HW4

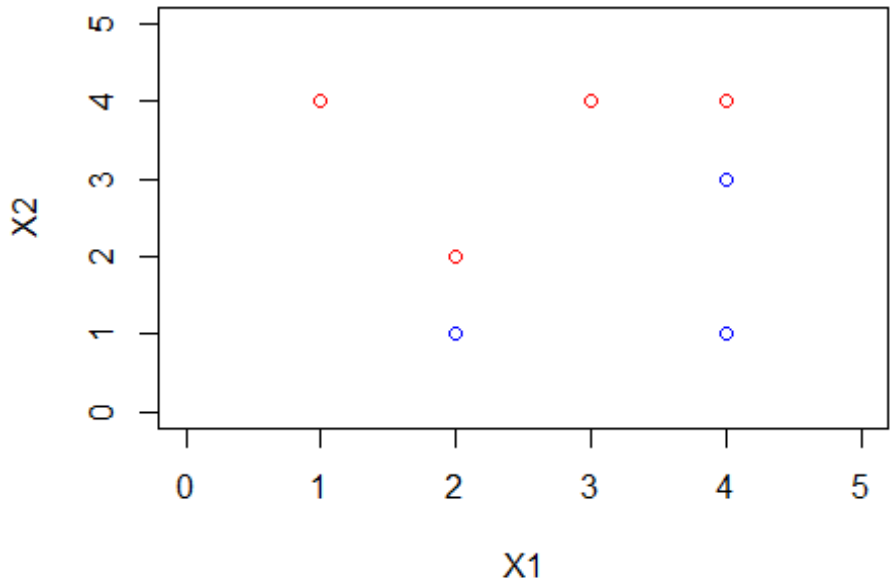
Renuka Ramachandran

20 November 2017

Question 1 - Ex 3 ch 9

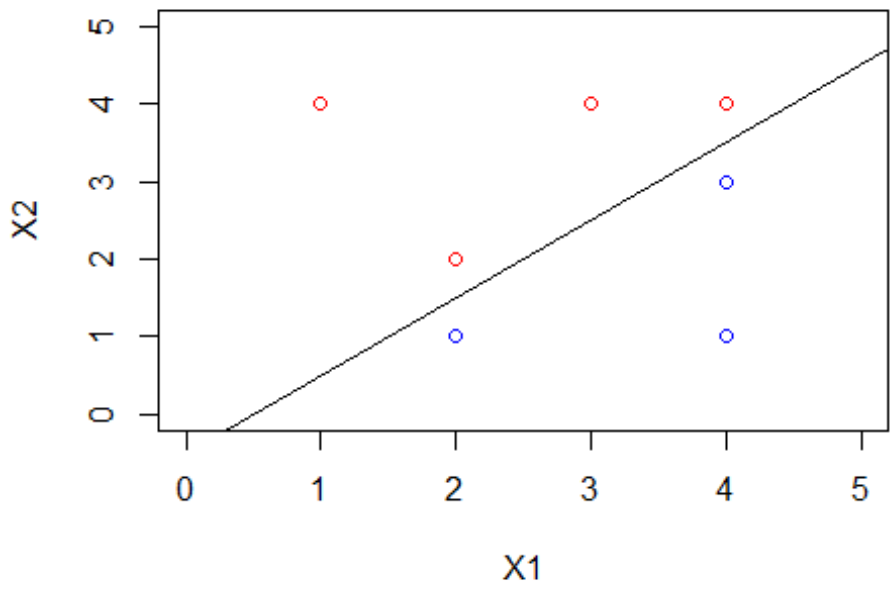
A) We are given n = 7 observations in p = 2 dimensions. For each observation, there is an associated class label. Sketch the observations.

```
X1 <- c(3, 2, 4, 1, 2, 4, 4)
X2 <- c(4, 2, 4, 4, 1, 3, 1)
colors <- c("red", "red", "red", "red", "blue", "blue", "blue")
plot(X1, X2, col = colors, xlim = c(0, 5), ylim = c(0, 5))
```



B) Sketch the optimal separating hyperplane, and provide the equation for this hyperplane

```
plot(X1, X2, col = colors, xlim = c(0, 5), ylim = c(0, 5))
#hyperplane
abline(-0.5, 1)
```



As seen in the plot, the optimal separating hyperplane will be between the points (2,1) and (2,2), and between the points (4,3) and (4,4).

=>(2,1.5),(4,3.5)

b = (3.5-1.5)/(4-2) = 1 ; a = X2 - X1 = 1.5 - 2 = -0.5

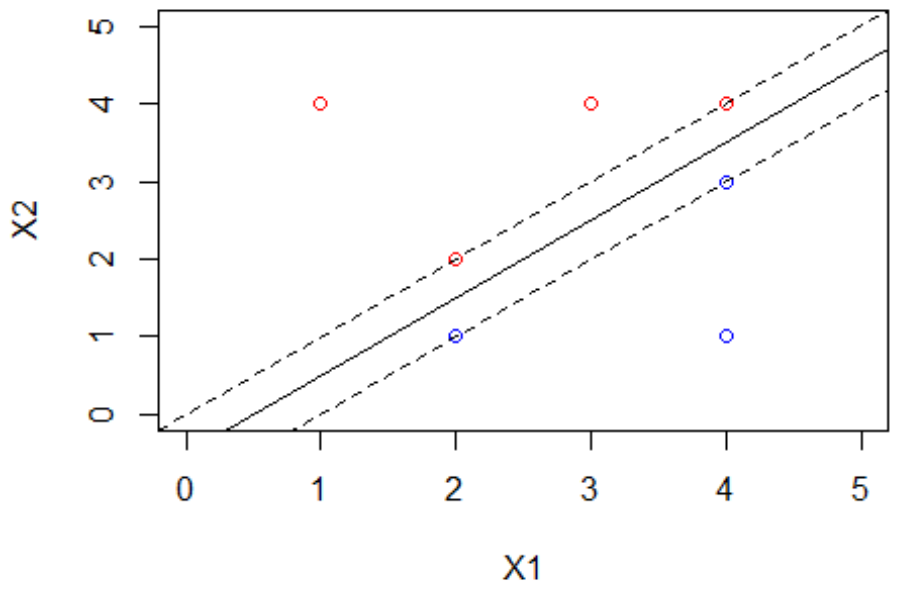
So it is a line that passes through the points (2,1.5) and (4,3.5) which equation is **X1-X2-0.5 = 0**

C) Describe the classification rule for the maximal margin classifier. It should be something along the lines of "Classify to Red if B0 + B1X1 + B2X2 > 0, and classify to Blue otherwise." Provide the values for B0, B1, and B2.

The classification rule is **classify to red if 0.5-X1+X2>0, and classify to blue otherwise.**

D) On your sketch, indicate the margin for the maximal margin hyperplane.

```
plot(X1, X2, col = colors, xlim = c(0, 5), ylim = c(0, 5))
#hyperplan
abline(-0.5, 1)
#maximal margin
abline(-1, 1, lty = 2)
abline(0, 1, lty = 2)
```



The maximal margin width is 1//4

E) Indicate the support vectors for the maximal margin classifier

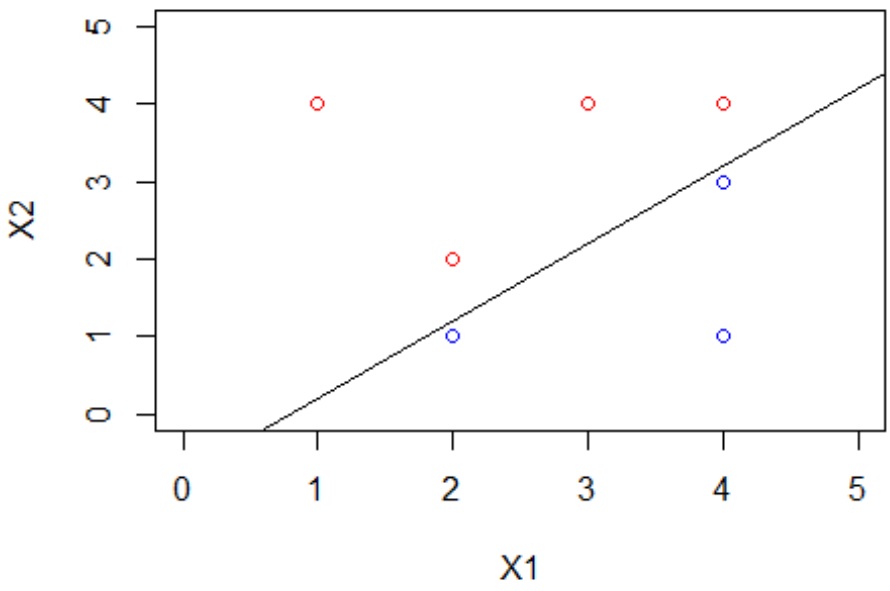
The support vectors are the points (2,1), (2,2), (4,3) and (4,4).

F) Argue that a slight movement of the seventh observation would not affect the maximal margin hyperplane.

From the plot, it can be understood that a slight movement of the observation (4,1) we would not change the maximal margin hyperplane as it is not a support vector (its movement would be outside of the margin).

G) Sketch a hyperplane that is not the optimal separating hyperplane, and provide the equation for this hyperplane.

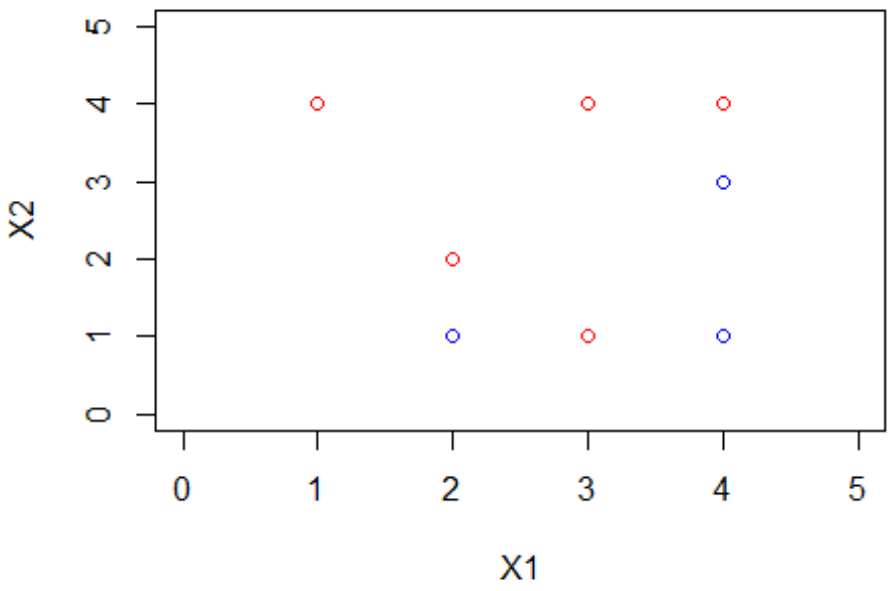
```
plot(X1, X2, col = colors, xlim = c(0, 5), ylim = c(0, 5))
#Sub-optimal hyperplane
abline(-0.8, 1)
```



The equation for this hyperplane is: **0.8-X1+X2 > 0**

H) Draw an additional observation on the plot so that the two classes are no longer separable by a hyperplane.

```
plot(X1, X2, col = colors, xlim = c(0, 5), ylim = c(0, 5))
points(c(3), c(1), col = c("red"))
```



When the red point (3,1) is added to the plot, the two classes are not separable by a hyperplane anymore.

Question 2 - Ex 7 ch 9

A) Create a binary variable that takes on a 1 for cars with gas mileage above the median, and a 0 for cars with gas mileage below the median.

```
library(ISLR)

## Warning: package 'ISLR' was built under R version 3.3.3

gas_median <- median(Auto$mpg)
bin_var <- ifelse(Auto$mpg > gas_median, 1, 0)
Auto$mpglevel <- as.factor(bin_var)
```

B) Fit a support vector classifier to the data with various values of cost, in order to predict whether a car gets high or low gas mileage. Report the cross-validation errors associated with different values of this parameter. Comment on your results.

```
library(e1071)

## Warning: package 'e1071' was built under R version 3.3.3

set.seed(234)
tune_op <- tune(svm, mpglevel ~ ., data = Auto, kernel = "linear", ranges =
list(cost = c(0.01,
```

```
    0.1, 0.5, 1, 3, 5, 10, 100)))
summary(tune_op)

##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##     1
##
## - best performance: 0.01282051
##
## - Detailed performance results:
##    cost      error dispersion
## 1 1e-02 0.07423077 0.05610576
## 2 1e-01 0.04865385 0.04754593
## 3 5e-01 0.02051282 0.03585671
## 4 1e+00 0.01282051 0.02179068
## 5 3e+00 0.01282051 0.02179068
## 6 5e+00 0.01538462 0.02477158
## 7 1e+01 0.01788462 0.02430352
## 8 1e+02 0.03326923 0.02974993
```

The best parameter for which cross-validation error is minimized is for cost=1.

C) Now repeat (b), this time using SVMs with radial and polynomial basis kernels, with different values of gamma and degree and cost. Comment on your results.

```
set.seed(35)
tune_op <- tune(svm, mpglevel ~ ., data = Auto, kernel = "polynomial", ranges
= list(cost =  c(0.01,
    0.1, 0.5, 1, 3, 5, 10), degree = c(2, 3, 4)))
summary(tune_op)

##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost degree
##    10      2
##
## - best performance: 0.4817949
##
## - Detailed performance results:
##     cost degree     error dispersion
## 1   0.01      2 0.5485256 0.03303346
## 2   0.10      2 0.5485256 0.03303346
```

```
## 3   0.50      2 0.5485256 0.03303346
## 4   1.00      2 0.5485256 0.03303346
## 5   3.00      2 0.5485256 0.03303346
## 6   5.00      2 0.5485256 0.03303346
## 7  10.00      2 0.4817949 0.12862190
## 8   0.01      3 0.5485256 0.03303346
## 9   0.10      3 0.5485256 0.03303346
## 10  0.50      3 0.5485256 0.03303346
## 11  1.00      3 0.5485256 0.03303346
## 12  3.00      3 0.5485256 0.03303346
## 13  5.00      3 0.5485256 0.03303346
## 14 10.00      3 0.5485256 0.03303346
## 15  0.01      4 0.5485256 0.03303346
## 16  0.10      4 0.5485256 0.03303346
## 17  0.50      4 0.5485256 0.03303346
## 18  1.00      4 0.5485256 0.03303346
## 19  3.00      4 0.5485256 0.03303346
## 20  5.00      4 0.5485256 0.03303346
## 21 10.00      4 0.5485256 0.03303346
```

The best parameter for which cross-validation error is minimized is cost = 100 and degree = 2.

```
set.seed(60)
tune_op <- tune(svm, mpglevel ~ ., data = Auto, kernel = "radial", ranges =
list(cost = c(0.01,
    0.1, 0.5, 1, 3, 5, 10), gamma = c(0.01, 0.1, 0.5, 1, 3, 5, 10, 100)))
summary(tune_op)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##    10  0.01
##
## - best performance: 0.02557692
##
## - Detailed performance results:
##     cost gamma      error dispersion
## 1   0.01 1e-02 0.58185897 0.04745904
## 2   0.10 1e-02 0.08948718 0.04579255
## 3   0.50 1e-02 0.07423077 0.04281275
## 4   1.00 1e-02 0.07166667 0.04168014
## 5   3.00 1e-02 0.05884615 0.04207264
## 6   5.00 1e-02 0.05115385 0.04018094
## 7  10.00 1e-02 0.02557692 0.02093679
## 8   0.01 1e-01 0.17621795 0.08902462
```

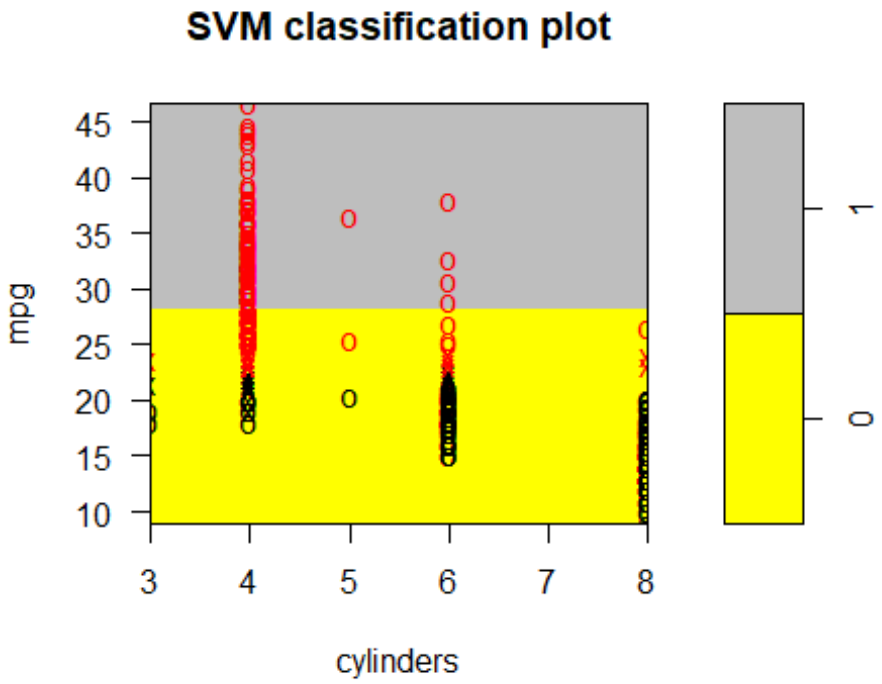
```
## 9   0.10 1e-01 0.07679487 0.04538856
## 10  0.50 1e-01 0.06397436 0.04067783
## 11  1.00 1e-01 0.05371795 0.04442061
## 12  3.00 1e-01 0.03076923 0.03375798
## 13  5.00 1e-01 0.03589744 0.03009712
## 14 10.00 1e-01 0.02814103 0.02246020
## 15  0.01 5e-01 0.58185897 0.04745904
## 16  0.10 5e-01 0.07673077 0.04204795
## 17  0.50 5e-01 0.06397436 0.03487661
## 18  1.00 5e-01 0.05378205 0.04095040
## 19  3.00 5e-01 0.04602564 0.03161353
## 20  5.00 5e-01 0.05108974 0.02967248
## 21 10.00 5e-01 0.05108974 0.02967248
## 22  0.01 1e+00 0.58185897 0.04745904
## 23  0.10 1e+00 0.58185897 0.04745904
## 24  0.50 1e+00 0.07423077 0.04107102
## 25  1.00 1e+00 0.06141026 0.03026776
## 26  3.00 1e+00 0.05878205 0.02985043
## 27  5.00 1e+00 0.06134615 0.03023342
## 28 10.00 1e+00 0.06134615 0.03023342
## 29  0.01 3e+00 0.58185897 0.04745904
## 30  0.10 3e+00 0.58185897 0.04745904
## 31  0.50 3e+00 0.57935897 0.05129078
## 32  1.00 3e+00 0.47544872 0.12954008
## 33  3.00 3e+00 0.43750000 0.14851496
## 34  5.00 3e+00 0.43750000 0.14851496
## 35 10.00 3e+00 0.43750000 0.14851496
## 36  0.01 5e+00 0.58185897 0.04745904
## 37  0.10 5e+00 0.58185897 0.04745904
## 38  0.50 5e+00 0.58185897 0.04745904
## 39  1.00 5e+00 0.53346154 0.05718980
## 40  3.00 5e+00 0.53089744 0.05751460
## 41  5.00 5e+00 0.53089744 0.05751460
## 42 10.00 5e+00 0.53089744 0.05751460
## 43  0.01 1e+01 0.58185897 0.04745904
## 44  0.10 1e+01 0.58185897 0.04745904
## 45  0.50 1e+01 0.58185897 0.04745904
## 46  1.00 1e+01 0.55134615 0.04898753
## 47  3.00 1e+01 0.55134615 0.05327369
## 48  5.00 1e+01 0.55134615 0.05327369
## 49 10.00 1e+01 0.55134615 0.05327369
## 50  0.01 1e+02 0.58185897 0.04745904
## 51  0.10 1e+02 0.58185897 0.04745904
## 52  0.50 1e+02 0.58185897 0.04745904
## 53  1.00 1e+02 0.58185897 0.04745904
## 54  3.00 1e+02 0.58185897 0.04745904
## 55  5.00 1e+02 0.58185897 0.04745904
## 56 10.00 1e+02 0.58185897 0.04745904
```

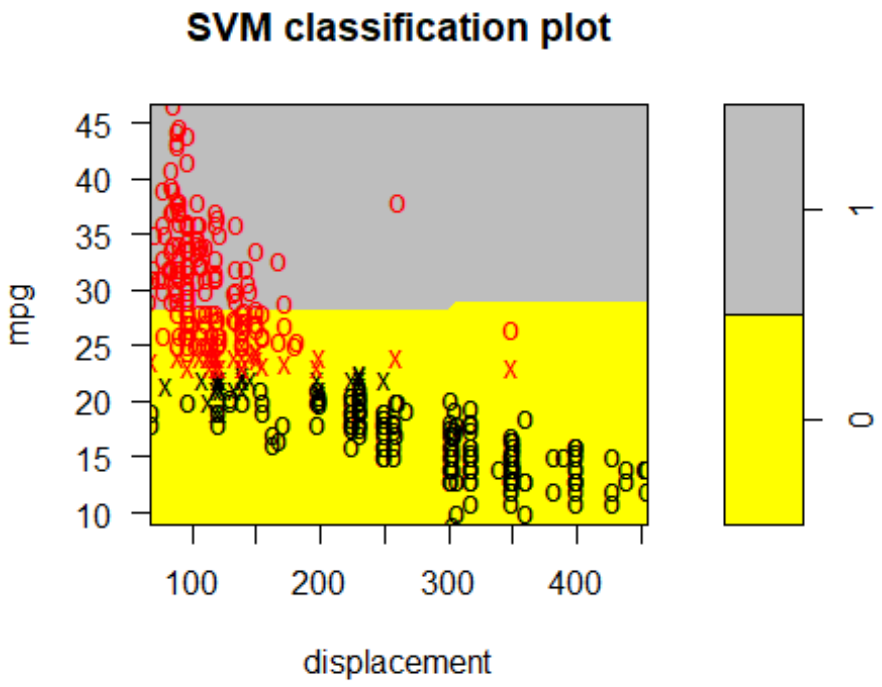
The best parameter for which cross-validation error is minimized is cost = 10 and gamma = 0.01.

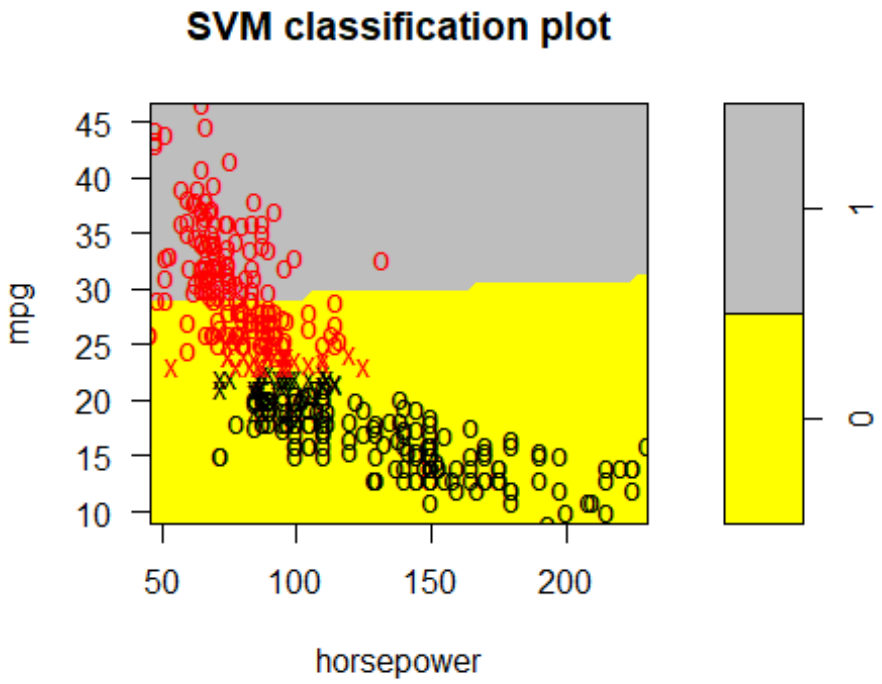
D)

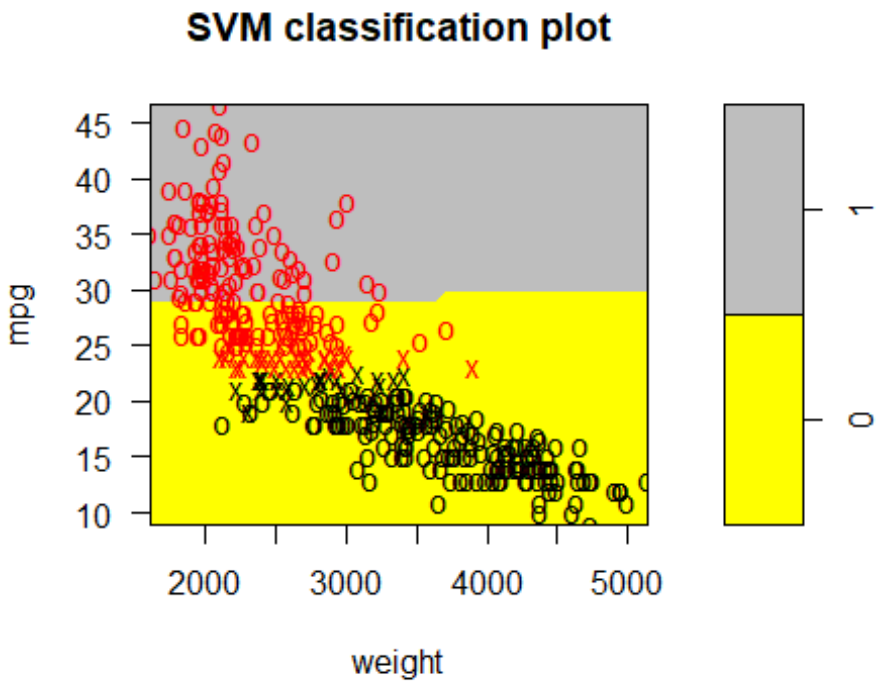
Make some plots to back up your assertions in (b) and (c).

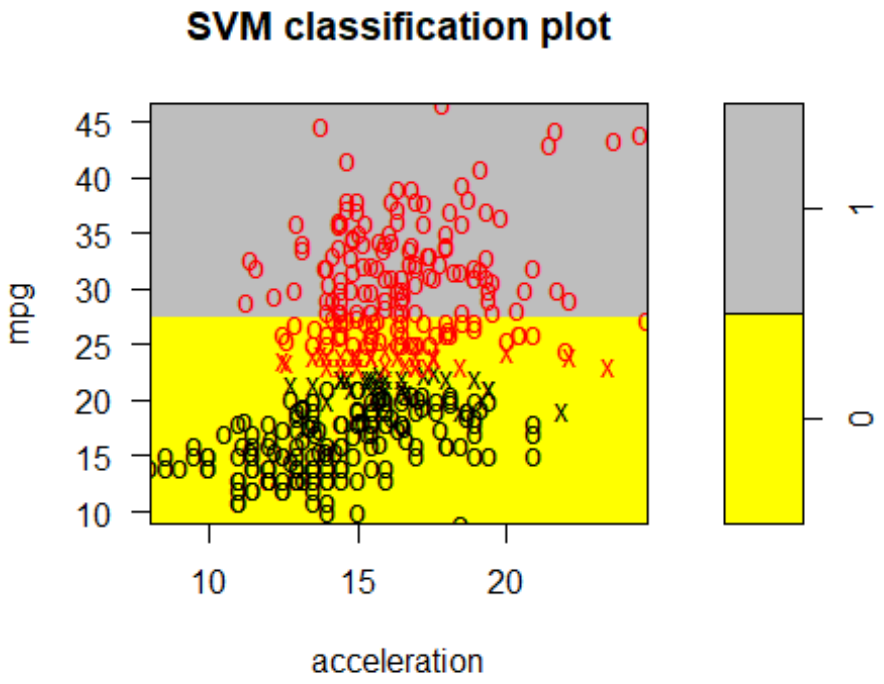
```
linear <- svm(mpglevel ~ ., data = Auto, kernel = "linear", cost = 1)
poly <- svm(mpglevel ~ ., data = Auto, kernel = "polynomial", cost = 10,
degree = 2)
radial <- svm(mpglevel ~ ., data = Auto, kernel = "radial", cost = 10, gamma
= 0.01)
plotting <- function(fit) {
    for (name in names(Auto)[!(names(Auto) %in% c("mpg", "mpglevel",
"name"))]) {
        plot(fit, Auto, as.formula(paste("mpg~", name, sep =
"")),col=c("yellow","gray"))
    }
}
plotting(linear)
```

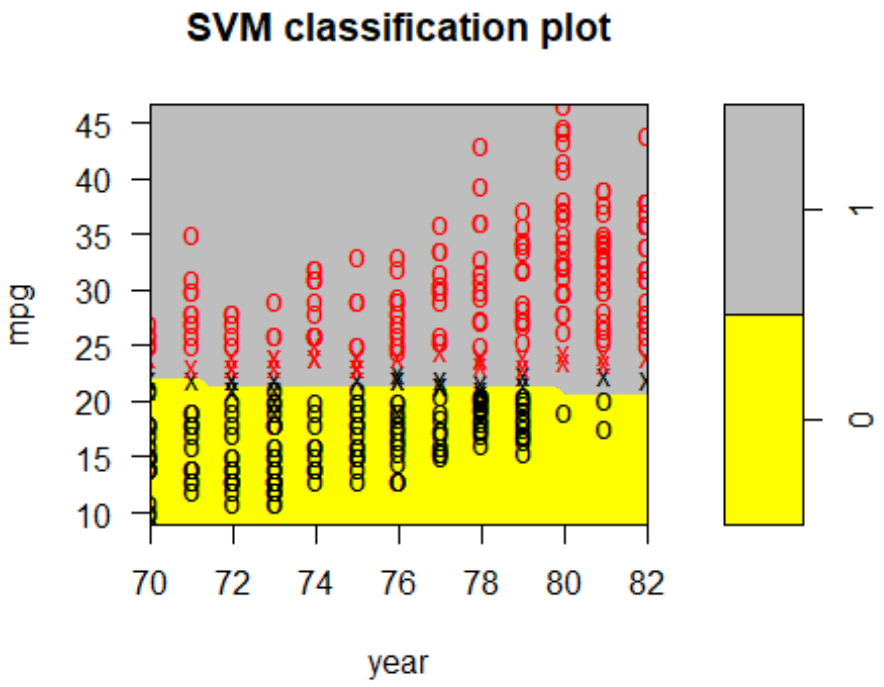

SVM classification plot
mpg
cylinders
45
40
35
30
25
20
15
10
3
4
5
6
7
8
1
0

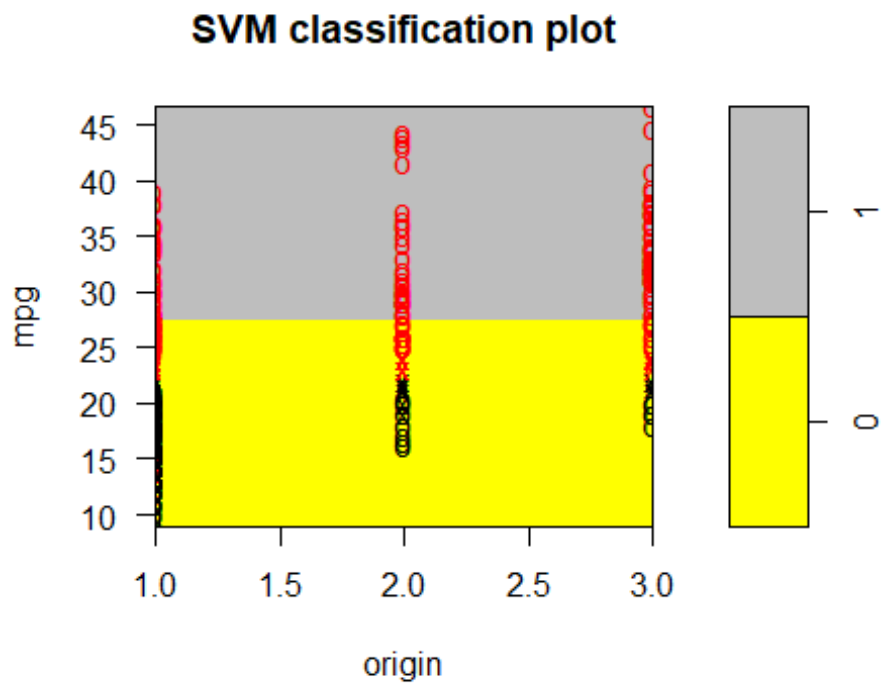

SVM classification plot
mpg
displacement
45
40
35
30
25
20
15
10
100
200
300
400
1
0


SVM classification plot
mpg
horsepower
45
40
35
30
25
20
15
10
50
100
150
200
1
0

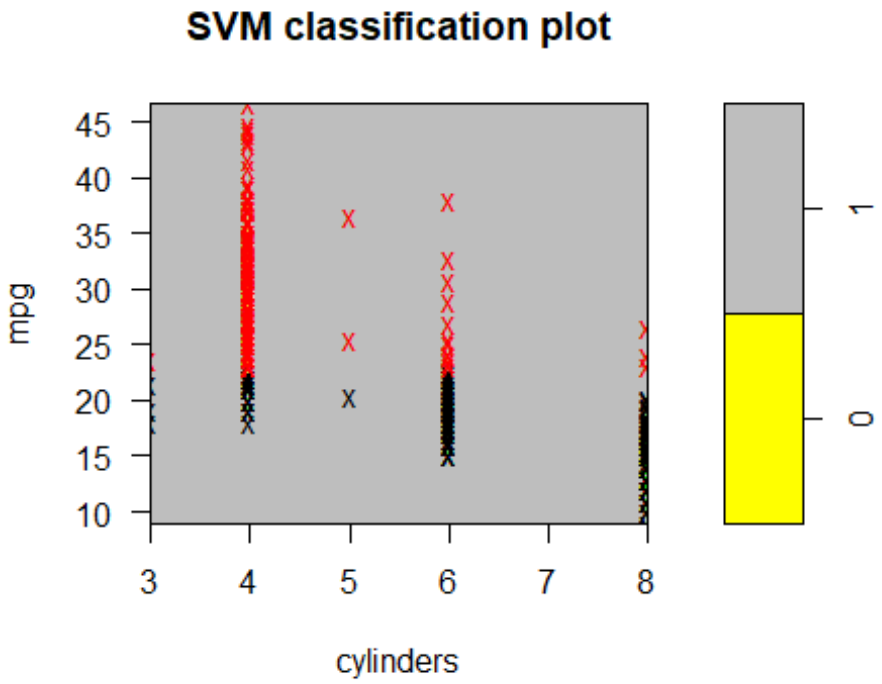

SVM classification plot
mpg
weight
45
40
35
30
25
20
15
10
2000
3000
4000
5000
1
0

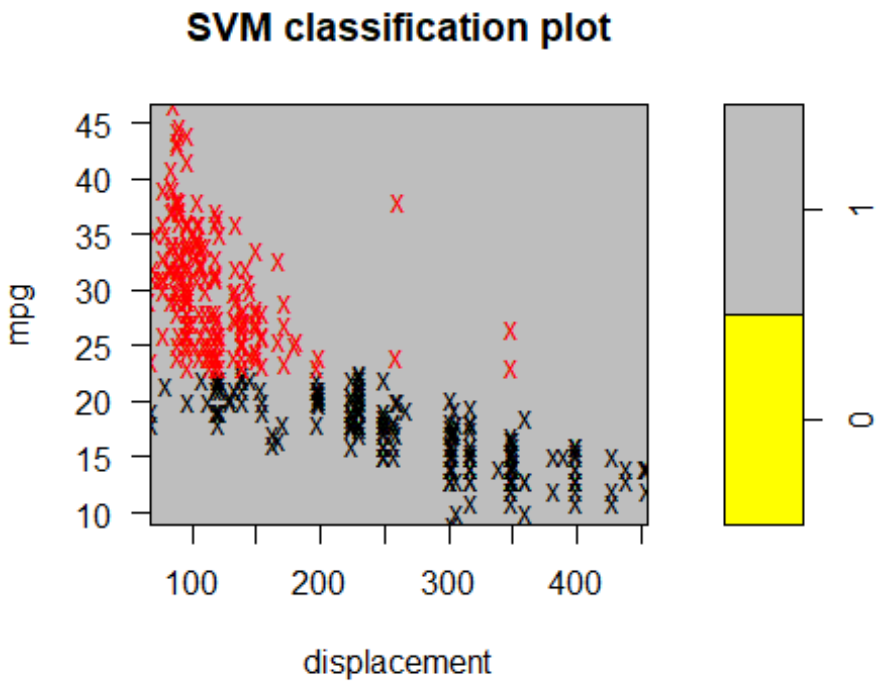

SVM classification plot
mpg
45
40
35
30
25
20
15
10
10
15
20
acceleration
1
0

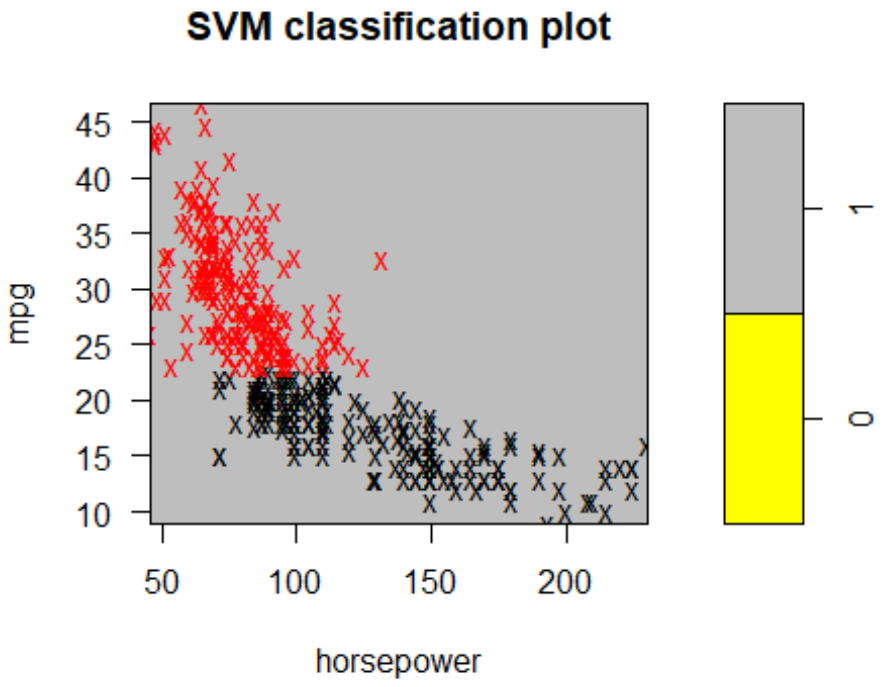

SVM classification plot
mpg
45
40
35
30
25
20
15
10
70
72
74
76
78
80
82
year
1
0

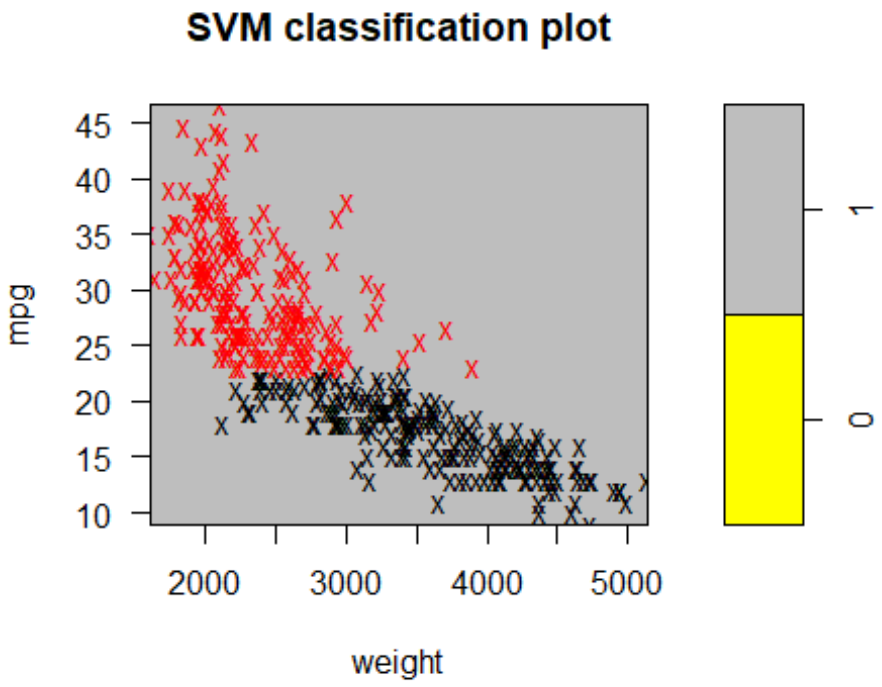


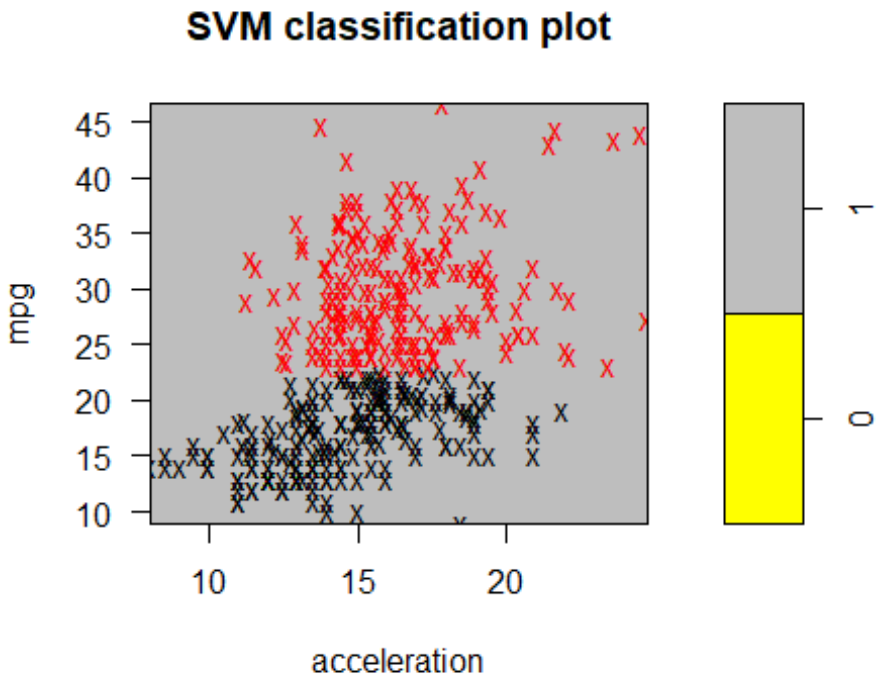

```
plotting(poly)
```

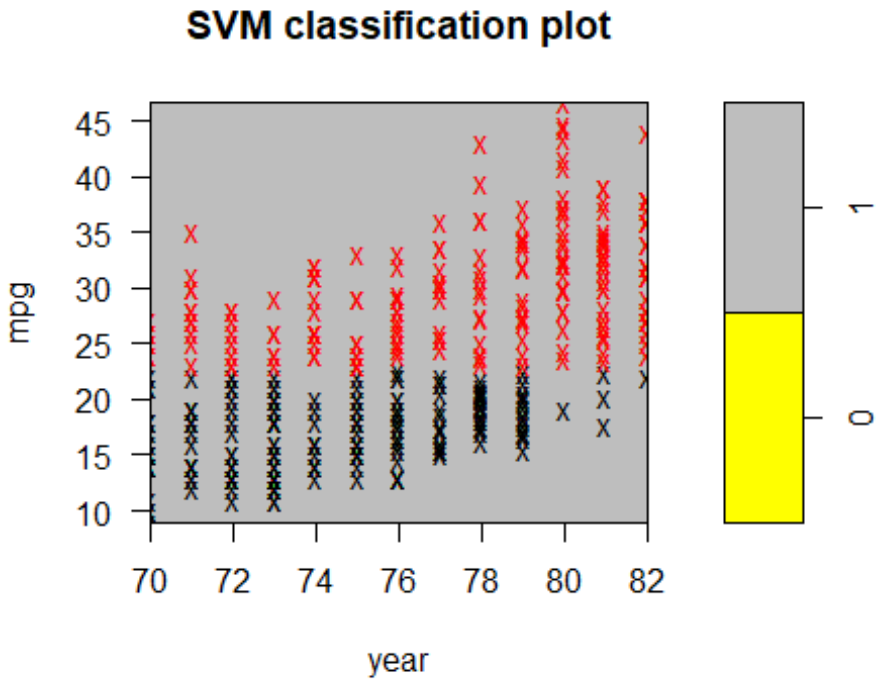

SVM classification plot
mpg
cylinders
45
40
35
30
25
20
15
10
3
4
5
6
7
8
1
0

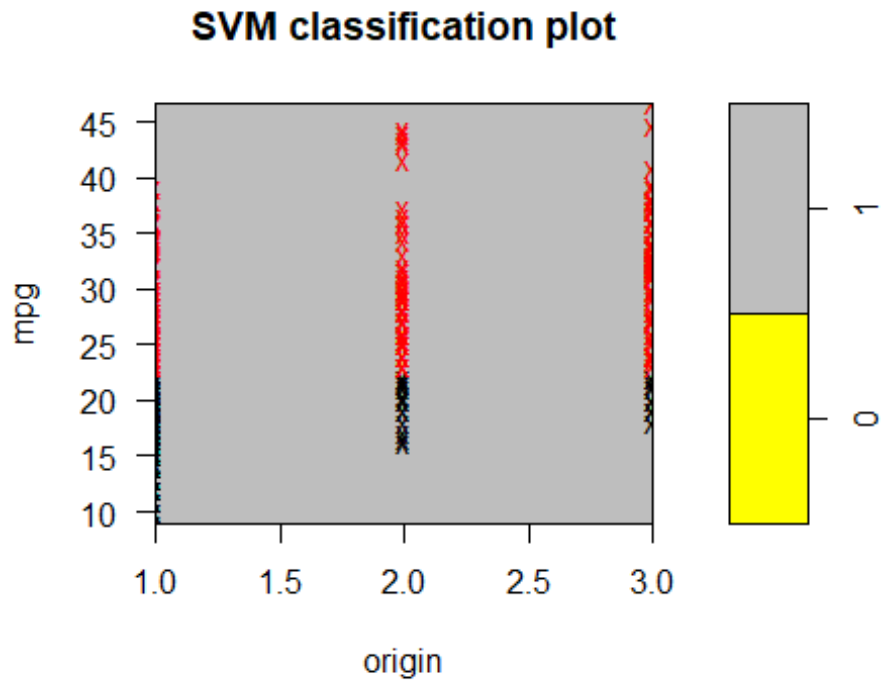

SVM classification plot
mpg
displacement
45
40
35
30
25
20
15
10
100
200
300
400
1
0


SVM classification plot
mpg
45
40
35
30
25
20
15
10
50
100
150
200
horsepower
1
0

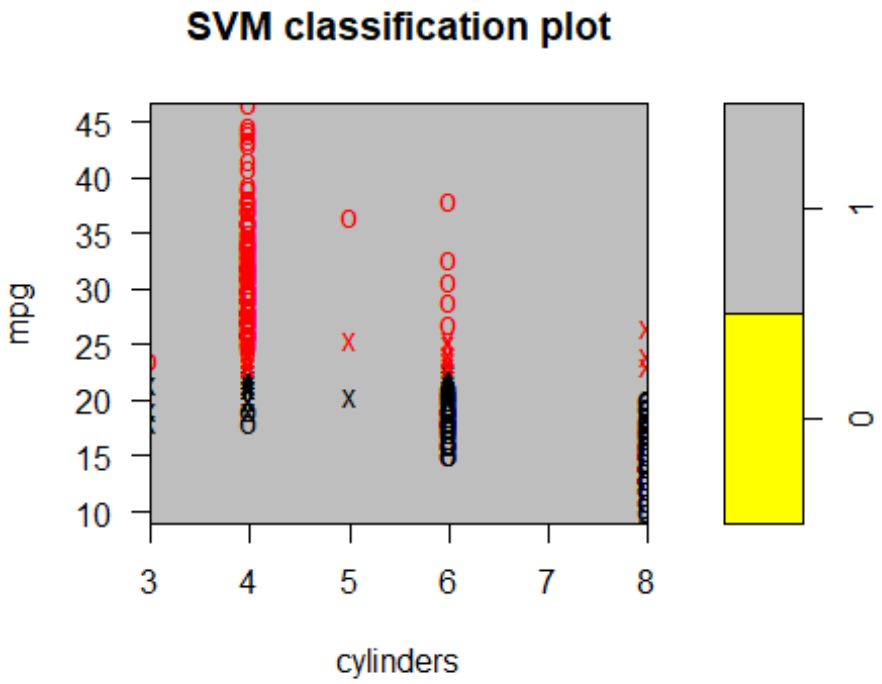

SVM classification plot
mpg
45
40
35
30
25
20
15
10
2000
3000
4000
5000
weight
1
0

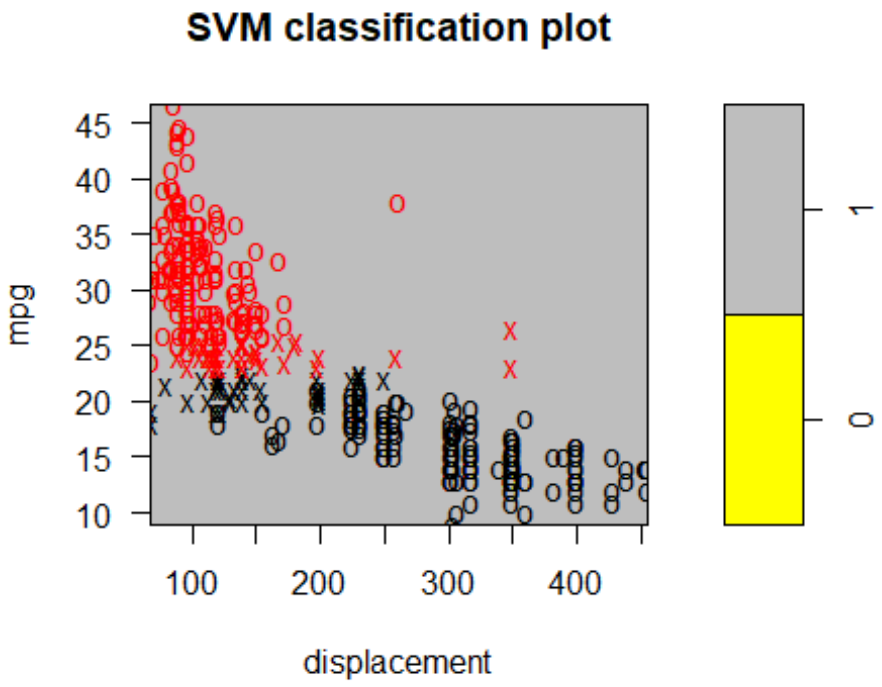

SVM classification plot
mpg
45
40
35
30
25
20
15
10
10
15
20
acceleration
1
0

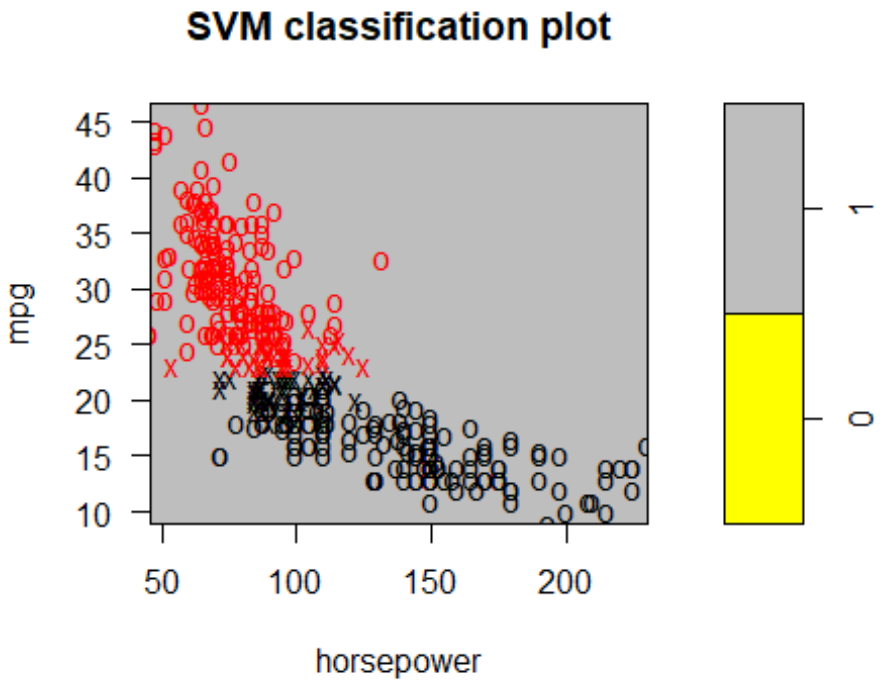

SVM classification plot
mpg
45
40
35
30
25
20
15
10
70
72
74
76
78
80
82
year
1
0

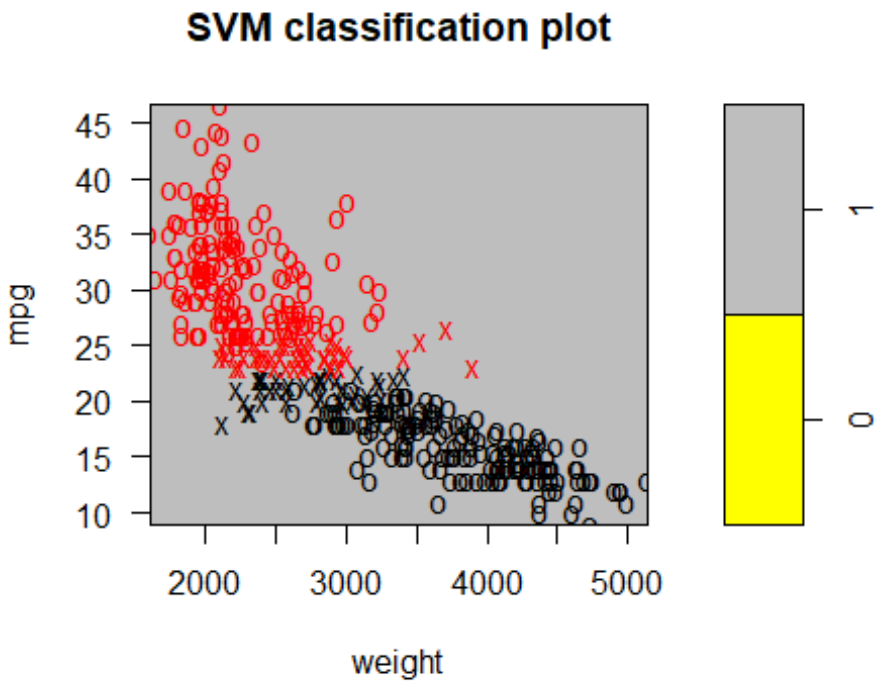


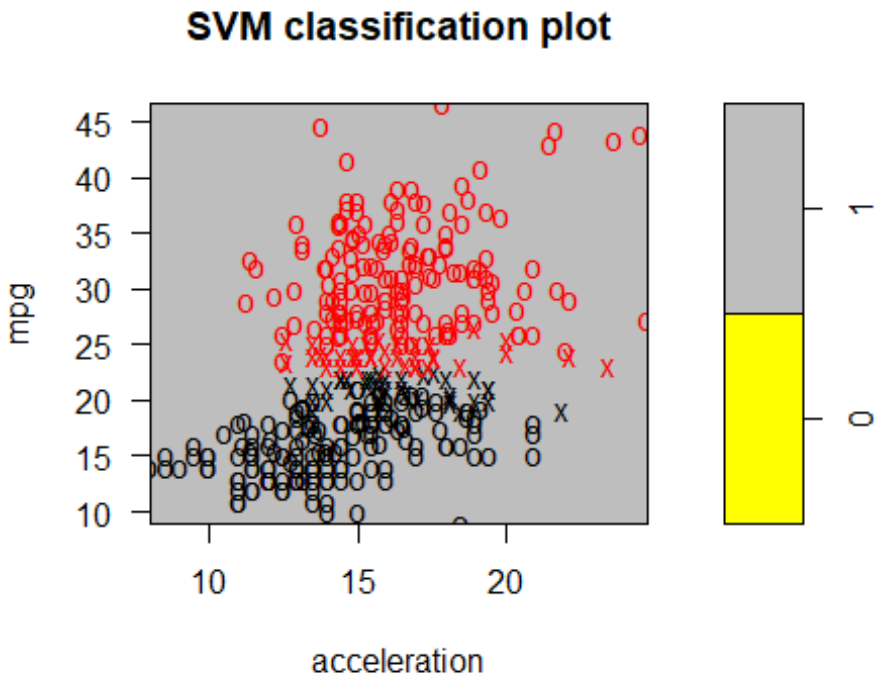

```
plotting(radial)
```

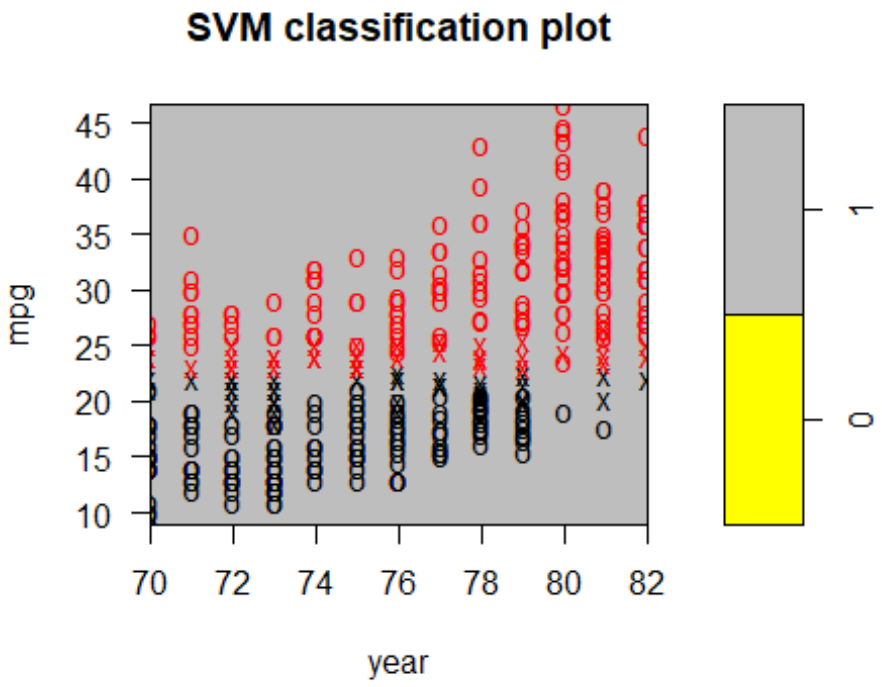

SVM classification plot
mpg
cylinders
45
40
35
30
25
20
15
10
3
4
5
6
7
8
1
0

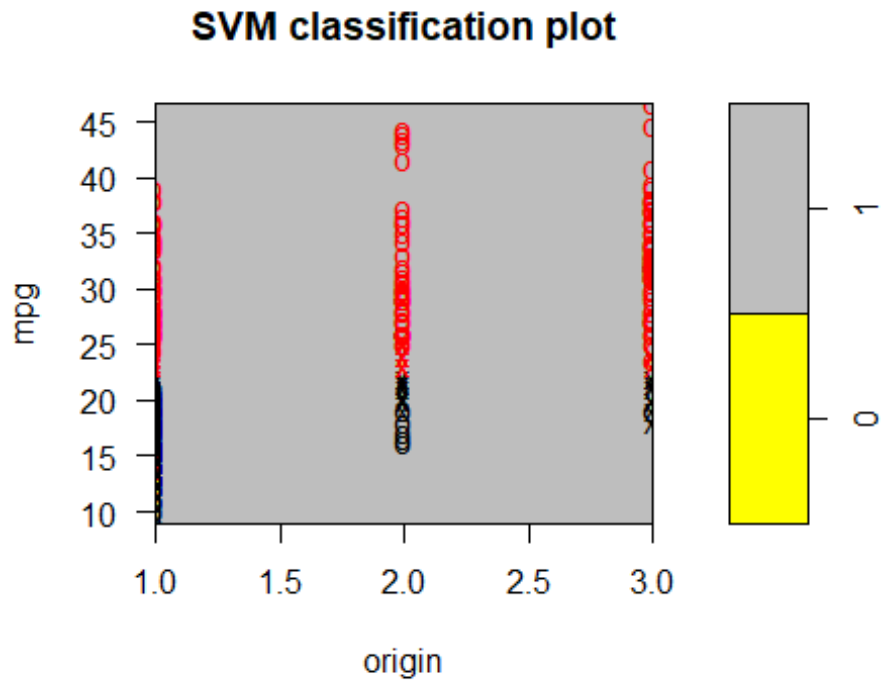

SVM classification plot
mpg
displacement
45
40
35
30
25
20
15
10
100
200
300
400
1
0


SVM classification plot
mpg
horsepower
45
40
35
30
25
20
15
10
50
100
150
200
1
0


SVM classification plot
mpg
weight
45
40
35
30
25
20
15
10
2000
3000
4000
5000
1
0


SVM classification plot
mpg
45
40
35
30
25
20
15
10
10
15
20
acceleration
1
0

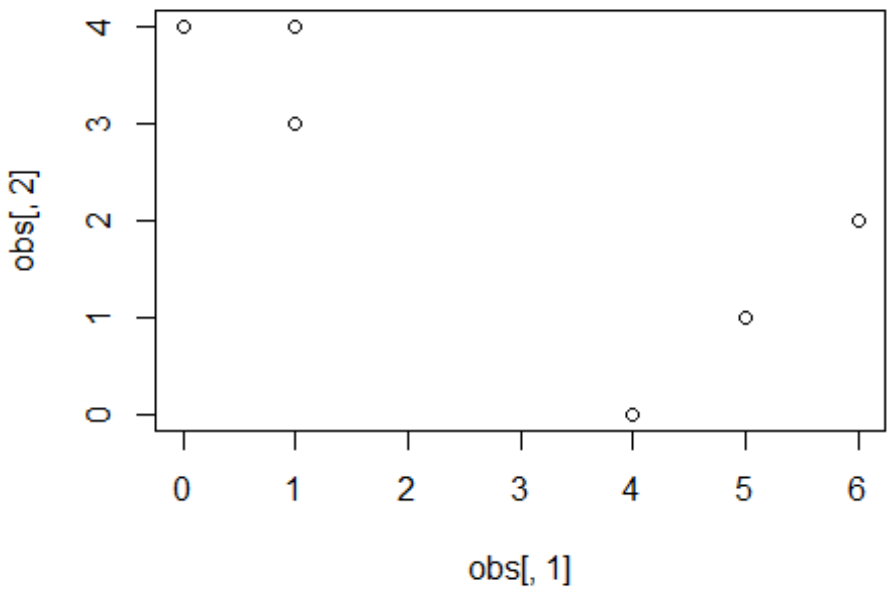

SVM classification plot
mpg
45
40
35
30
25
20
15
10
70
72
74
76
78
80
82
year
1
0



Question 3 - Ex 3 ch 10

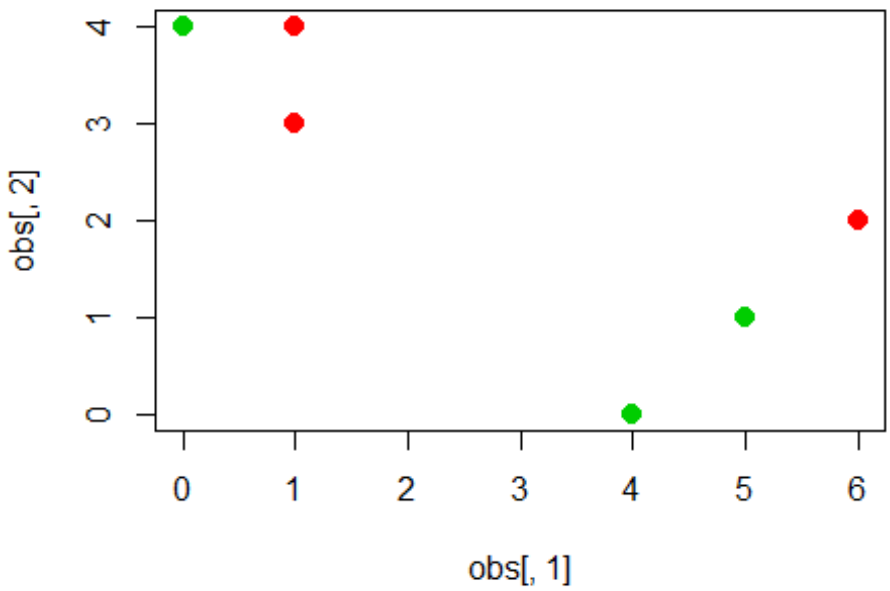
A) Plot the observations.

```
obs <- cbind(c(1, 1, 0, 5, 6, 4), c(4, 3, 4, 1, 2, 0))
plot(obs[,1], obs[,2])
```



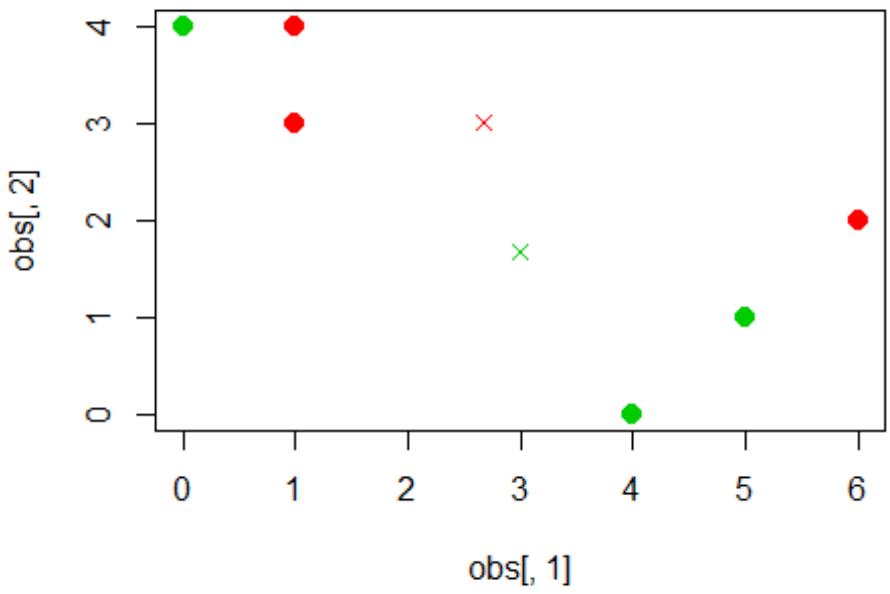
B) Randomly assign a cluster label to each observation. Report the cluster labels for each observation.

```
set.seed(1)
label <- sample(2, nrow(obs), replace = T)
label

## [1] 1 1 2 2 1 2

plot(obs[, 1], obs[, 2], col = (label + 1), pch = 20, cex = 2)
```



C) Compute the centroid for each cluster.

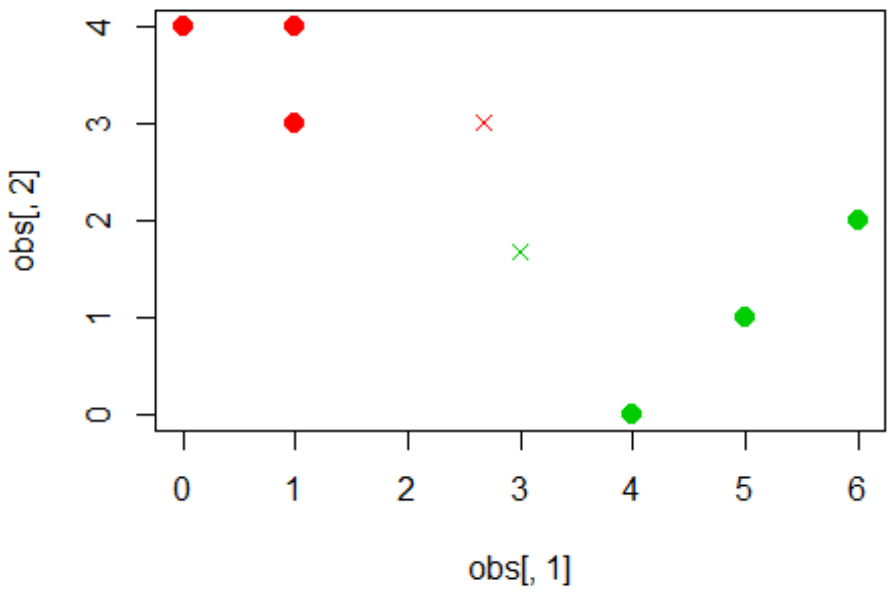
```
centroid1 <- c(mean(obs[label == 1, 1]), mean(obs[label == 1, 2]))
centroid2 <- c(mean(obs[label == 2, 1]), mean(obs[label == 2, 2]))
plot(obs[,1], obs[,2], col=(label + 1), pch = 20, cex = 2)
points(centroid1[1], centroid1[2], col = 2, pch = 4)
points(centroid2[1], centroid2[2], col = 3, pch = 4)
```



Centroid is calculated by taking the mean of points having same label and have been marked as x.

D) Assign each observation to the centroid to which it is closest, in terms of Euclidean distance. Report the cluster labels for each observation.

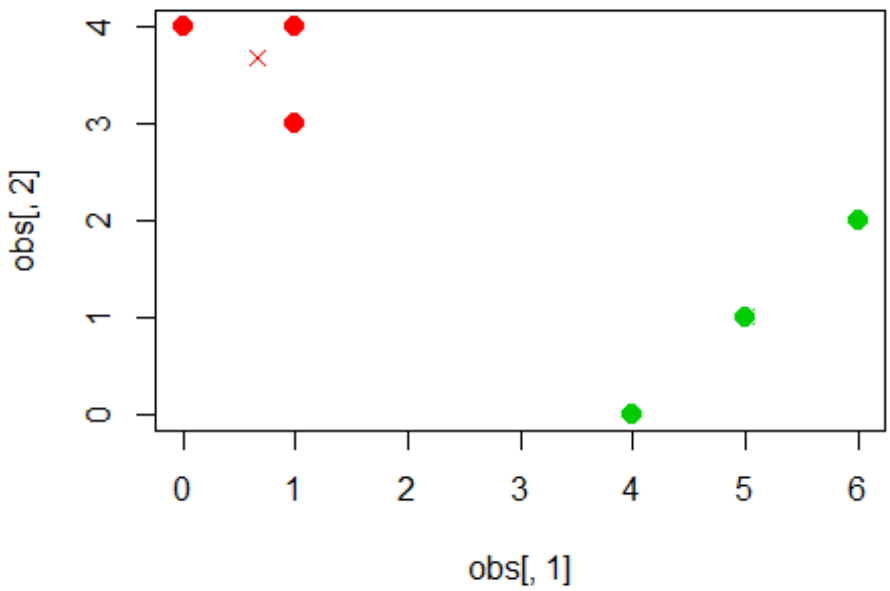
```
label <- c(1, 1, 1, 2, 2, 2)
plot(obs[, 1], obs[, 2], col = (label + 1), pch = 20, cex = 2)
points(centroid1[1], centroid1[2], col = 2, pch = 4)
points(centroid2[1], centroid2[2], col = 3, pch = 4)
```



Reassigned the points to that cluster whose centroid it is closer to (seen in the above plot).

E) Repeat (c) and (d) until the answers obtained stop changing.

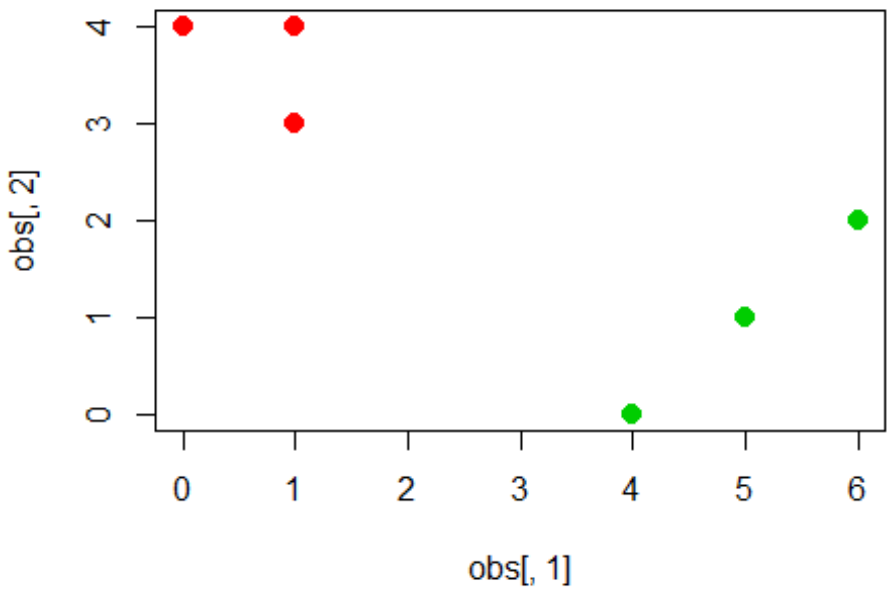
```
centroid1 <- c(mean(obs[label == 1, 1]), mean(obs[label == 1, 2]))
centroid2 <- c(mean(obs[label == 2, 1]), mean(obs[label == 2, 2]))
plot(obs[,1], obs[,2], col=(label + 1), pch = 20, cex = 2)
points(centroid1[1], centroid1[2], col = 2, pch = 4)
points(centroid2[1], centroid2[2], col = 3, pch = 4)
```



- We computed the centroids for the two clusters again. - When we assign each observation to the centroid to which it is closest, the clusters to which the points belong don't change, so the algorithm ends at this step.

F) In your plot from (a), color the observations according to the clusters labels obtained.

```
plot(obs[, 1], obs[, 2], col=(label + 1), pch = 20, cex = 2)
```



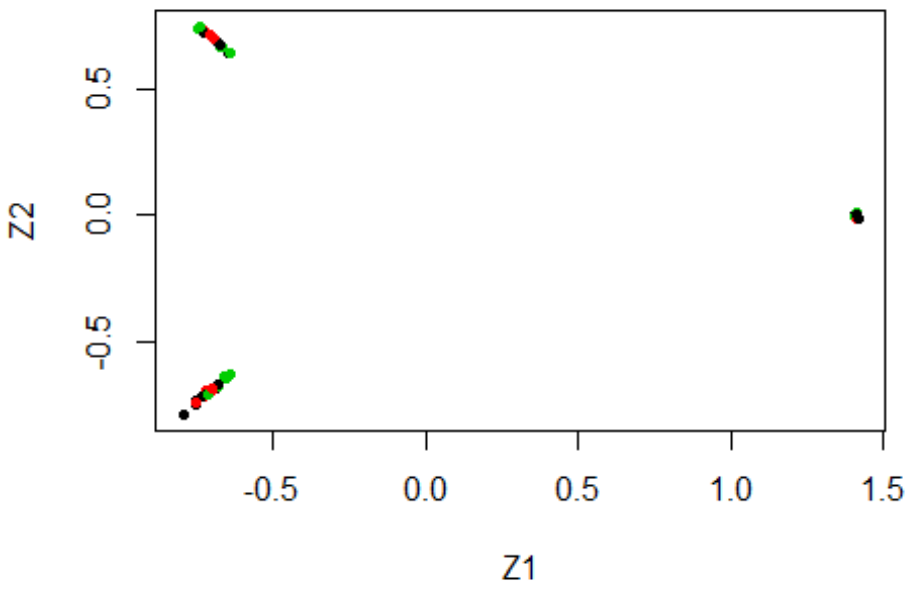
Question 4 - Ex 10 ch 10

A)
Generate a simulated data set with 20 observations in each of three classes (i.e. 60 observations total), and 50 variables.

```
set.seed(2)
x<-matrix(rnorm(20*3*50,mean=0,sd=0.05),ncol=50)
x[1:25,2]<-1
x[21:40,1]<-2
x[21:40,2]<-2
x[41:60,1]<-1
true_labels<-c(rep(1,20),rep(2,20),rep(3,20))
```

B) Perform PCA on the 60 observations and plot the first two principal component score vectors. Use a different color to indicate the observations in each of the three classes. If the three classes appear separated in this plot, then continue on to part (c). If not, then return to part (a) and modify the simulation so that there is greater separation between the three classes. Do not continue to part (c) until the three classes show at least some separation in the first two principal component score vectors.

```
pca<-prcomp(x)
plot(pca$x[,1:2],col=1:3,xlab="Z1",ylab="Z2",pch=20)
```



C) Perform K-means clustering of the observations with K = 3. How well do the clusters that you obtained in K-means clustering compare to the true class labels?

```
km <- kmeans(x, 3, nstart = 20)
table(true_labels, km$cluster)

##
## true_labels  1  2  3
##           1 20  0  0
##           2  0 20  0
##           3  0  0 20
```

- We have 3 classes in the original data with 20 observations each which are perfectly clustered into three classes after performing kmeans clustering as seen above.

D) Perform K-means clustering with K = 2. Describe your results.

```
km2 <- kmeans(x, 2, nstart = 20)
table(true_labels, km2$cluster)

##
## true_labels  1  2
##           1 20  0
##           2  0 20
##           3 20  0
```

- The original data having 3 classes are now clustered into 2 classes. - The third cluster has combined with the first cluster completely.

E) Now perform K-means clustering with K = 4, and describe your results.

```
km4<- kmeans(x, 4, nstart = 20)
table(true_labels, km4$cluster)

##
## true_labels  1  2  3  4
##           1  0 11  0  9
##           2 20  0  0  0
##           3  0  0 20  0
```

- The original data with 3 classes have been segmented into 4 clusters.

- Among the original cluster two of the classes remain as such (cluster 2 is labelled 1 and cluster 3 is labelled 3).

- But observations of class one is now distributed over two clusters, 11 observations in cluster 2 and 9 observations in cluster 4.

F) Now perform K-means clustering with K = 3 on the first two principal component score vectors, rather than on the raw data. That is, perform K-means clustering on the 60 × 2 matrix of which the first column is the first principal component score vector, and the second column is the second principal component score vector. Comment on the results.

```
km_pca<-kmeans(pca$x[,1:2],3,nstart=20)
table(true_labels,km_pca$cluster)

##
## true_labels  1  2  3
##           1  0  0 20
##           2  0 20  0
##           3 20  0  0
```

The observations are perfectly clustered together once again, with their labels alone assigned differently.

G) Using the scale() function, perform K-means clustering with K = 3 on the data after scaling each variable to have standard deviation one. How do these results compare to those obtained in (b)? Explain.

```
km_scale<-kmeans(scale(x),3,nstart=20)
table(true_labels,km_scale$cluster)

##
## true_labels  1  2  3
##           1  8  2 10
##           2  0 19  1
##           3 11  1  8
```

- Now the observations are no more perfectly clustered and the results are not as good as the unscaled clustering.

-Scaling affects the distance between the observations and this can change the clusters altogether.