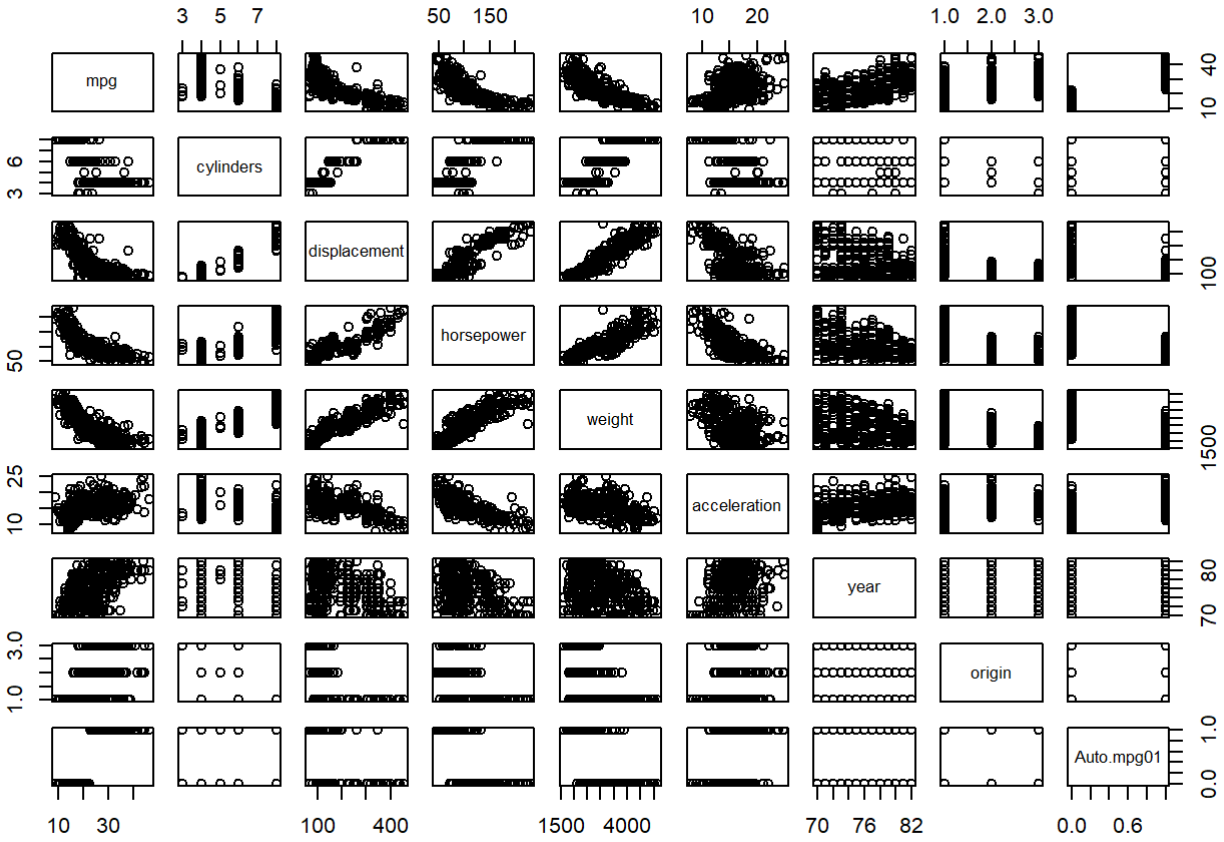
#(a)

```
Auto$mpg01 = 0
for(i in 1:dim(Auto)[1]){
  if (Auto$mpg[i] > median(Auto$mpg)){
    Auto$mpg01[i] = 1
  }else{
    Auto$mpg01[i] = 0
  }
}
data = data.frame(Auto[,1:8], Auto$mpg01)
head(data)
```

```
##   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
##   Auto.mpg01
## 1          0
## 2          0
## 3          0
## 4          0
## 5          0
## 6          0
```

#(b)

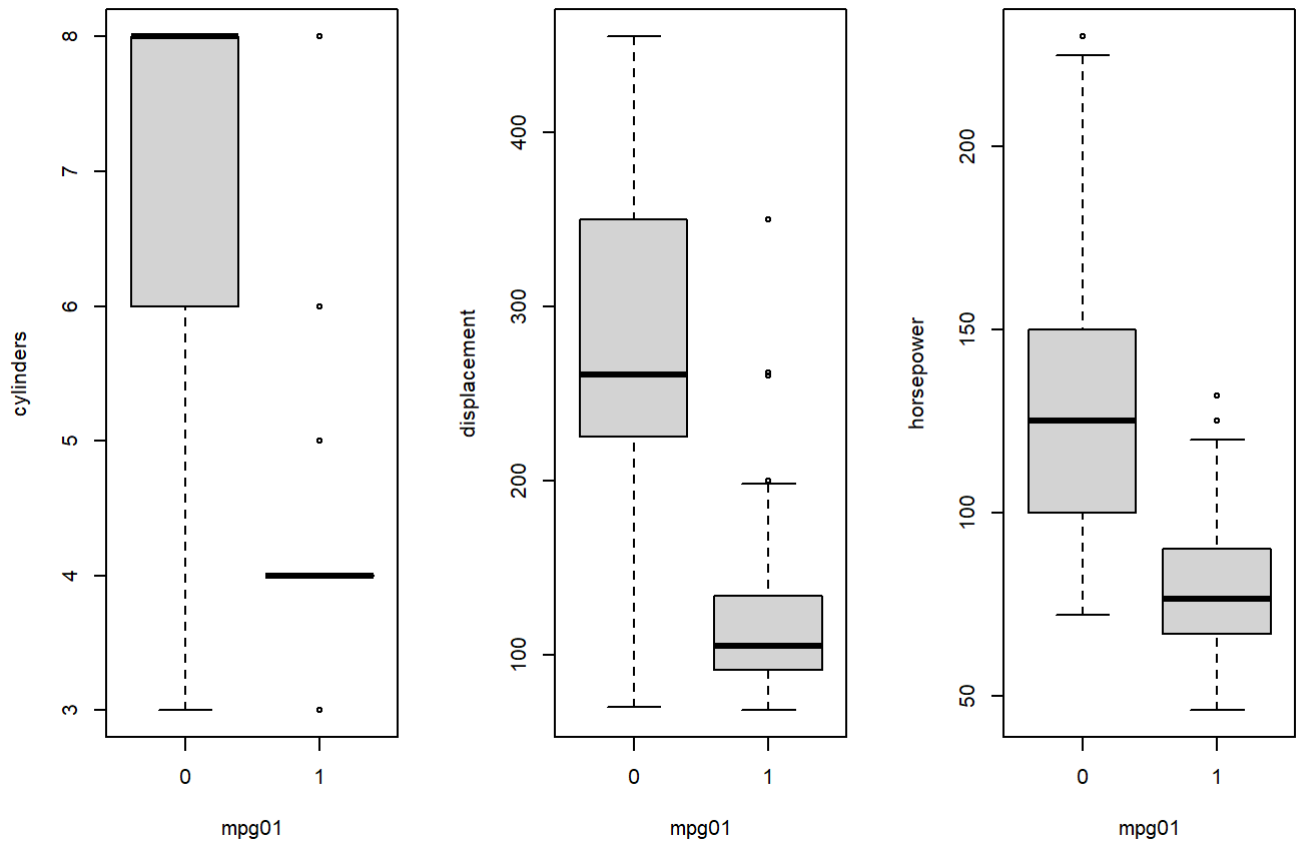
```
pairs(data)
```



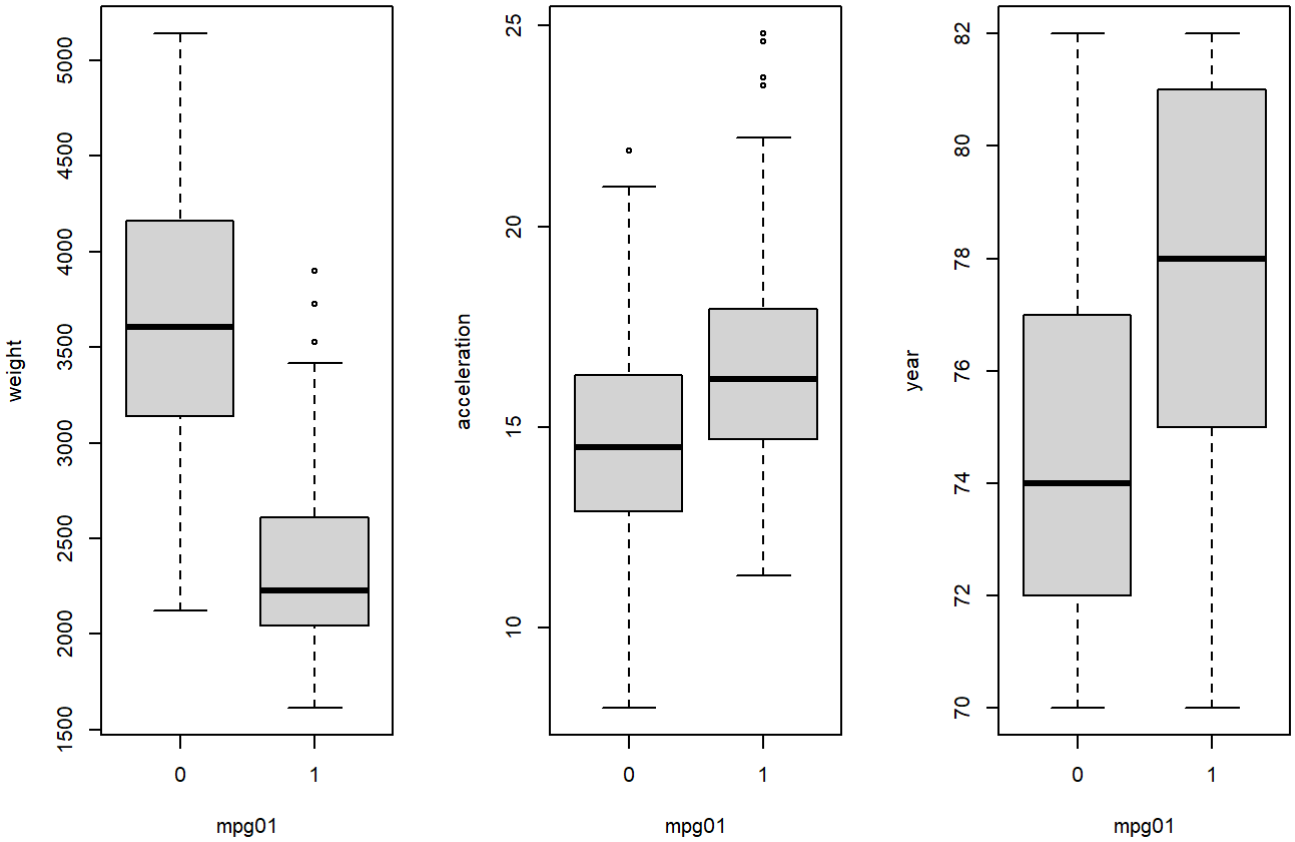

```
cor(data)
```

```
##                     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
## Auto.mpg01    0.8369392 -0.7591939   -0.7534766 -0.6670526 -0.7577566
##              acceleration       year     origin Auto.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
## Auto.mpg01      0.3468215  0.4299042  0.5136984  1.0000000
```

```
par(mfrow=c(1,3))
boxplot(cylinders ~ mpg01, data = Auto)
boxplot(displacement ~ mpg01, data = Auto)
boxplot(horsepower ~ mpg01, data = Auto)
```



```
boxplot(weight ~ mpg01, data = Auto)
boxplot(acceleration ~ mpg01, data = Auto)
boxplot(year ~ mpg01, data = Auto)
```




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

```
We can assume that displacement, horsepower and weight seem most likely to be useful in predict
ing mpg01.
```

#(c)

```
set.seed(8080)
seed = sample(nrow(data), nrow(data)*2/3, replace = FALSE)
train = data[seed,]
test = data[-seed,]
fix(train)
realmpg = data$Auto.mpg01[-seed]
```

#(d)

```
lda1 = lda(Auto.mpg01 ~ displacement+horsepower+weight, data=train)
lda1
```

```
## Call:
## lda(Auto.mpg01 ~ displacement + horsepower + weight, data = train)
##
## Prior probabilities of groups:
##         0         1
## 0.4980843 0.5019157
##
## Group means:
##   displacement horsepower   weight
## 0     276.9846  129.90000 3639.569
## 1     114.9962   79.16794 2325.405
##
## Coefficients of linear discriminants:
##                         LD1
## displacement -0.0105449944
## horsepower    0.0092039998
## weight       -0.0009427213
```

```
lda_pred = predict(lda1,test)
pred1 = lda_pred$class
table(pred1, realmpg)
```

```
##      realmpg
## pred1  0  1
##     0 54  2
##     1 12 63
```

```
mean(pred1 != realmpg)
```

```
## [1] 0.1068702
```

```
We can see the test error is 10.6%
```

#(f)

```
logistic = glm(Auto.mpg01 ~ displacement + horsepower + weight, data = train, family = binomia
l)
logistic
```

```
##
## Call:  glm(formula = Auto.mpg01 ~ displacement + horsepower + weight,
##     family = binomial, data = train)
##
## Coefficients:
##  (Intercept)  displacement    horsepower        weight
##     10.85292      -0.01468      -0.02962      -0.00195
##
## Degrees of Freedom: 260 Total (i.e. Null);  257 Residual
## Null Deviance:       361.8
## Residual Deviance: 137.9     AIC: 145.9
```

```
log_pred = predict(logistic, test, type="response")
pred2 = rep(0, nrow(test))
pred2[log_pred > 0.5] = 1
table(pred2, realmpg)
```

```
##      realmpg
## pred2  0  1
##     0 56  4
##     1 10 61
```

```
mean(pred2 != realmpg)
```

```
## [1] 0.1068702
```

```
The test error is 10.6% as well.
```

#(g)

```
library(class)
train.x = cbind(train$displacement, train$horsepower, train$weight)
test.x = cbind(test$displacement, test$horsepower, test$weight)
train.mpg01 = train$Auto.mpg01
test.mpg01 = test$Auto.mpg01

#knn, k = 1
set.seed(80)
knn.pred = knn(train.x, test.x, train.mpg01, k = 1)
table(knn.pred, test.mpg01)
```

```
##         test.mpg01
## knn.pred  0  1
##        0 54 11
##        1 12 54
```

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

```
## [1] 0.1755725
```

```
#knn, k = 5
set.seed(80)
knn2.pred = knn(train.x, test.x, train.mpg01, k = 5)
table(knn2.pred, test.mpg01)
```

```
##          test.mpg01
## knn2.pred  0  1
##         0 56  8
##         1 10 57
```

```
mean(knn2.pred != test.mpg01)
```

```
## [1] 0.1374046
```

```
#knn, k = 8
set.seed(80)
knn3.pred = knn(train.x, test.x, train.mpg01, k = 5)
table(knn3.pred, test.mpg01)
```

```
##          test.mpg01
## knn3.pred  0  1
##         0 56  8
##         1 10 57
```

```
mean(knn3.pred != test.mpg01)
```

```
## [1] 0.1374046
```

```
We can see the test error are 17.5%, 13.7% and 13.7%, so knn model with k = 5 or k = 8 seems be
tter.
```

#4.7 Exercises; Problem 12 (a)

```
power = function(x, a){
  print(x^a)
}
power(2,3)
```

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

#(b)

```
power2 = function(x, a){
  print(x^a)
}
power2(2,3)
```

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

```
power2(3,8)
```

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

c.

```
power2(10,3)
```

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

```
power2(8,17)
```

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

```
power2(131,3)
```

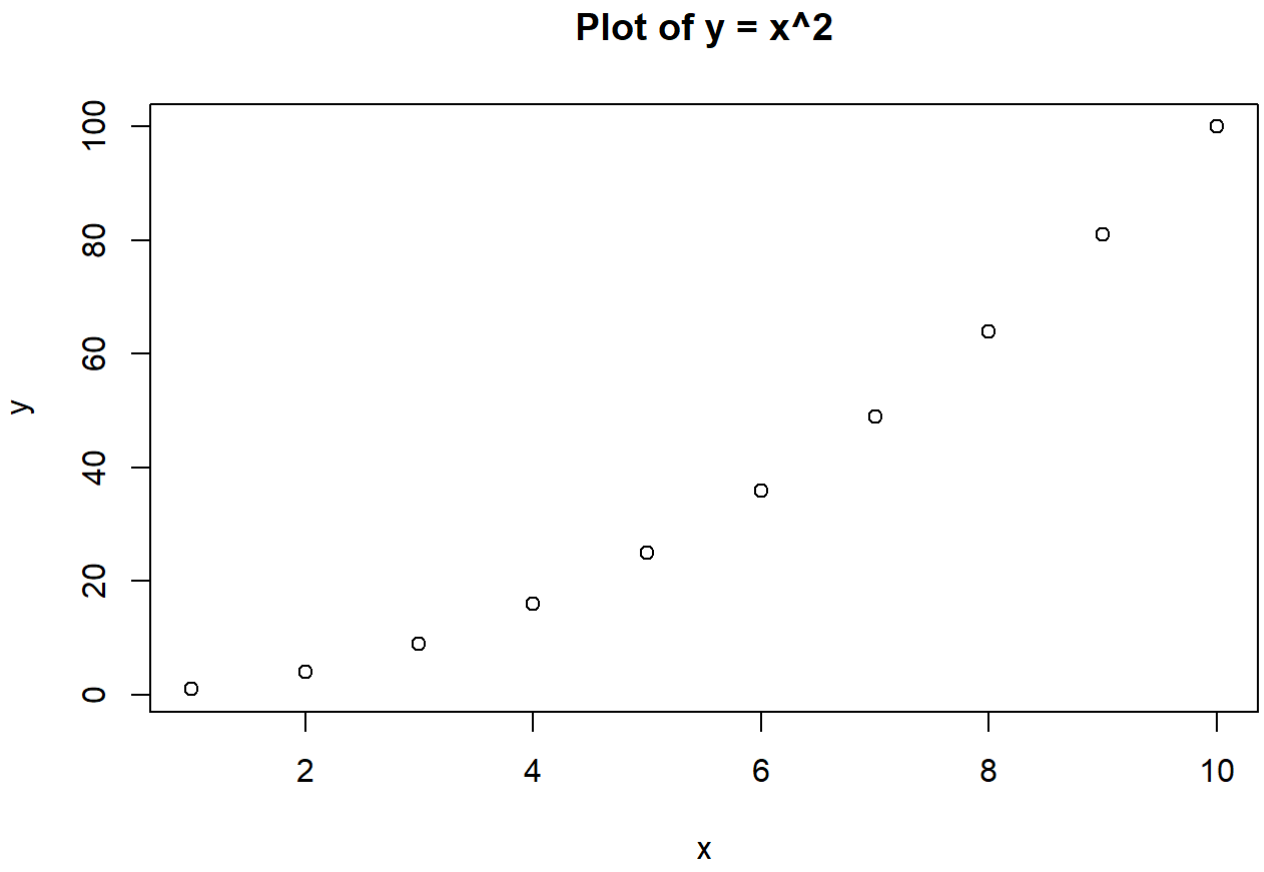
```
## [1] 2248091
```

#(d)

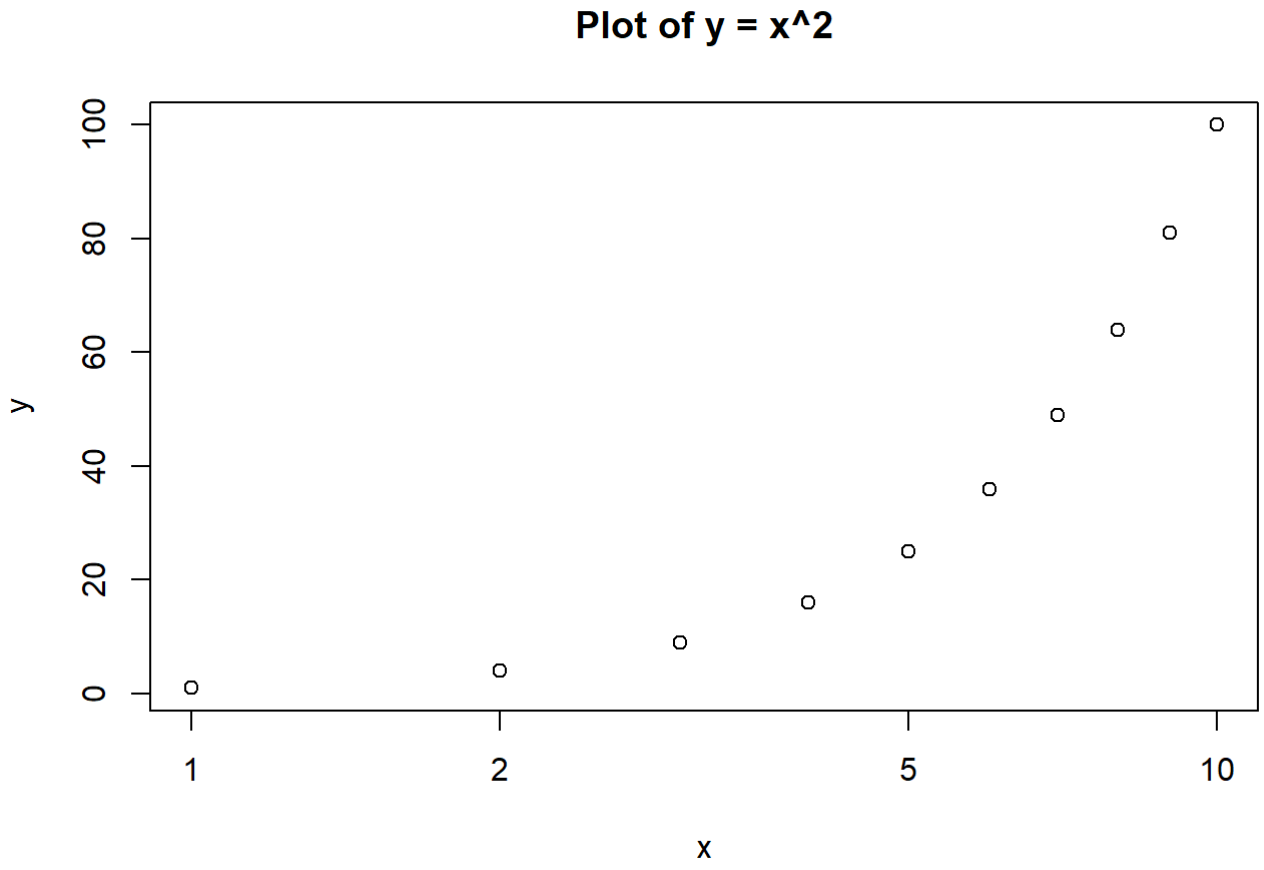
```
Power3 = function(x,a){
result = x^a
return(result)
}
```

#(e)

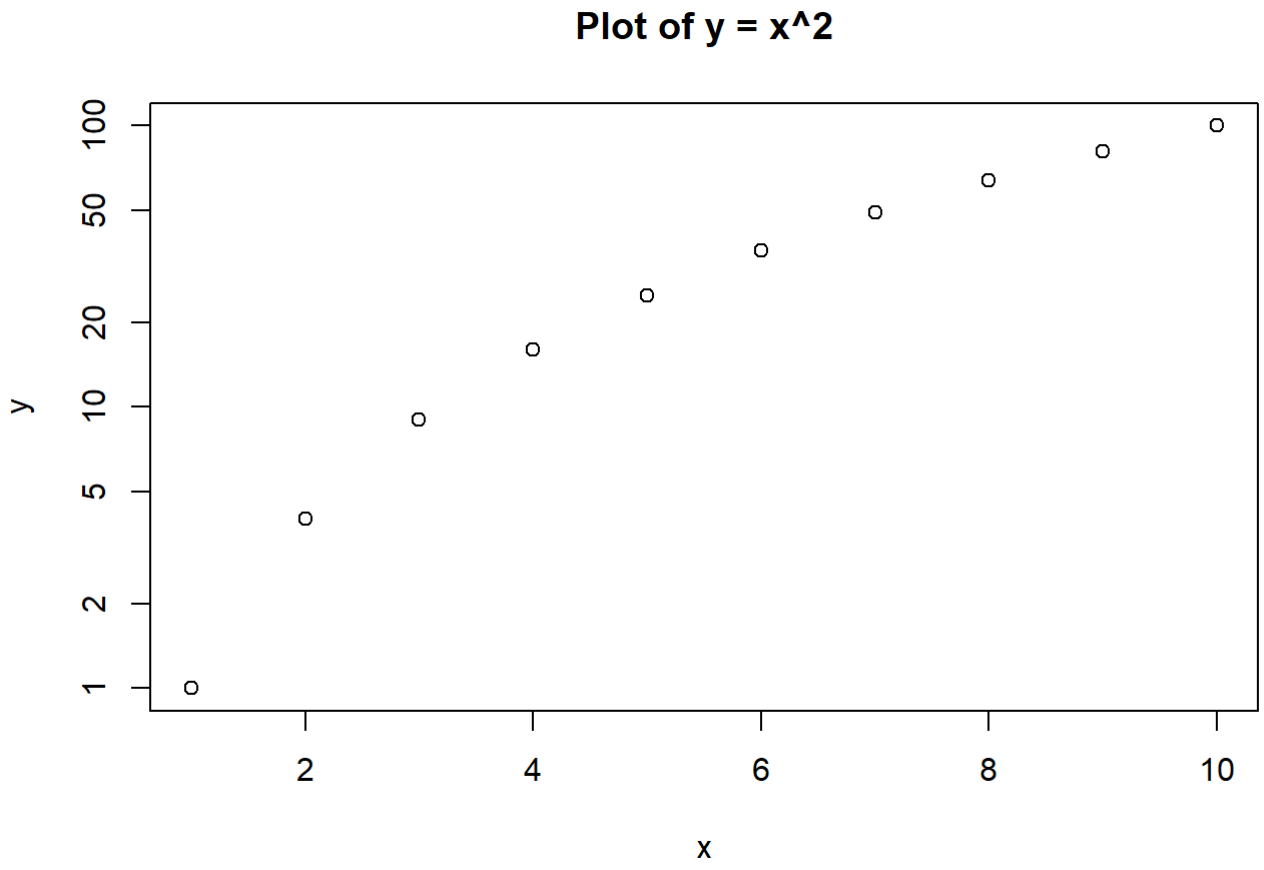
```
x = 1:10
y = Power3(x,2)
plot(x,y, main="Plot of y = x^2")
```



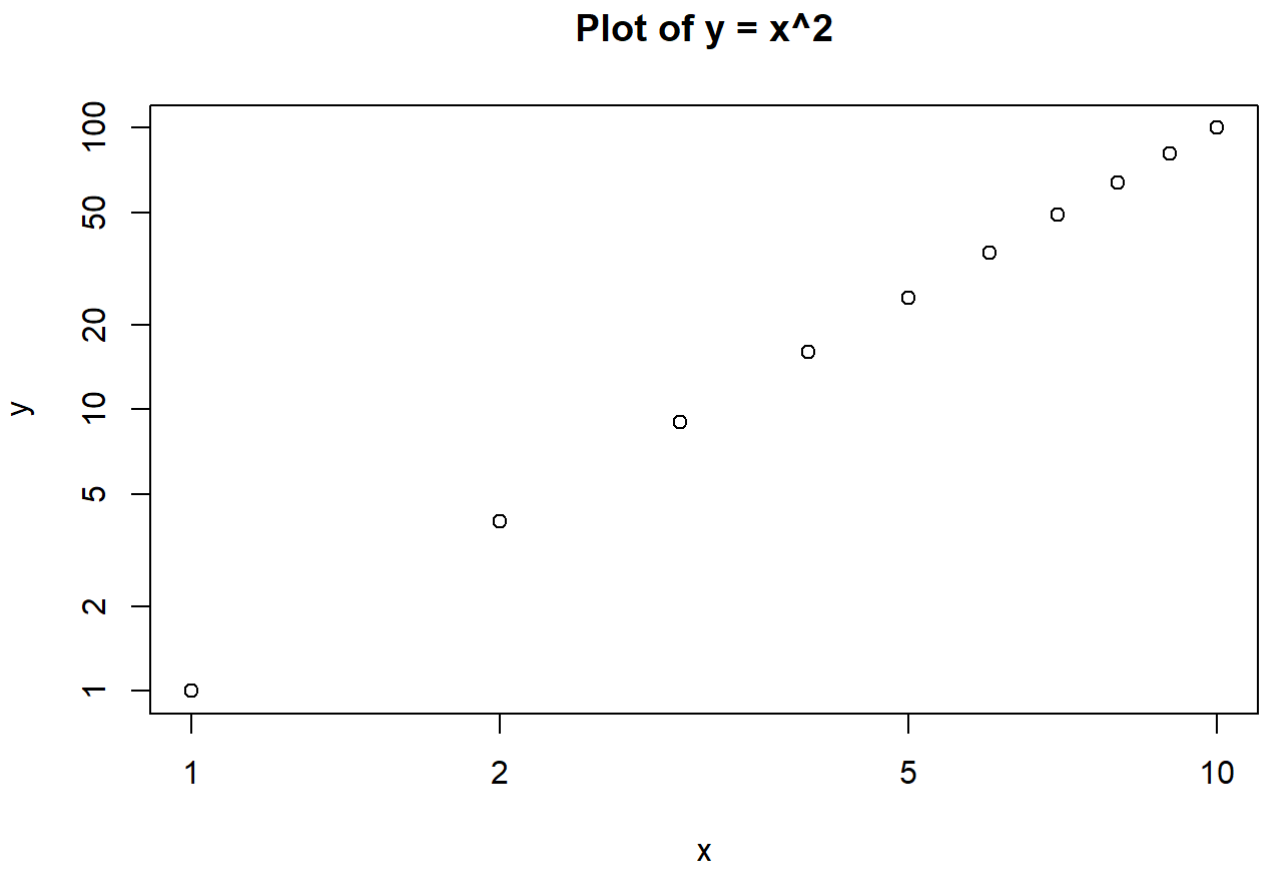

```
plot(x,y, log = 'x', main="Plot of y = x^2")
```




```
plot(x,y, log = 'y', main="Plot of y = x^2")
```

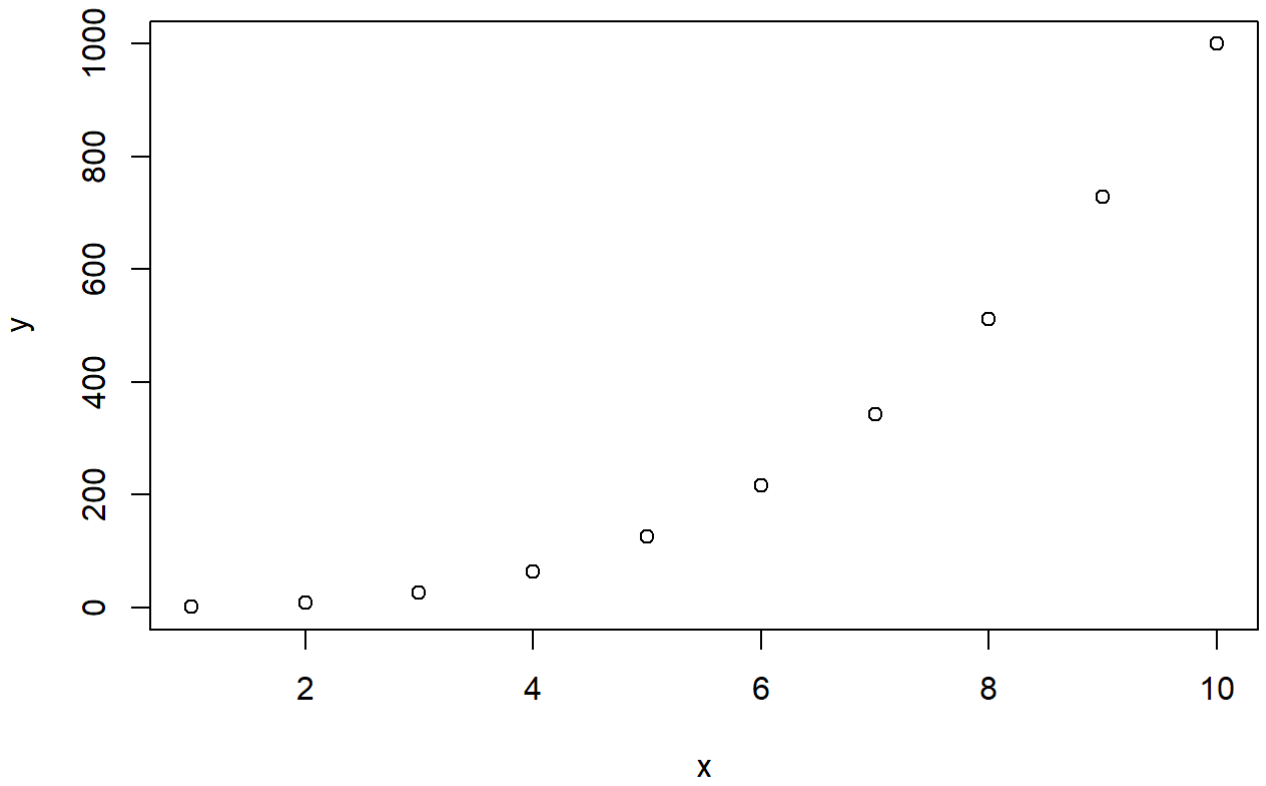


```
plot(x,y, log = 'xy', main="Plot of y = x^2")
```



#(f)

```
PlotPower = function(a,b){
x = a
y = a^b
plot(x, y)
}
PlotPower(1:10,3)
```


y
0
200
400
600
800
1000
2
4
6
8
10
x