Rakesh Veetekat

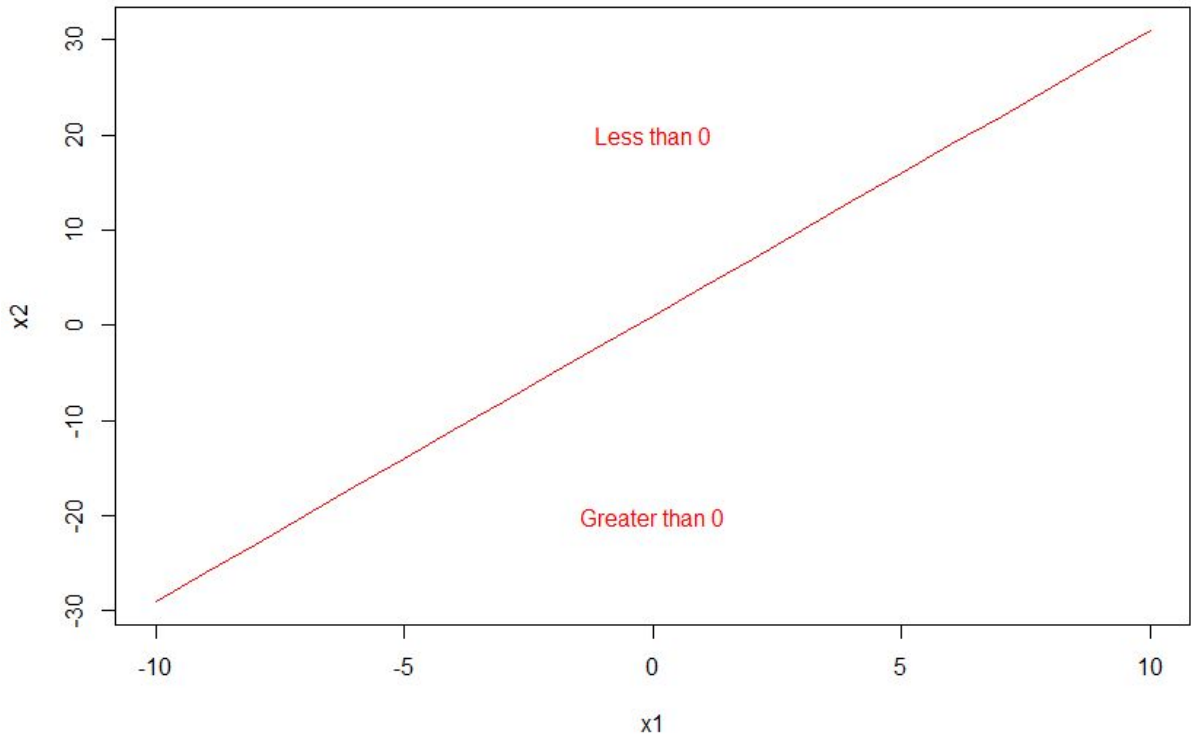
Conceptual and theoretical questions

1. (Ch. 9, Question 1)

   This problem involves hyperplanes in two dimensions.

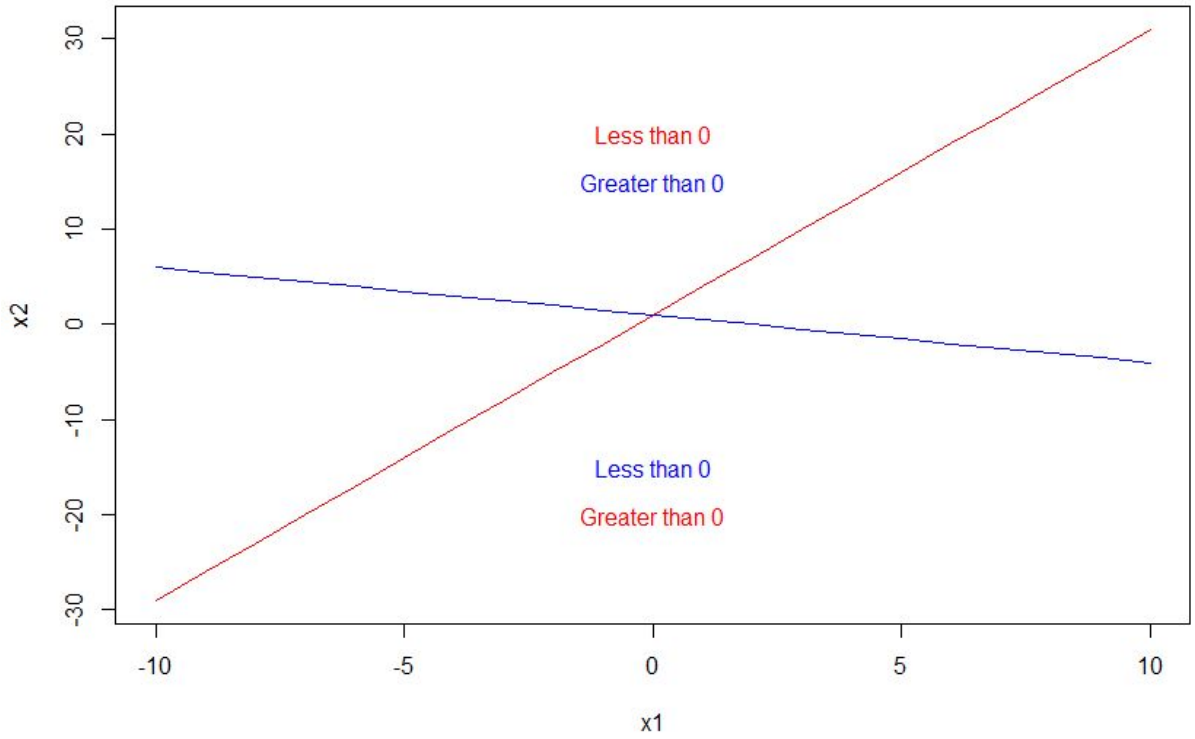
   a. Sketch the hyperplane 1 + 3X1 − X2 = 0. Indicate the set of points for which 1 + 3X1 − X2 > 0, as well as the set of points for which 1 + 3X1 − X2 < 0.

```
> x1 = -10:10
> x2 = 1 + 3 * x1
> plot(x1, x2, type = "l", col = "red")
> text(c(0), c(-20), "Greater than 0", col = "red")
> text(c(0), c(20), "Less than 0", col = "red")
>
```



   b. On the same plot, sketch the hyperplane −2 + X1 + 2X2 = 0. Indicate the set of points for which −2 + X1 + 2X2 > 0, as well as the set of points for which −2 + X1 + 2X2 < 0.

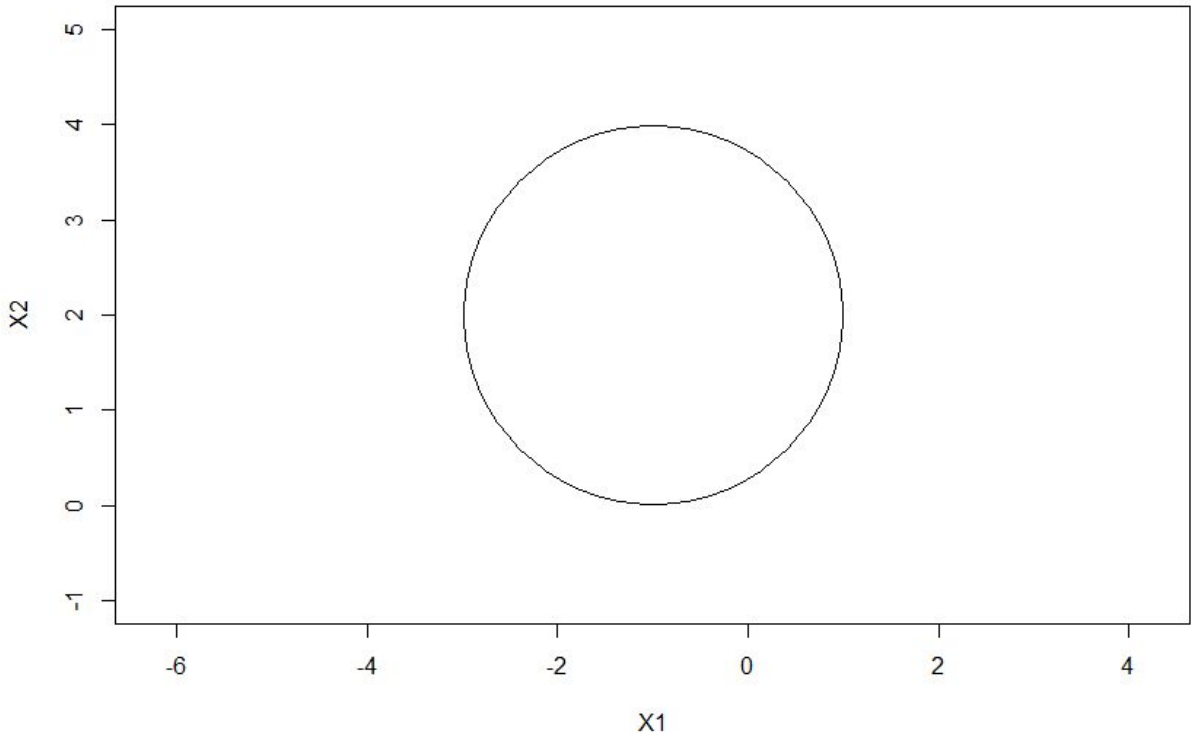
```
> lines(x1, 1 - x1/2, col = "blue")
> text(c(0), c(-15), "Less than 0", col = "blue")
> text(c(0), c(15), "Greater than 0", col = "blue")
>
```



2. (Ch. 9, Question 2)
   We have seen that in p = 2 dimensions, a linear decision boundary takes the form β0+β1X1+β2X2 = 0. We now investigate a non-linear decision boundary.
   a. Sketch the curve

$$(1 + X_1)^2 + (2 - X_2)^2 = 4.$$

```
> plot(NA, NA, type = "n", xlim = c(-4, 2), ylim = c(-1, 5), asp = 1, xlab = "X1", ylab = "X2")
> symbols(c(-1), c(2), circles = c(2), add = TRUE, inches = FALSE)
>
```



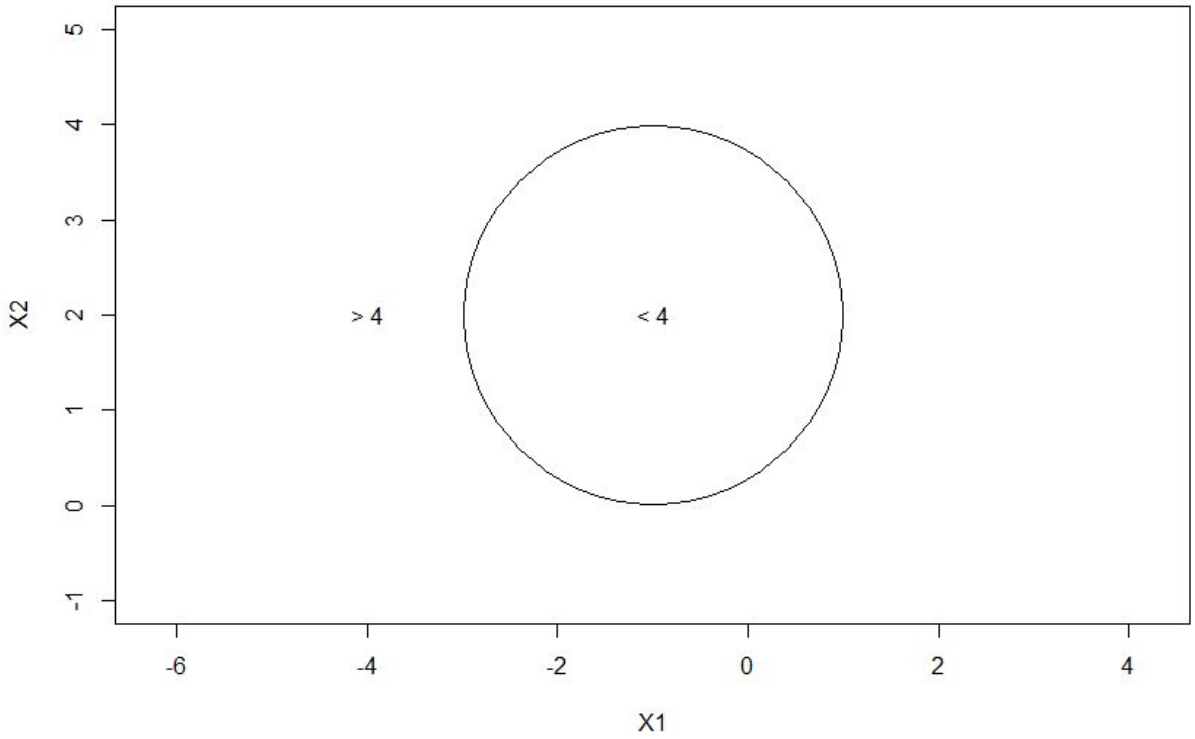
b. On your sketch, indicate the set of points for which

$$(1 + X_1)^2 + (2 - X_2)^2 > 4,$$

as well as the set of points for which

$$(1 + X_1)^2 + (2 - X_2)^2 \leq 4.$$

```
> plot(NA, NA, type = "n", xlim = c(-4, 2), ylim = c(-1, 5), asp = 1, xlab = "X1", ylab = "X2")
> symbols(c(-1), c(2), circles = c(2), add = TRUE, inches = FALSE)
> text(c(-1), c(2), "< 4")
> text(c(-4), c(2), "> 4")
>
```

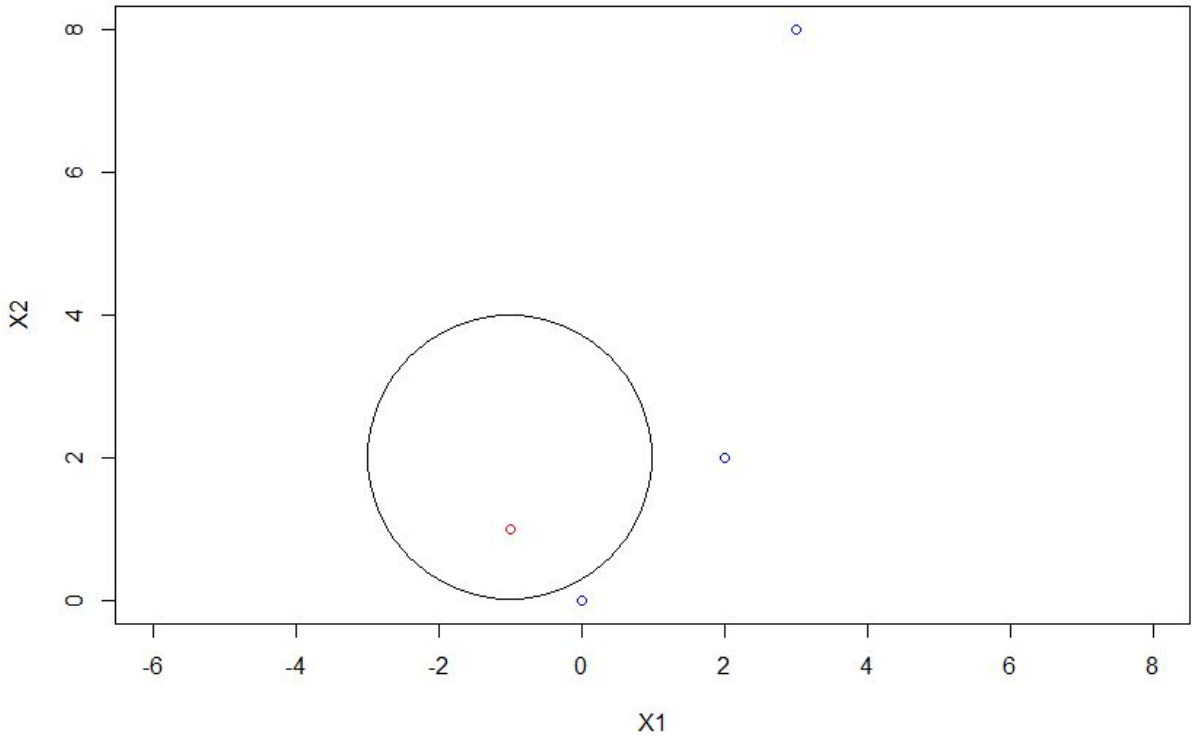


c. Suppose that a classifier assigns an observation to the blue class if

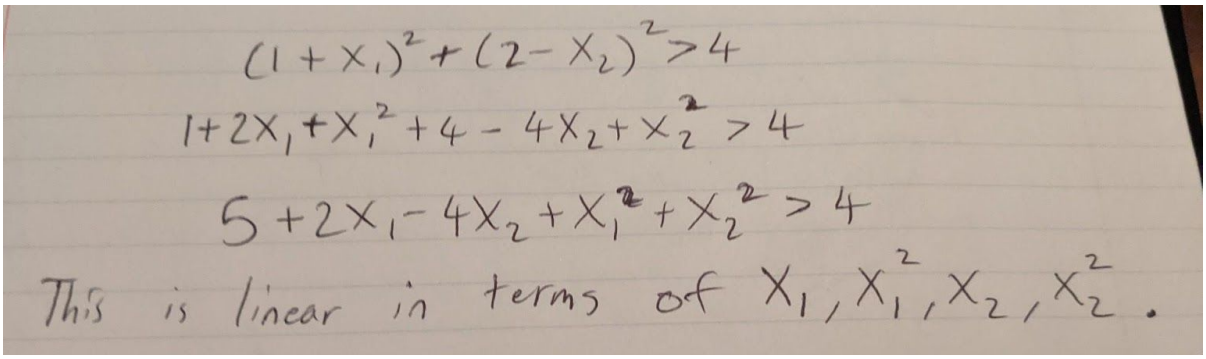
$$(1 + X_1)^2 + (2 - X_2)^2 > 4,$$

and to the red class otherwise. To what class is the observation (0, 0) classified? (−1, 1)? (2, 2)? (3, 8)?

```
> plot(c(0, -1, 2, 3), c(0, 1, 2, 8), col = c("blue", "red", "blue", "blue"), type = "p", asp = 1, xlab = "X1", ylab = "X2")
> symbols(c(-1), c(2), circles = c(2), add = TRUE, inches = FALSE)
>
```



d. Argue that while the decision boundary in (c) is not linear in terms of X1 and X2, it is linear in terms of X1, X2 1 , X2, and X2 2 .



3. (Ch. 9, Question 3)

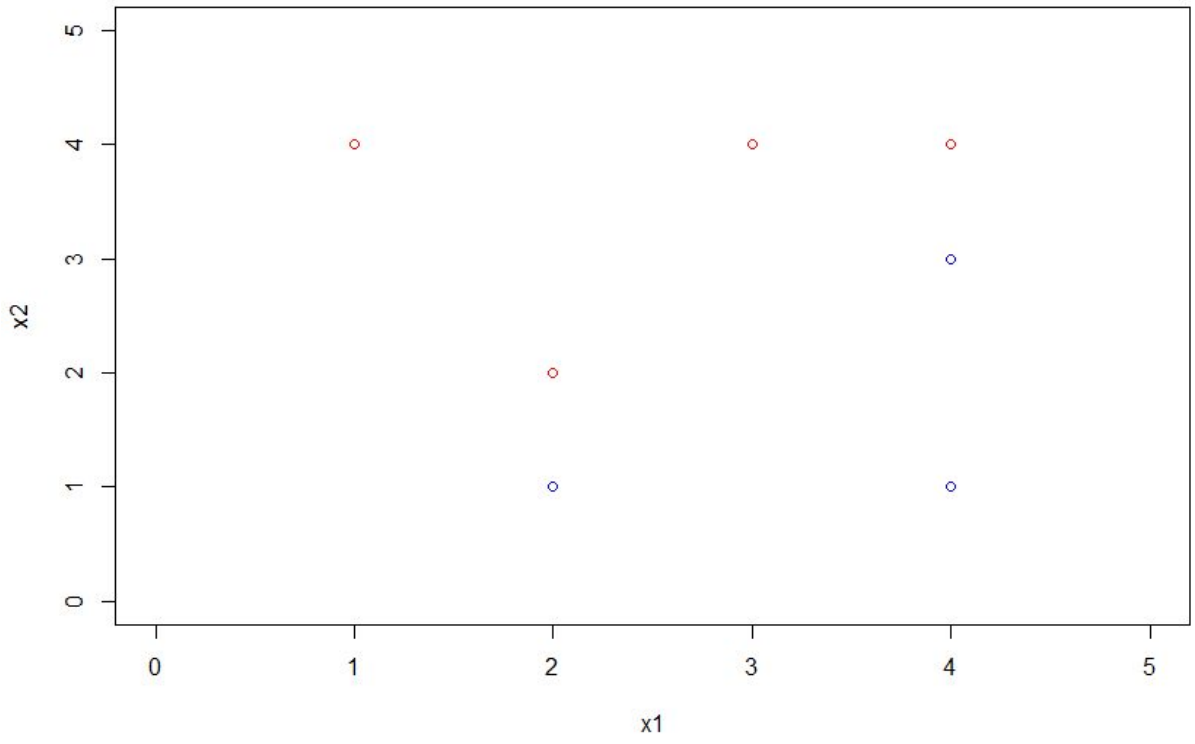
Here we explore the maximal margin classifier on a toy data set.

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

Obs.	X_1	X_2	Y
1	3	4	Red
2	2	2	Red
3	4	4	Red
4	1	4	Red
5	2	1	Blue
6	4	3	Blue
7	4	1	Blue

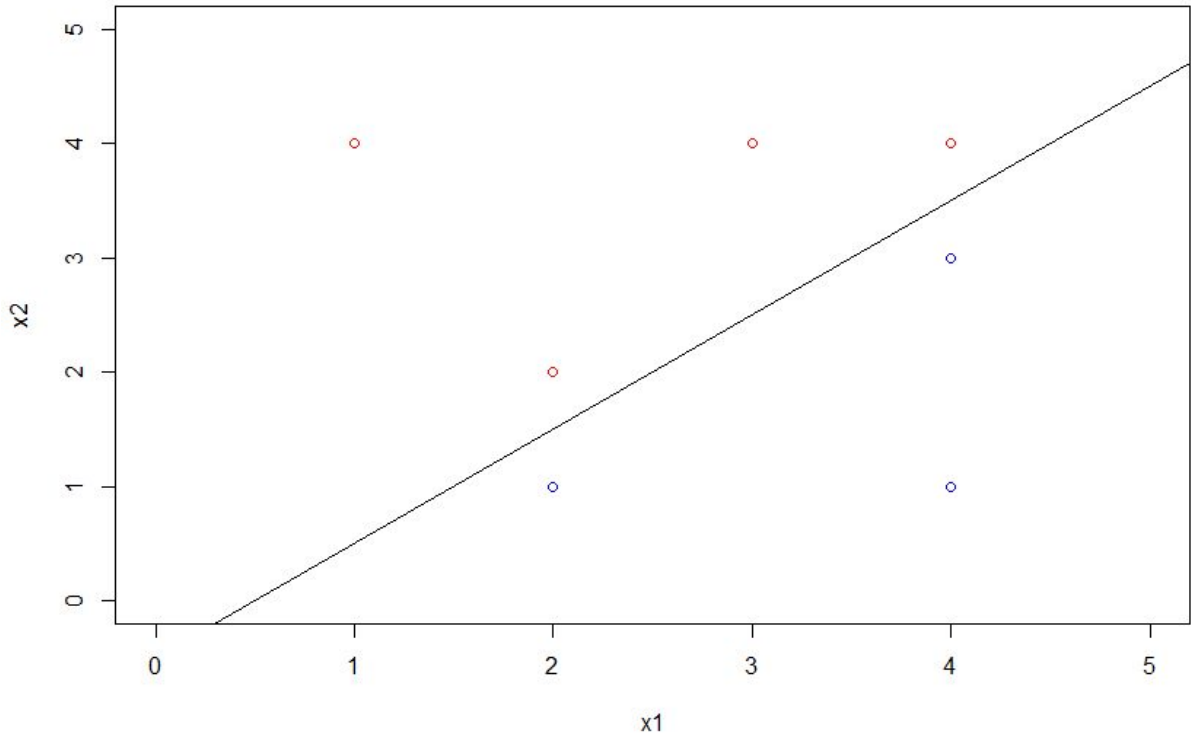
Sketch the observations.

```
> 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 (of the form (9.1)).

```
> plot(x1, x2, col = colors, xlim = c(0, 5), ylim = c(0, 5))
> abline(-0.5, 1)
>
```

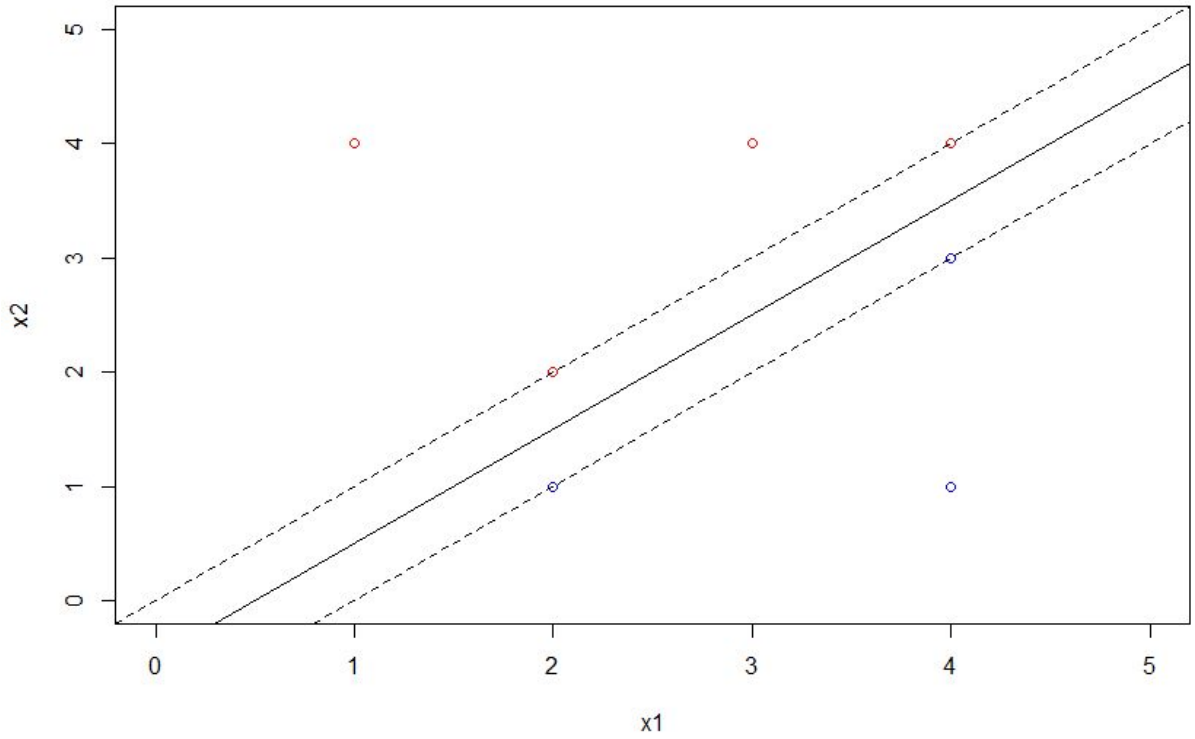


c. Describe the classification rule for the maximal margin classifier. It should be something along the lines of “Classify to Red if β0 + β1X1 + β2X2 > 0, and classify to Blue otherwise.” Provide the values for β0, β1, and β2.

The classification rule for the classifier is the observation should classify as Red if $0.5 - X_1 + X_2 > 0$

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))
> abline(-0.5, 1)
> abline(-1, 1, lty = 2)
> abline(0, 1, lty = 2)
>
```



e. Indicate the support vectors for the maximal margin classifier.

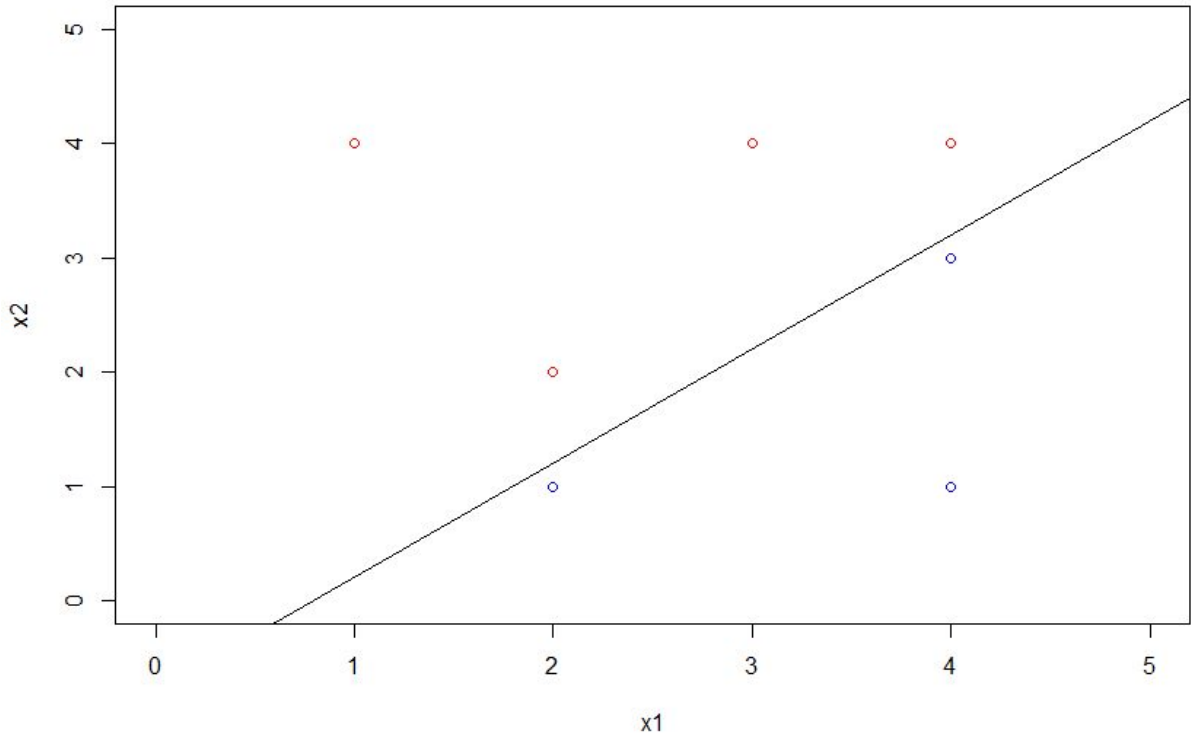
The support vectors for the maximal margin classifier here are the vectors from the points (2,1), (2,2), (4,3), and (4,4) to the optimal separating hyperplane.

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

A slight movement in the seventh observation would not affect the maximal margin hyperplane because its movement would be outside of the margin, and it is also not a support vector.

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

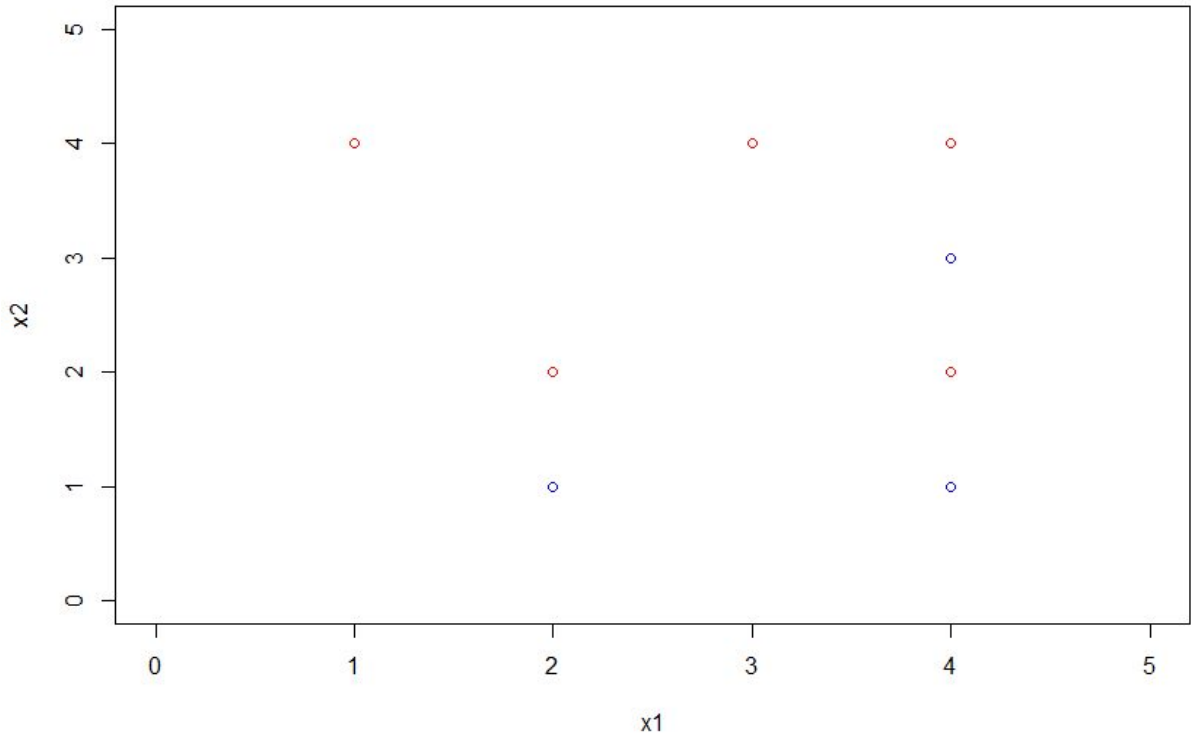
```
> plot(x1, x2, col = colors, xlim = c(0, 5), ylim = c(0, 5))
> abline(-0.8, 1)
>
```



The equation for this hyperplane is $-0.8 - X_1 + X_2 = 0$

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(4), c(2), col = c("red"))
>
```



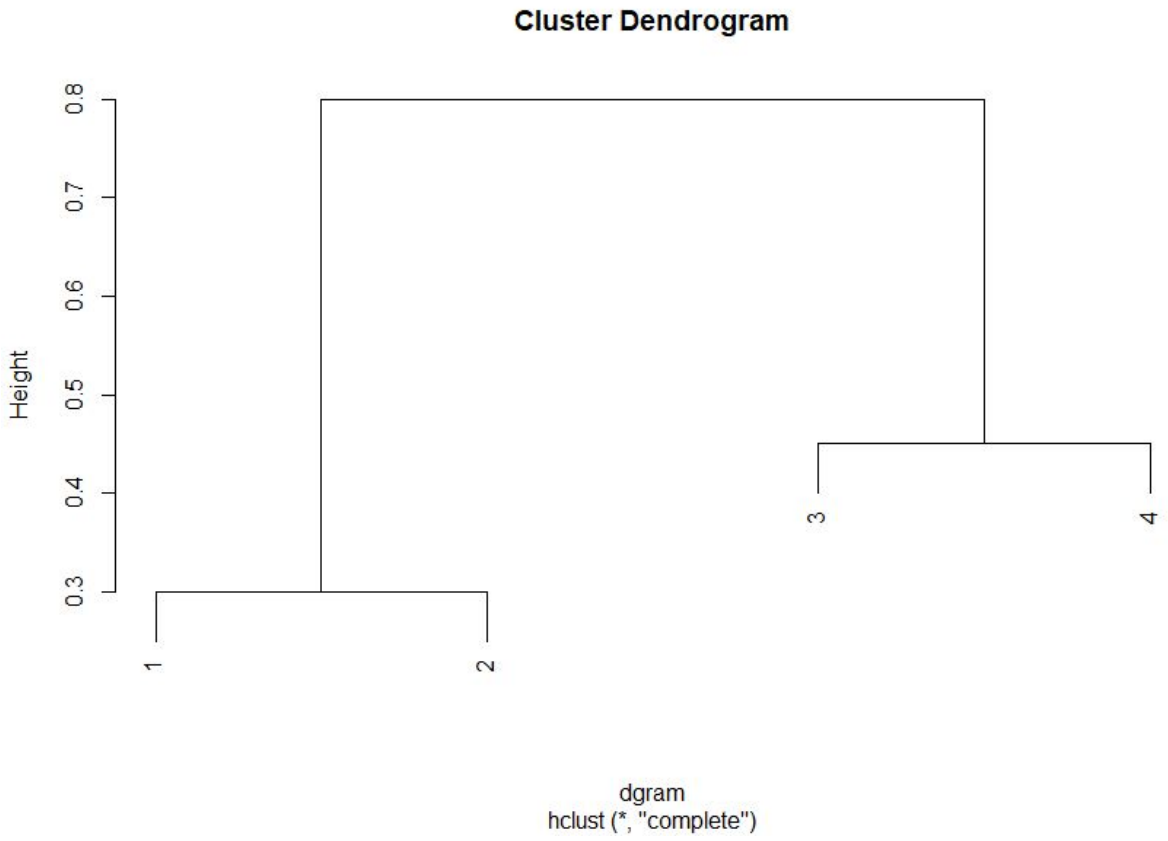
4. (Ch. 10, Question 2)
   Suppose that we have four observations, for which we compute a dissimilarity matrix, given by

$$\begin{bmatrix} & 0.3 & 0.4 & 0.7 \\ 0.3 & & 0.5 & 0.8 \\ 0.4 & 0.5 & & 0.45 \\ 0.7 & 0.8 & 0.45 & \end{bmatrix}$$

   For instance, the dissimilarity between the first and second observations is 0.3, and the dissimilarity between the second and fourth observations is 0.8.

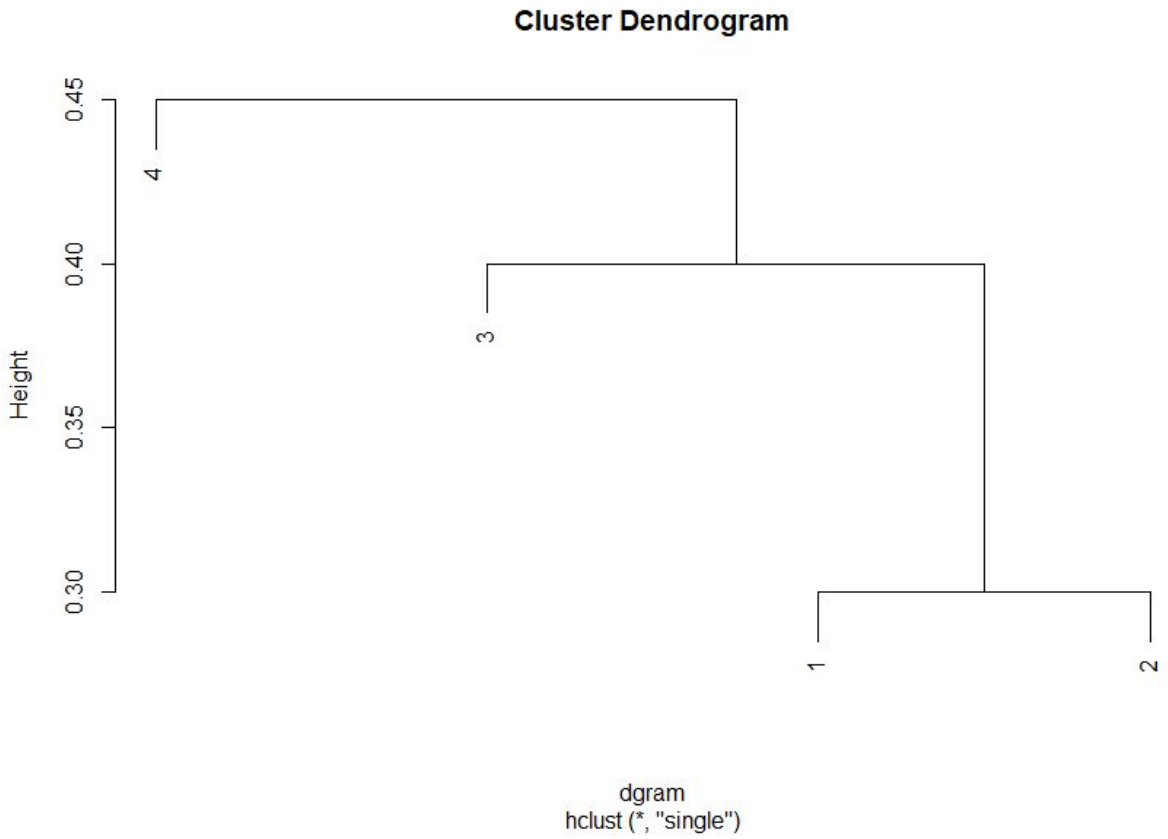
   a. On the basis of this dissimilarity matrix, sketch the dendrogram that results from hierarchically clustering these four observations using complete linkage. Be sure to indicate on the plot the height at which each fusion occurs, as well as the observations corresponding to each leaf in the dendrogram.

```
> dgram = as.dist(matrix(c(0, 0.3, 0.4, 0.7,
+                          0.3, 0, 0.5, 0.8,
+                          0.4, 0.5, 0.0, 0.45,
+                          0.7, 0.8, 0.45, 0.0), nrow=4))
> plot(hclust(dgram, method="complete"))
>
```



b. Repeat (a), this time using single linkage clustering.

```
> plot(hclust(dgram, method="single"))
>
```



c. Suppose that we cut the dendogram obtained in (a) such that two clusters result. Which observations are in each cluster?

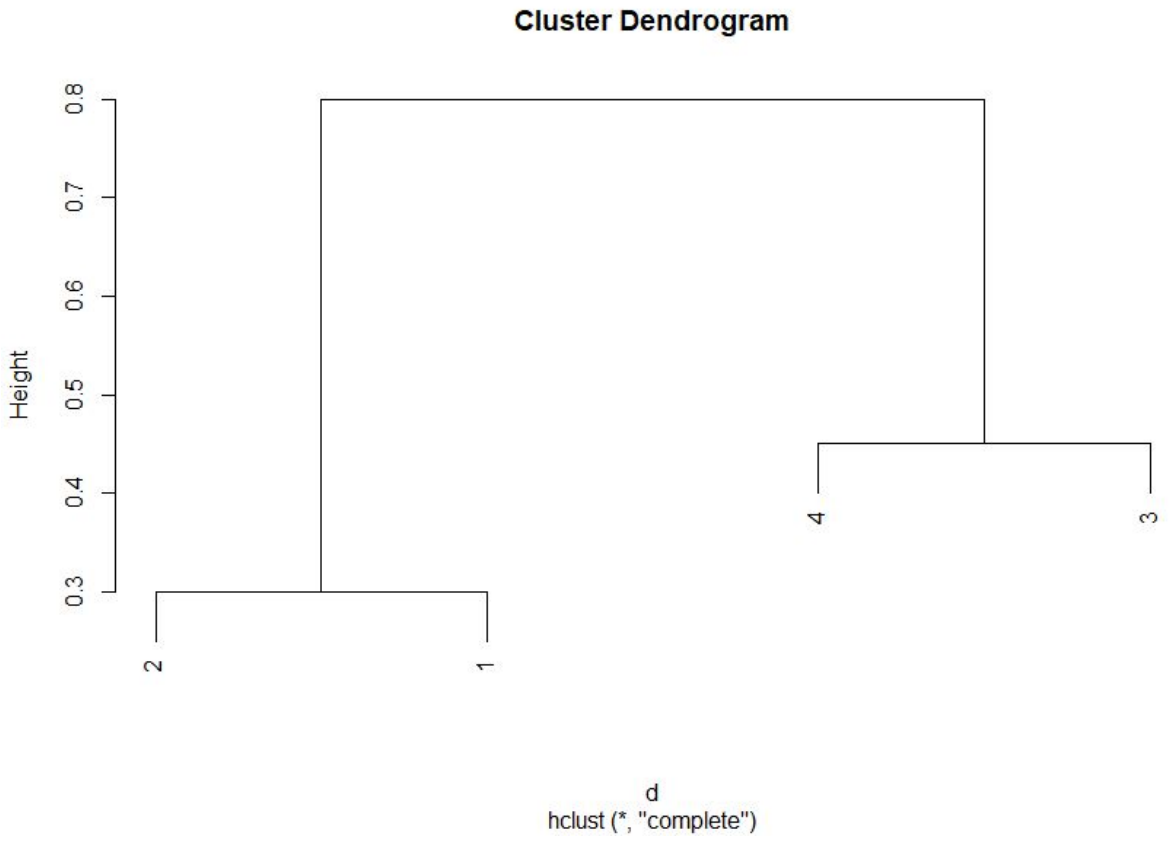
The resulting clusters would be (1,2) and (3,4).

d. Suppose that we cut the dendogram obtained in (b) such that two clusters result. Which observations are in each cluster?

The resulting clusters would be (1,2,3) and (4).

e. It is mentioned in the chapter that at each fusion in the dendrogram, the position of the two clusters being fused can be swapped without changing the meaning of the dendrogram. Draw a dendrogram that is equivalent to the dendrogram in (a), for which two or more of the leaves are repositioned, but for which the meaning of the dendrogram is the same.

```
> plot(hclust(d, method="complete"), labels=c(2,1,4,3))
>
```

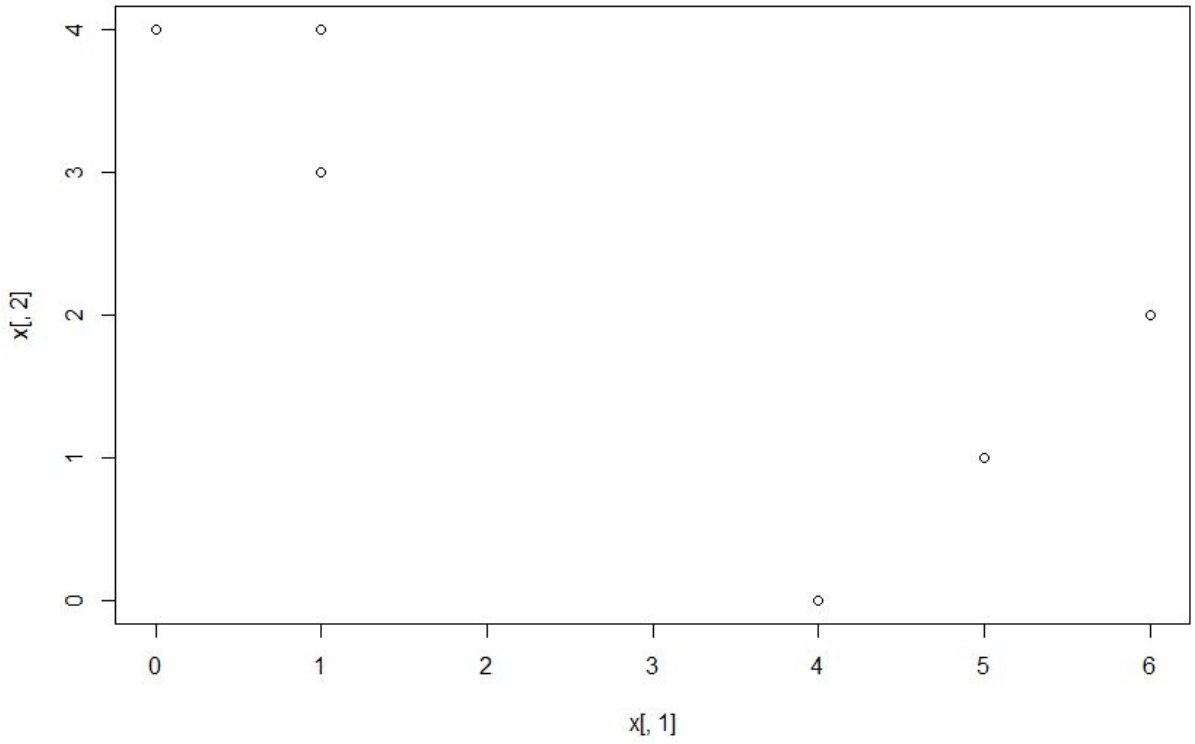


5. (Ch. 10, Question 3)
   In this problem, you will perform K-means clustering manually, with K = 2, on a small example with n = 6 observations and p = 2 features. The observations are as follows.

Obs.	X_1	X_2
1	1	4
2	1	3
3	0	4
4	5	1
5	6	2
6	4	0

   a. Plot the observations.

```
> set.seed(1)
>
> x = cbind(c(1, 1, 0, 5, 6, 4), c(4, 3, 4, 1, 2, 0))
>
> plot(x[,1], x[,2])
>
```

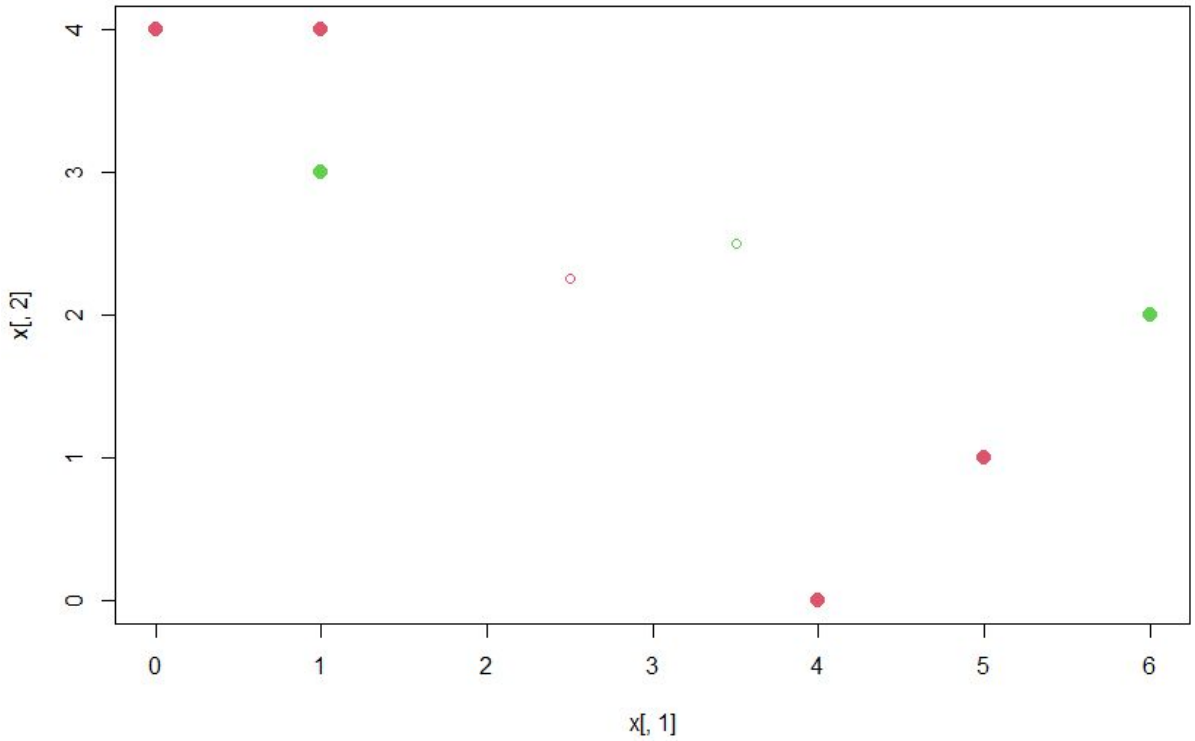


b. Randomly assign a cluster label to each observation. You can use the sample() command in R to do this. Report the cluster labels for each observation.

```
> labels = sample(2, nrow(x), replace=T)
> labels
[1] 1 2 1 1 2 1
>
```

c. Compute the centroid for each cluster.

```
> centroid1 = c(mean(x[labels==1, 1]), mean(x[labels==1, 2]))
> centroid2 = c(mean(x[labels==2, 1]), mean(x[labels==2, 2]))
>
> plot(x[,1], x[,2], col=(labels+1), pch=20, cex=2)
> points(centroid1[1], centroid1[2], col=2)
> points(centroid2[1], centroid2[2], col=3)
>
```



d. Assign each observation to the centroid to which it is closest, in terms of Euclidean distance. Report the cluster labels for each observation.
e. Repeat (c) and (d) until the answers obtained stop changing.
f. In your plot from (a), color the observations according to the cluster labels obtained.

6. (Ch. 10, Question 4)
Suppose that for a particular data set, we perform hierarchical clustering using single linkage and using complete linkage. We obtain two dendrograms.
    a. At a certain point on the single linkage dendrogram, the clusters {1, 2, 3} and {4, 5} fuse. On the complete linkage dendrogram, the clusters {1, 2, 3} and {4, 5} also fuse at a certain point. Which fusion will occur higher on the tree, or will they fuse at the same height, or is there not enough information to tell?

There is not enough information to tell because we do not know how the maximal intercluster dissimilarity and minimial intercluster dissimilarity relate. If they are equal, then they would fuse at the same height. Otherwise, the single linkage dendogram would fuse at a lower height on the tree.

    b. At a certain point on the single linkage dendrogram, the clusters {5} and {6} fuse. On the complete linkage dendrogram, the clusters {5} and {6} also fuse at a certain point. Which fusion will occur higher on the tree, or will they fuse at the same height, or is there not enough information to tell?

The clusters would fuse at the same height because, in general, linkage does not affect leaf-to-leaf fusion.

7. (Ch. 10, Question 5)
   In words, describe the results that you would expect if you performed K-means clustering of the eight shoppers in Figure 10.14, on the basis of their sock and computer purchases, with K = 2. Give three answers, one for each of the variable scalings displayed. Explain.

Result 1: (3,4,6,8) and (1,2,7,8). The first cluster is less socks and computers, while the second is more socks and computers.

Result 2: (5,6,7,8) and (1,2,3,4). The first cluster is for computer purchases, while the second is for no computer purchases.

Result 3: (5,6,7,8) and (1,2,3,4). Similar to result 2, the first cluster is for computer purchases, while the second is for no computer purchases. However, in this result, the distance for the computer dimension is greater than the distance for the socks dimension.

Applied

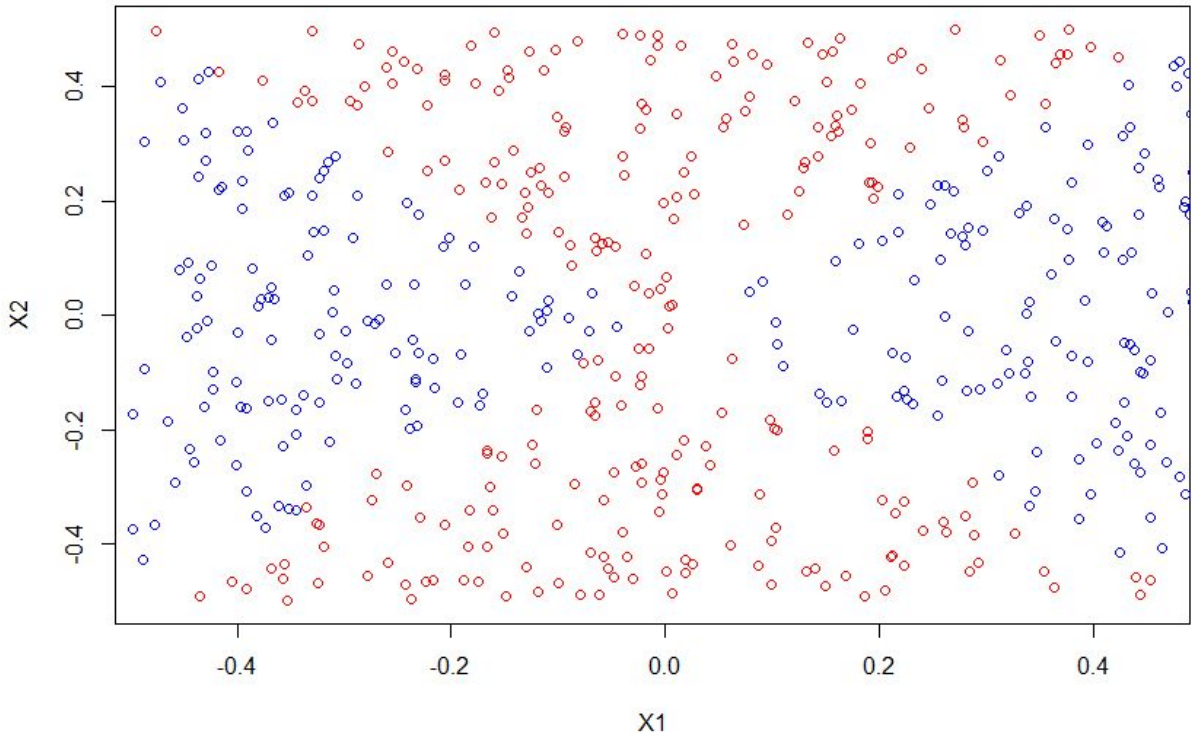
8. (Ch. 9, Question 5)
   We have seen that we can fit an SVM with a non-linear kernel in order to perform classification using a non-linear decision boundary. We will now see that we can also obtain a non-linear decision boundary by performing logistic regression using non-linear transformations of the features.
   a. Generate a data set with n = 500 and p = 2, such that the observations belong to two classes with a quadratic decision boundary between them. For instance, you can do this as follows:

```
> x1=runif(500)-0.5
> x2=runif(500)-0.5
> y=1*(x1^2-x2^2 > 0)
```

```
> set.seed(1)
> x1 = runif(500) - 0.5
> x2 = runif(500) - 0.5
> y = 1 * (x1^2 - x2^2 > 0)
>
```

   b. Plot the observations, colored according to their class labels. Your plot should display X1 on the x-axis, and X2 on the yaxis.

```
> plot(x1[y == 0], x2[y == 0], col = "red", xlab = "X1", ylab = "X2")
> points(x1[y == 1], x2[y == 1], col = "blue")
>
```

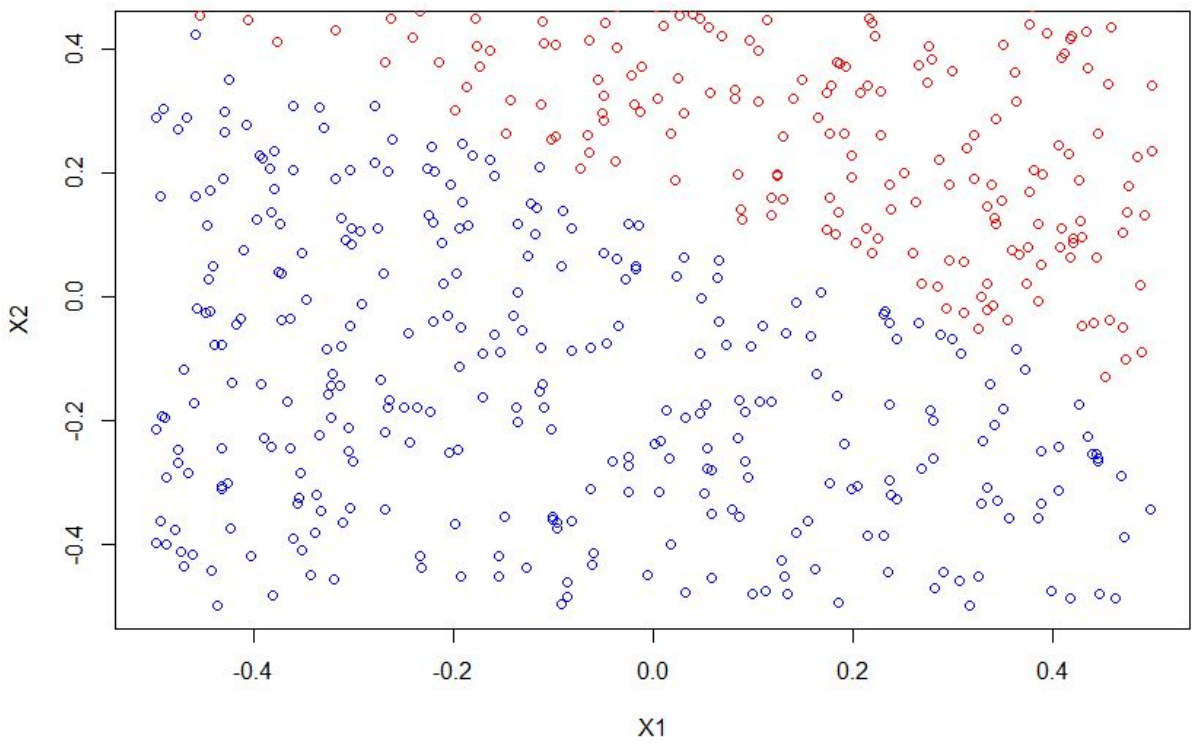


c. Fit a logistic regression model to the data, using X1 and X2 as predictors.

```
> lm.fit = glm(y ~ x1 + x2, family = binomial)
> summary(lm.fit)

Call:
glm(formula = y ~ x1 + x2, family = binomial)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-1.179  -1.139  -1.112   1.206   1.257

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.087260   0.089579  -0.974    0.330
x1           0.196199   0.316864   0.619    0.536
x2          -0.002854   0.305712  -0.009    0.993

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 692.18  on 499  degrees of freedom
Residual deviance: 691.79  on 497  degrees of freedom
AIC: 697.79

Number of Fisher Scoring iterations: 3

> |
```

d. Apply this model to the training data in order to obtain a predicted class label for each training observation. Plot the observations, colored according to the predicted class labels. The decision boundary should be linear.

```
> data = data.frame(x1 = x1, x2 = x2, y = y)
> lm.prob = predict(lm.fit, data, type = "response")
> lm.pred = ifelse(lm.prob > 0.52, 1, 0)
> data.pos = data[lm.pred == 1, ]
> data.neg = data[lm.pred == 0, ]
> plot(data.pos$x1, data.pos$x2, col = "blue", xlab = "X1", ylab = "X2")
> points(data.neg$x1, data.neg$x2, col = "red")
> |
```



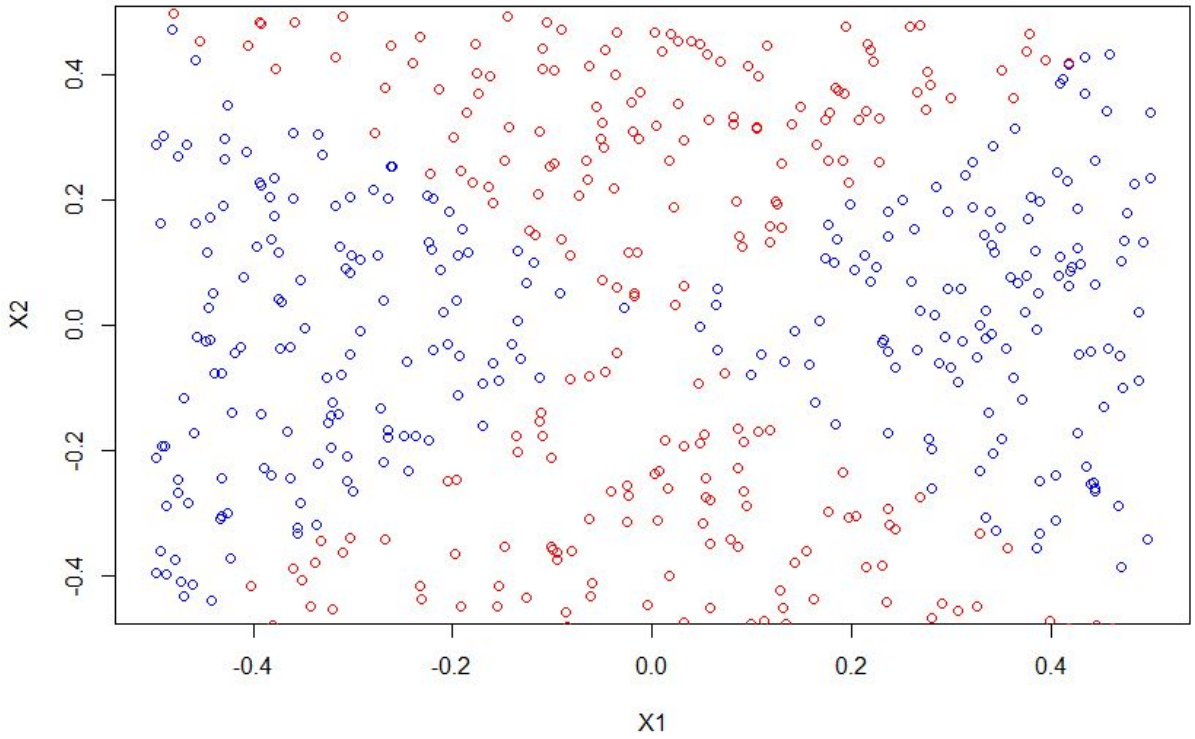
The decision boundary is clearly linear.

e. Now fit a logistic regression model to the data using non-linear functions of X1 and X2 as predictors (e.g. X2 1 , X1×X2, log(X2), and so forth).

```
> lm.fit = glm(y ~ poly(x1, 2) + poly(x2, 2) + I(x1 * x2), data = data, family = binomial)
Warning messages:
1: glm.fit: algorithm did not converge
2: glm.fit: fitted probabilities numerically 0 or 1 occurred
> summary(lm.fit)

Call:
glm(formula = y ~ poly(x1, 2) + poly(x2, 2) + I(x1 * x2), family = binomial,
    data = data)

Deviance Residuals:
       Min          1Q      Median          3Q         Max
-1.389e-03  -2.000e-08   2.000e-08   2.000e-08   1.625e-03

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)     130.0     2858.5   0.045    0.964
poly(x1, 2)1  -3834.5    86056.3  -0.045    0.964
poly(x1, 2)2  43990.0   958266.0   0.046    0.963
poly(x2, 2)1    695.0    38234.3   0.018    0.985
poly(x2, 2)2 -43627.5   959282.2  -0.045    0.964
I(x1 * x2)      584.1    23099.8   0.025    0.980

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 6.9110e+02  on 499  degrees of freedom
Residual deviance: 8.5033e-06  on 494  degrees of freedom
AIC: 12

Number of Fisher Scoring iterations: 25

>
```

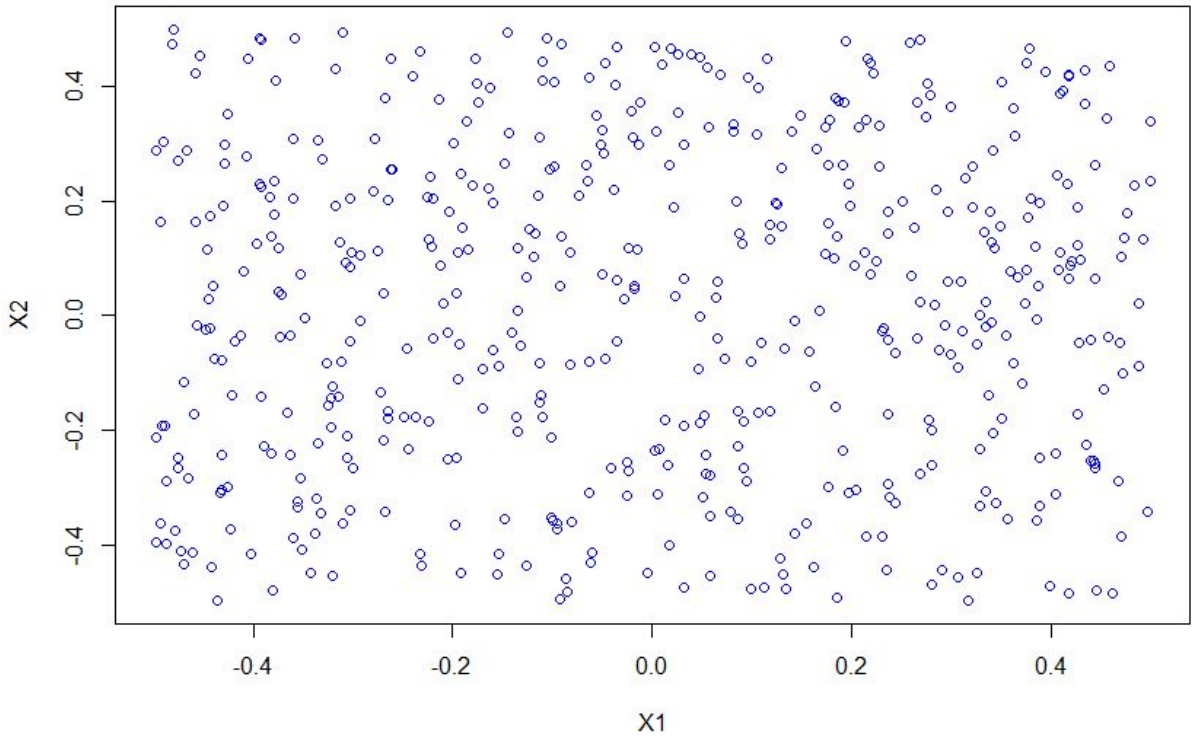
f. Apply this model to the training data in order to obtain a predicted class label for each training observation. Plot the observations, colored according to the predicted class labels. The decision boundary should be obviously non-linear. If it is not, then repeat (a)-(e) until you come up with an example in which the predicted class labels are obviously non-linear.

```
> lm.prob = predict(lm.fit, data, type = "response")
> lm.pred = ifelse(lm.prob > 0.5, 1, 0)
> data.pos = data[lm.pred == 1, ]
> data.neg = data[lm.pred == 0, ]
> plot(data.pos$x1, data.pos$x2, col = "blue", xlab = "X1", ylab = "X2")
> points(data.neg$x1, data.neg$x2, col = "red")
>
```



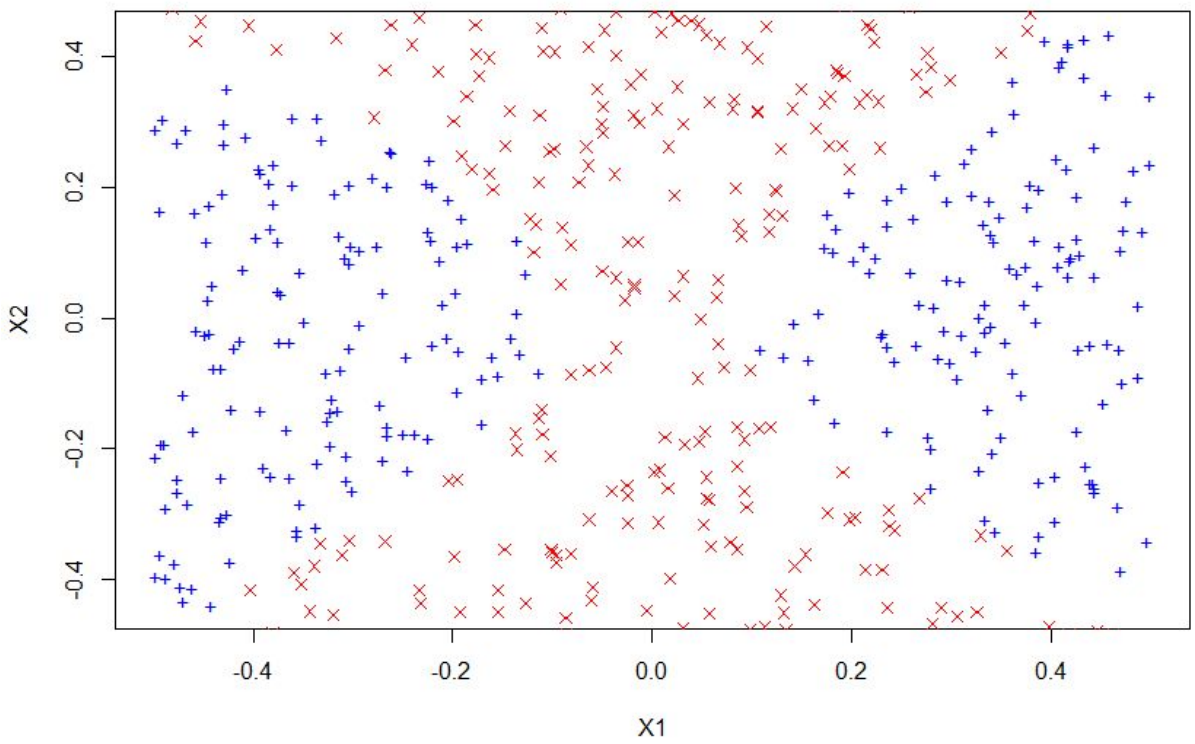
g. Fit a support vector classifier to the data with X1 and X2 as predictors. Obtain a class prediction for each training observation. Plot the observations, colored according to the predicted class labels.

```
> svm.fit = svm(as.factor(y) ~ x1 + x2, data, kernel = "linear", cost = 0.1)
> svm.pred = predict(svm.fit, data)
> data.pos = data[svm.pred == 1, ]
> data.neg = data[svm.pred == 0, ]
> plot(data.pos$x1, data.pos$x2, col = "blue", xlab = "X1", ylab = "X2")
> points(data.neg$x1, data.neg$x2, col = "red")
>
```



h. Fit a SVM using a non-linear kernel to the data. Obtain a class prediction for each training observation. Plot the observations, colored according to the predicted class labels.

```
> svm.fit = svm(as.factor(y) ~ x1 + x2, data, gamma = 1)
> svm.pred = predict(svm.fit, data)
> data.pos = data[svm.pred == 1, ]
> data.neg = data[svm.pred == 0, ]
> plot(data.pos$x1, data.pos$x2, col = "blue", xlab = "X1", ylab = "X2", pch = "+")
> points(data.neg$x1, data.neg$x2, col = "red", pch = 4)
>
```



i. Comment on your results.

It appears that SVMs with a non-linear kernel are adept at finding non-linear boundaries. However, both linear regression with non-interactions and SVMs with linear kernels did not perform well when trying to find the decision boundary.

9. (Ch. 9, Question 7)
   In this problem, you will use support vector approaches in order to predict whether a given car gets high or low gas mileage based on the Auto data set.
   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)
> gas.med = median(Auto$mpg)
> new.var = ifelse(Auto$mpg > gas.med, 1, 0)
> Auto$mpglevel = as.factor(new.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.

```
> set.seed(2500)
> tune.out = tune(svm, mpglevel ~ ., data = Auto, kernel = "linear", ranges = list(cost = c(0.01, 0.1, 1, 5, 10, 100)))
> summary(tune.out)

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 0.07653846 0.03823198
2 1e-01 0.04333333 0.02101243
3 1e+00 0.01282051 0.01813094
4 5e+00 0.02044872 0.01619554
5 1e+01 0.02301282 0.02244393
6 1e+02 0.03320513 0.02116312

> |
```

Cost = 1 appears to minimize the error for cross validation the most.

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(1500)
> tune.out = tune(svm, mpglevel ~ ., data = Auto, kernel = "polynomial", ranges = list(cost = c(0.1, 1, 5, 10), degree = c(2,
 3, 4)))
> summary(tune.out)

Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 cost degree
   10      2

- best performance: 0.5380769

- Detailed performance results:
   cost degree     error dispersion
1   0.1      2 0.5714103 0.04950555
2   1.0      2 0.5714103 0.04950555
3   5.0      2 0.5714103 0.04950555
4  10.0      2 0.5380769 0.09784701
5   0.1      3 0.5714103 0.04950555
6   1.0      3 0.5714103 0.04950555
7   5.0      3 0.5714103 0.04950555
8  10.0      3 0.5714103 0.04950555
9   0.1      4 0.5714103 0.04950555
10  1.0      4 0.5714103 0.04950555
11  5.0      4 0.5714103 0.04950555
12 10.0      4 0.5714103 0.04950555

> |
```

Cost = 10 and a degree of 10 appears to minimize the error for the polynomial basis kernel.

```
> set.seed(2000)
> tune.out = tune(svm, mpglevel ~ ., data = Auto, kernel = "radial", ranges = list(cost = c(0.1, 1, 5, 10), gamma = c(0.01, 0.
1, 1, 5, 10, 100)))
> summary(tune.out)

Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 cost gamma
   10   0.1

- best performance: 0.03320513

- Detailed performance results:
   cost gamma      error dispersion
1   0.1 1e-02 0.08929487 0.07175801
2   1.0 1e-02 0.07153846 0.06267003
3   5.0 1e-02 0.05108974 0.05924783
4  10.0 1e-02 0.03326923 0.05276735
5   0.1 1e-01 0.07660256 0.06735702
6   1.0 1e-01 0.05878205 0.05682158
7   5.0 1e-01 0.03576923 0.04386727
8  10.0 1e-01 0.03320513 0.04360838
9   0.1 1e+00 0.55371795 0.04009904
10  1.0 1e+00 0.06391026 0.05709732
11  5.0 1e+00 0.06384615 0.05310437
12 10.0 1e+00 0.06384615 0.05310437
13  0.1 5e+00 0.55371795 0.04009904
14  1.0 5e+00 0.46942308 0.07369041
15  5.0 5e+00 0.46173077 0.08386530
16 10.0 5e+00 0.46173077 0.08386530
17  0.1 1e+01 0.55371795 0.04009904
18  1.0 1e+01 0.50782051 0.06557970
19  5.0 1e+01 0.49506410 0.05681354
20 10.0 1e+01 0.49506410 0.05681354
21  0.1 1e+02 0.55371795 0.04009904
22  1.0 1e+02 0.55371795 0.04009904
23  5.0 1e+02 0.55371795 0.04009904
24 10.0 1e+02 0.55371795 0.04009904

>
```

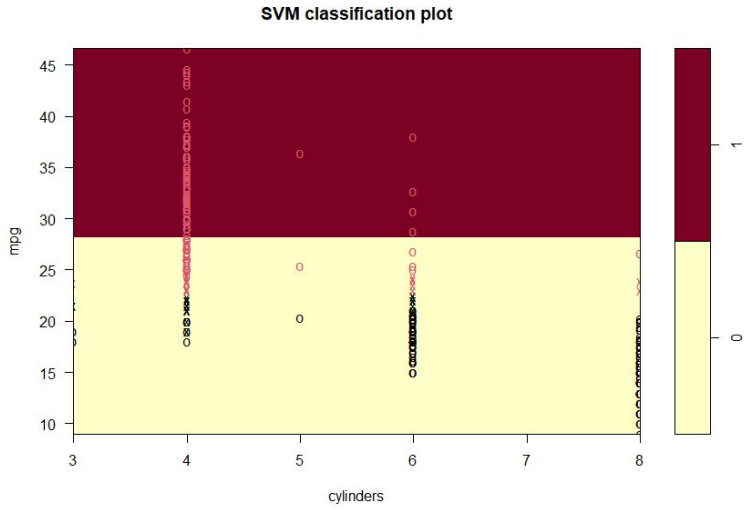
Cost = 10 and a gamma of 0.1 appears to minimize the error for the radial basis kernel.

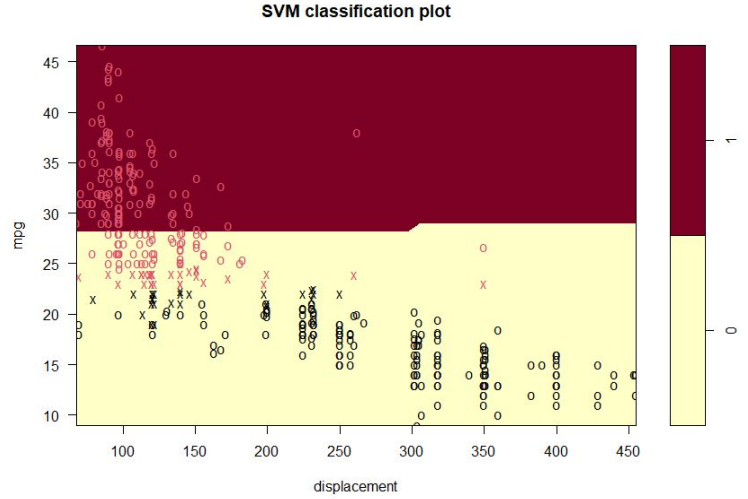
d. Make some plots to back up your assertions in (b) and (c).

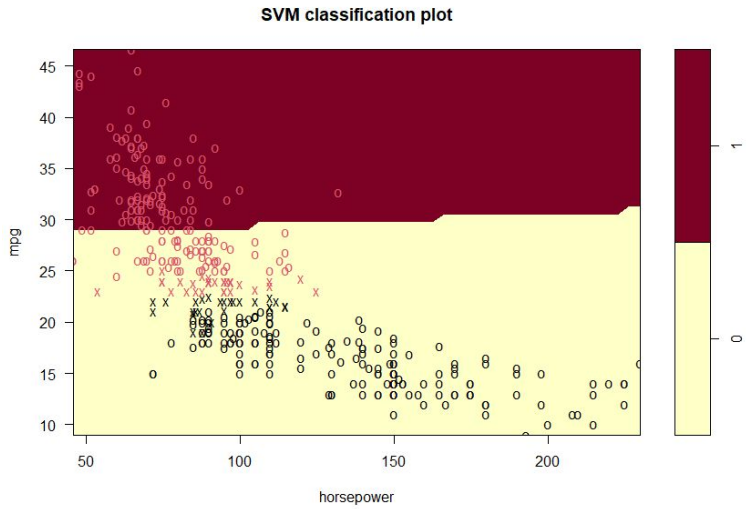
```
> svm.linear = svm(mpglevel ~ ., data = Auto, kernel = "linear", cost = 1)
> svm.poly = svm(mpglevel ~ ., data = Auto, kernel = "polynomial", cost = 10,
+                 degree = 2)
> svm.radial = svm(mpglevel ~ ., data = Auto, kernel = "radial", cost = 10, gamma = 0.01)
>
> plotpairs = function(fit)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

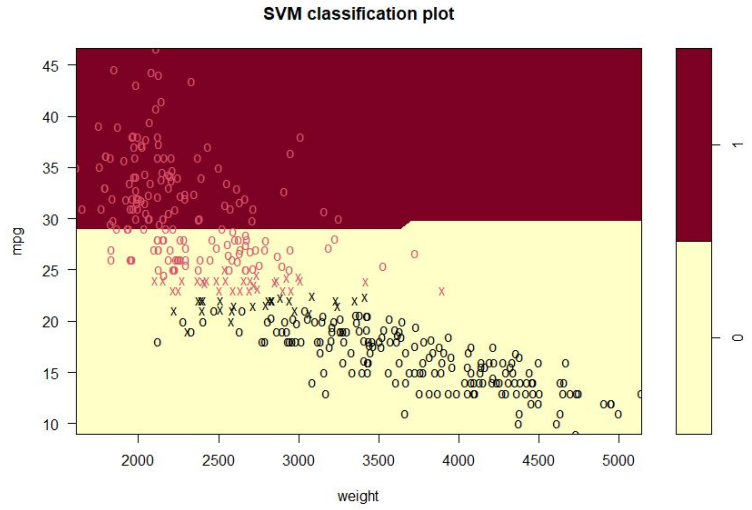
Linear:

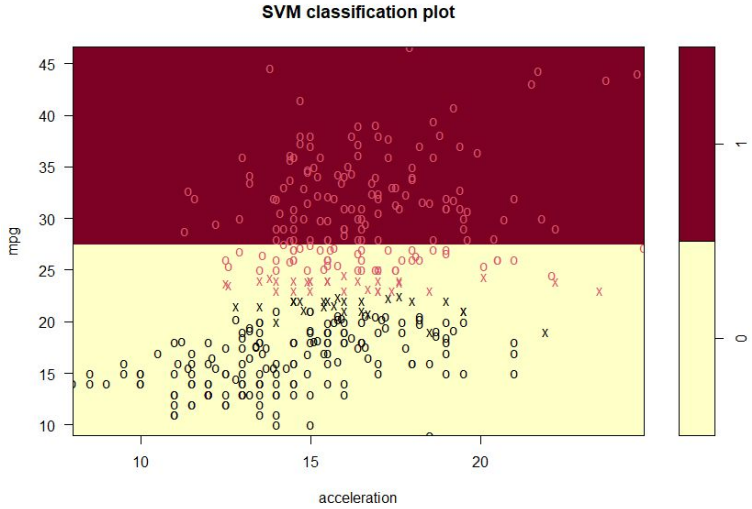
```
> plotpairs(svm.linear)
```

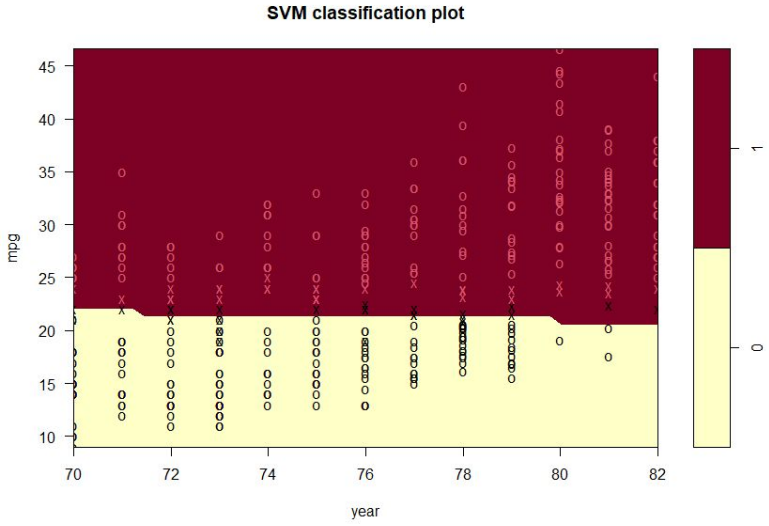

SVM classification plot
mpg
cylinders
45
40
35
30
25
20
15
10
3
4
5
6
7
8
1
0

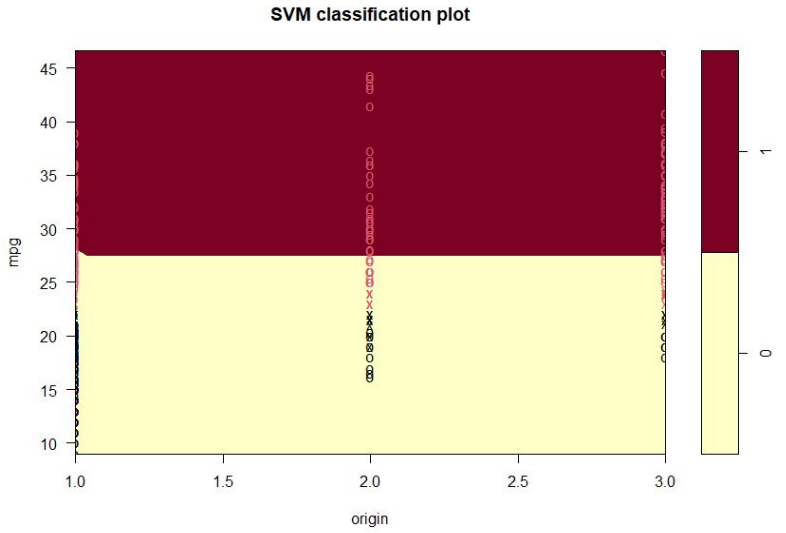

SVM classification plot
mpg
displacement
45
40
35
30
25
20
15
10
100
150
200
250
300
350
400
450
1
0


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
2500
3000
3500
4000
4500
5000
1
0

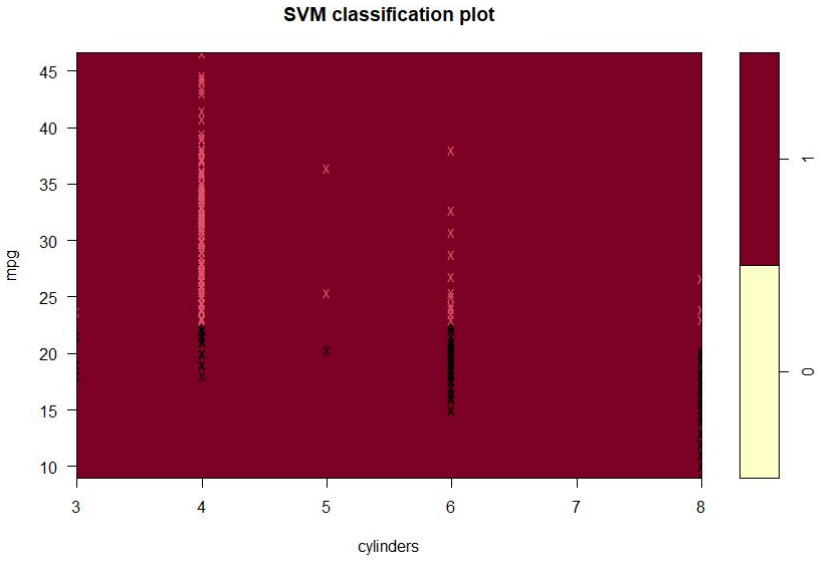

SVM classification plot
mpg
acceleration
45
40
35
30
25
20
15
10
10
15
20
1
0

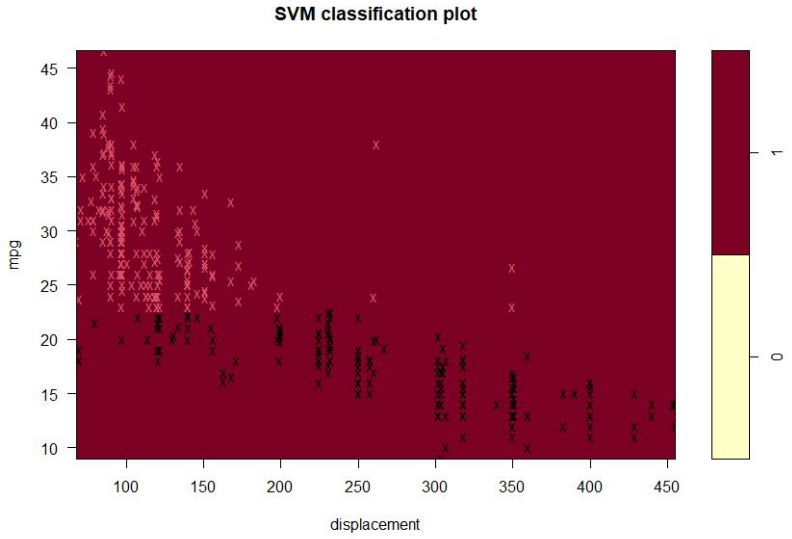

SVM classification plot
mpg
year
45
40
35
30
25
20
15
10
70
72
74
76
78
80
82
1
0

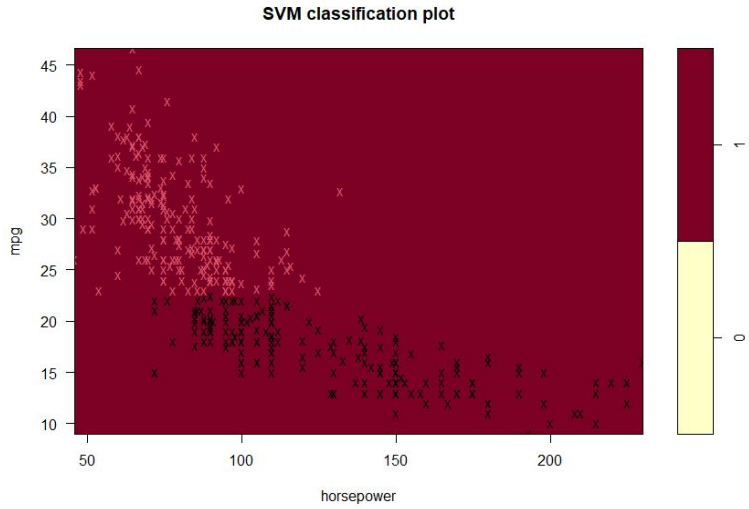


Polynomial:

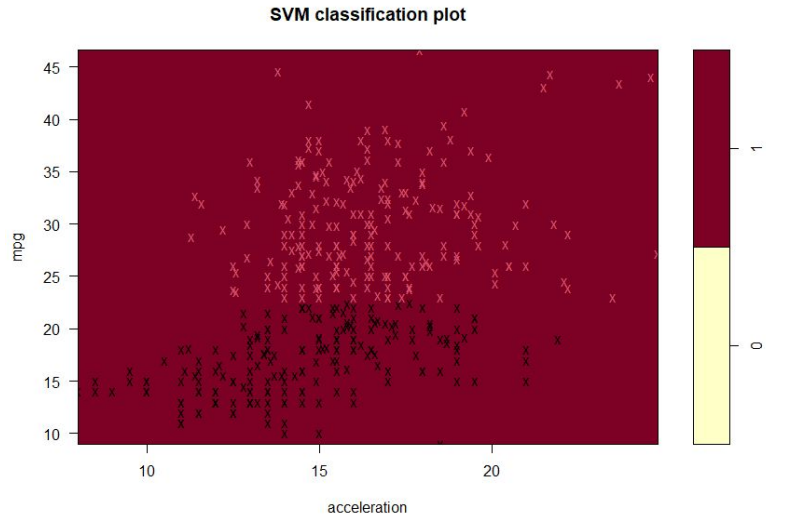
```
> plotpairs(svm.poly)
```

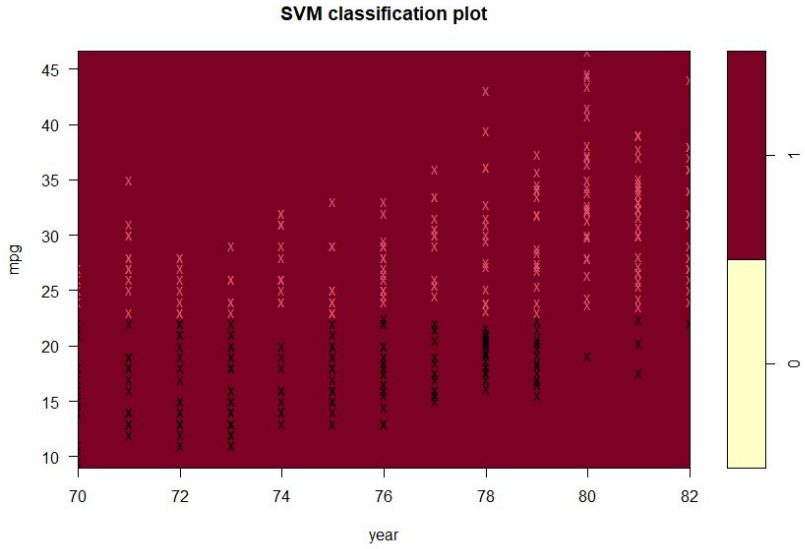


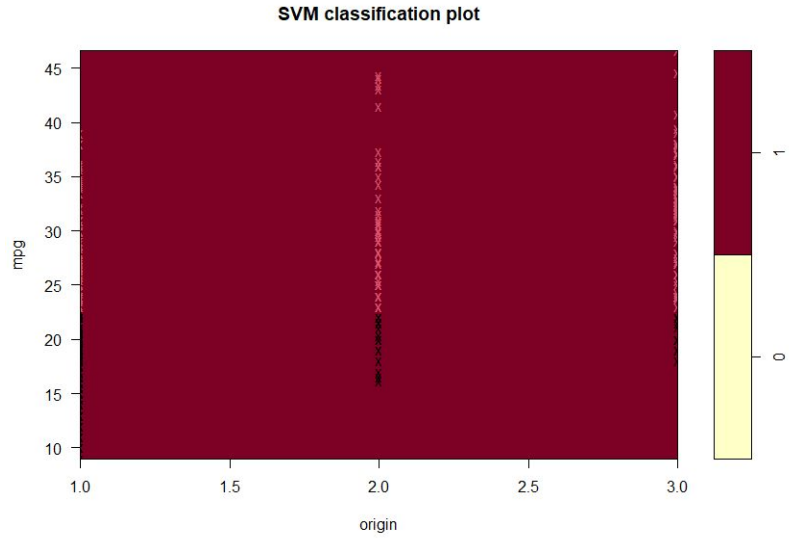

SVM classification plot
mpg
displacement
10
15
20
25
30
35
40
45
100
150
200
250
300
350
400
450
0
1


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


SVM classification plot
mpg
weight
10
15
20
25
30
35
40
45
2000
2500
3000
3500
4000
4500
5000
0
1

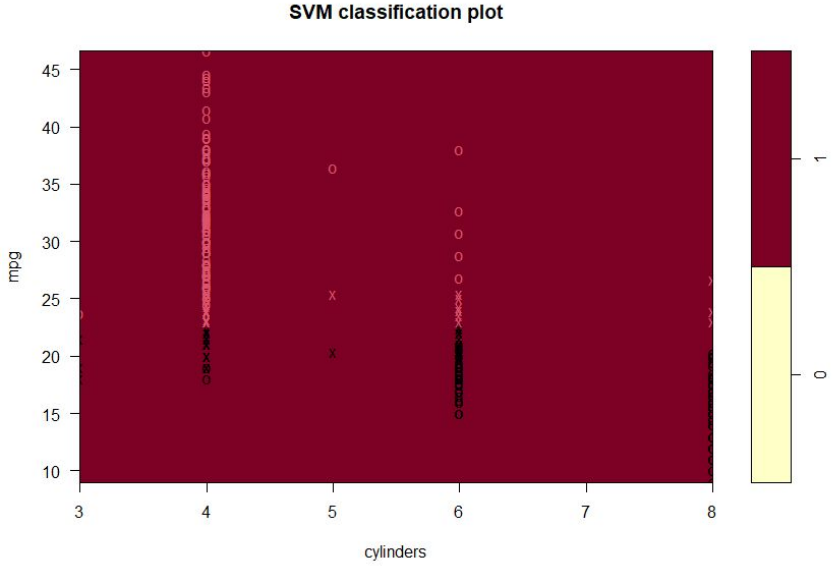

SVM classification plot
mpg
acceleration
45
40
35
30
25
20
15
10
10
15
20
1
0


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

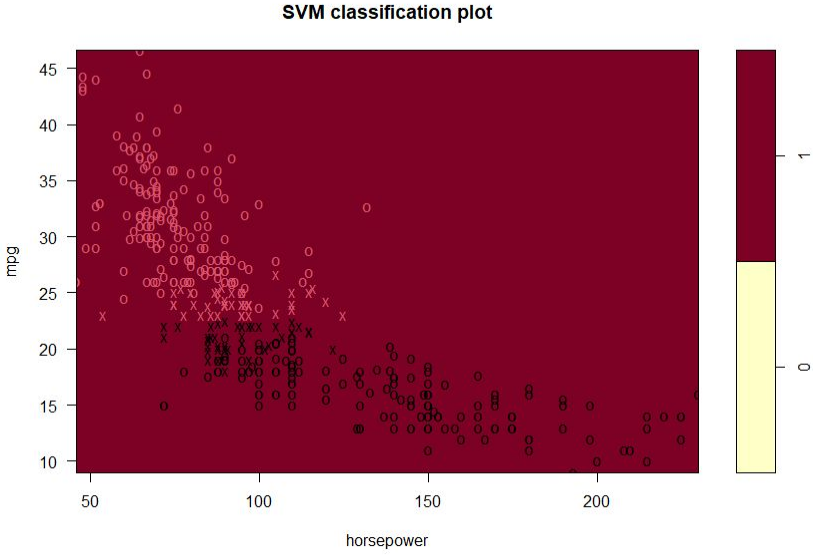


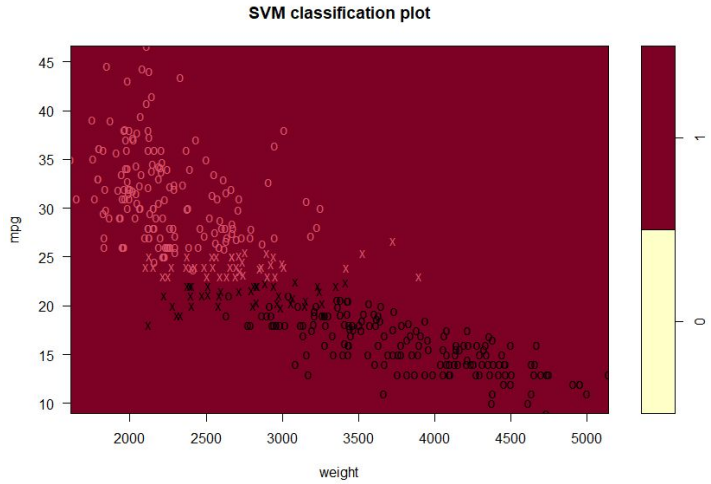
Radial:

```
> plotpairs(svm.radial)
```

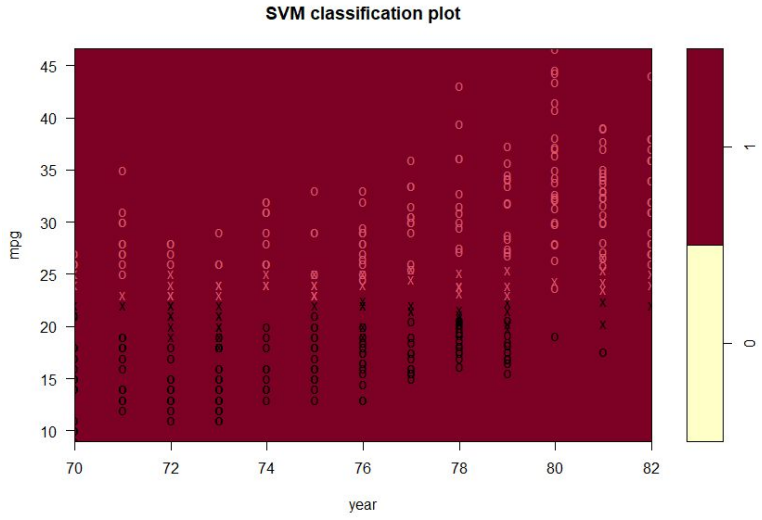


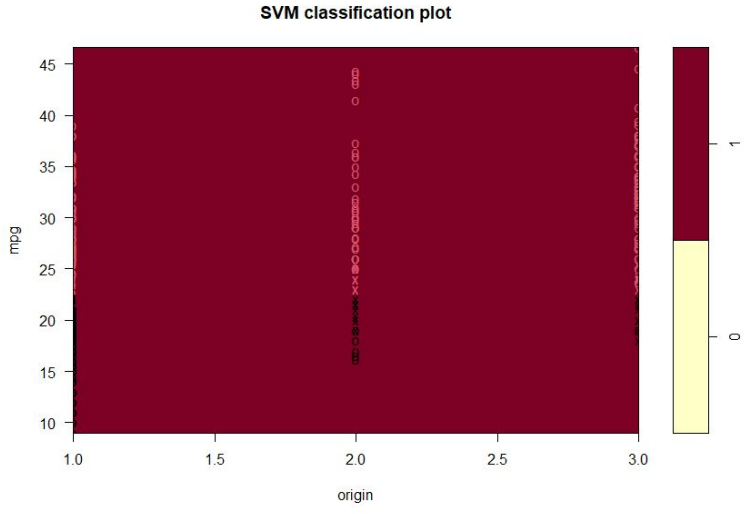

SVM classification plot
mpg
displacement
45
40
35
30
25
20
15
10
100
150
200
250
300
350
400
450
1
0


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
2500
3000
3500
4000
4500
5000
1
0


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


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

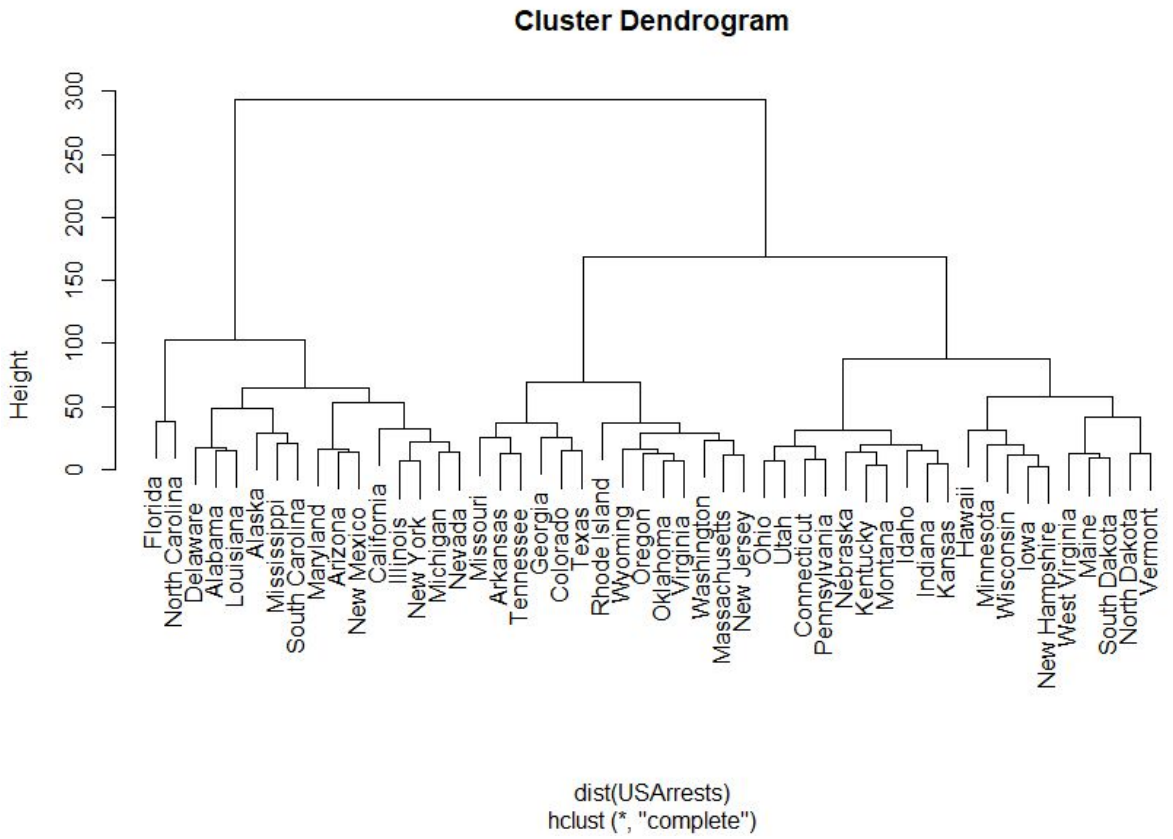

SVM classification plot
mpg
origin
45
40
35
30
25
20
15
10
1.0
1.5
2.0
2.5
3.0
1
0

10. (Ch. 10, Question 9)

Consider the USArrests data. We will now perform hierarchical clustering on the states.

a. Using hierarchical clustering with complete linkage and Euclidean distance, cluster the states.

```
> library(ISLR)
>
> set.seed(1)
> hc.complete = hclust(dist(USArrests), method="complete")
> plot(hc.complete)
>
```

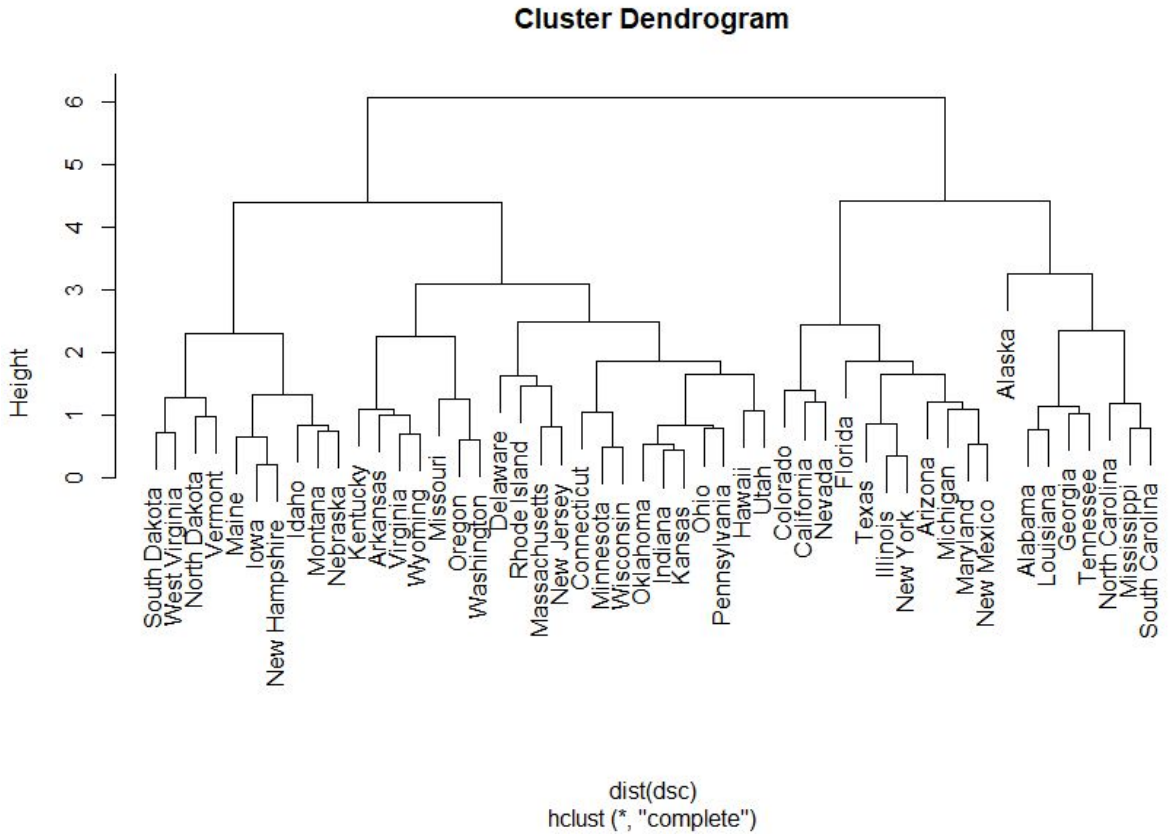


b. Cut the dendrogram at a height that results in three distinct clusters. Which states belong to which clusters?

```
> cutree(hc.complete, 3)
       Alabama         Alaska        Arizona       Arkansas     California       Colorado    Connecticut       Delaware
             1              1              1              2              1              2              3              1
       Florida        Georgia         Hawaii          Idaho       Illinois        Indiana           Iowa         Kansas
             1              2              3              3              1              3              3              3
      Kentucky      Louisiana          Maine       Maryland  Massachusetts       Michigan      Minnesota    Mississippi
             3              1              3              1              2              1              3              1
      Missouri        Montana       Nebraska         Nevada  New Hampshire     New Jersey     New Mexico       New York
             2              3              3              1              3              2              1              1
North Carolina   North Dakota           Ohio       Oklahoma         Oregon   Pennsylvania   Rhode Island South Carolina
             1              3              3              2              2              3              2              1
  South Dakota      Tennessee          Texas           Utah        Vermont       Virginia     Washington  West Virginia
             3              2              2              3              3              2              2              3
     Wisconsin        Wyoming
             3              2
>
```

c. Hierarchically cluster the states using complete linkage and Euclidean distance, after scaling the variables to have standard deviation one.

```
> dsc = scale(USArrests)
> hc.s.complete = hclust(dist(dsc), method="complete")
> plot(hc.s.complete)
>
```



d. What effect does scaling the variables have on the hierarchical clustering obtained? In your opinion, should the variables be scaled before the inter-observation dissimilarities are computed? Provide a justification for your answer.

```
> cutree(hc.s.complete, 3)
       Alabama         Alaska        Arizona       Arkansas     California       Colorado    Connecticut       Delaware
             1              1              2              3              2              2              3              3
       Florida        Georgia         Hawaii          Idaho       Illinois        Indiana           Iowa         Kansas
             2              1              3              3              2              3              3              3
      Kentucky      Louisiana          Maine       Maryland  Massachusetts       Michigan      Minnesota    Mississippi
             3              1              3              2              3              2              3              1
      Missouri        Montana       Nebraska         Nevada New Hampshire     New Jersey     New Mexico       New York
             3              3              3              2              3              3              2              2
North Carolina   North Dakota           Ohio       Oklahoma         Oregon   Pennsylvania   Rhode Island South Carolina
             1              3              3              3              3              3              3              1
  South Dakota      Tennessee          Texas           Utah        Vermont       Virginia     Washington  West Virginia
             3              1              2              3              3              3              3              3
     Wisconsin        Wyoming
             3              3
>
```

```
> table(cutree(hc.s.complete, 3), cutree(hc.complete, 3))

    1  2  3
1   6  2  0
2   9  2  0
3   1 10 20
>
```

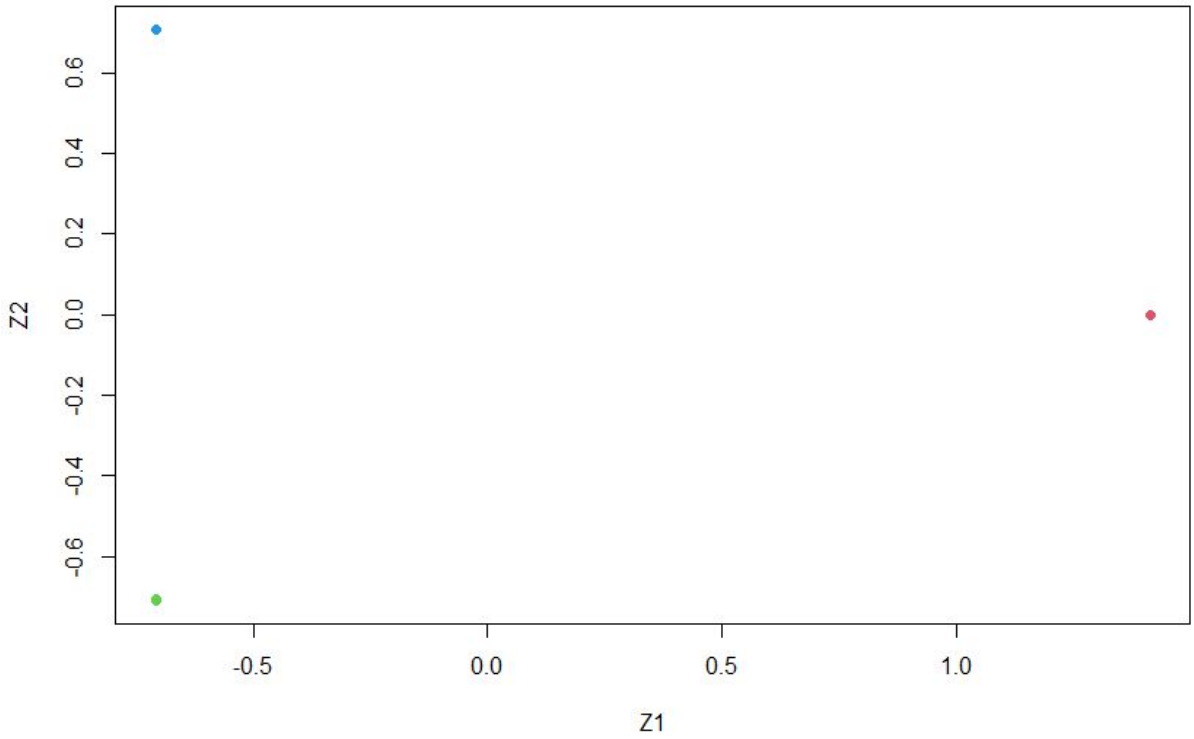
Scaling the variables affects the max height of the clusters obtained. In my opinion, the variables should be scaled prior because of the difference in units within the data.

11. (Ch. 10, Question 10)
    In this problem, you will generate simulated data, and then perform PCA and K-means clustering on the data.
    a. Generate a simulated data set with 20 observations in each of three classes (i.e. 60 observations total), and 50 variables. Hint: There are a number of functions in R that you can use to generate data. One example is the rnorm() function; runif() is another option. Be sure to add a mean shift to the observations in each class so that there are three distinct classes.

```
> set.seed(1)
>
> x = matrix(rnorm(20 * 3 * 50, mean = 0, sd = 0.001), ncol = 50)
> x[1:20, 2] = 1
> x[21:40, 1] = 2
> x[21:40, 2] = 2
> x[41:60, 1] = 1
>
```

    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.out = prcomp(x)
> plot(pca.out$x[,1:2], col=2:4, xlab="Z1", ylab="Z2", pch=19)
>
```



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? Hint: You can use the table() function in R to compare the true class labels to the class labels obtained by clustering. Be careful how you interpret the results: K-means clustering will arbitrarily number the clusters, so you cannot simply check whether the true class labels and clustering labels are the same.

```
> km.out = kmeans(x, 3, nstart=20)
> table(km.out$cluster, c(rep(1,20), rep(2,20), rep(3,20)))

    1  2  3
1   0 20  0
2   0  0 20
3  20  0  0
>
```

The clusters compare very well to the true class labels.

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

```
> km.out = kmeans(x, 2, nstart=20)
> table(km.out$cluster, c(rep(1,20), rep(2,20), rep(3,20)))

    1  2  3
1  20  0 20
2   0 20  0
>
```

The observations of, what would have been the third cluster, has been taken in by the first cluster.

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

```
> km.out = kmeans(x, 4, nstart=20)
> table(km.out$cluster, c(rep(1,20), rep(2,20), rep(3,20)))

    1  2  3
1  10  0  0
2   0 20  0
3  10  0  0
4   0  0 20
>
```

One of the original clusters has been split into two with the observations going into the first and third clusters.

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.out = kmeans(pca.out$x[,1:2], 3, nstart=20)
> table(km.out$cluster, c(rep(1,20), rep(2,20), rep(3,20)))

    1  2  3
1   0  0 20
2  20  0  0
3   0 20  0
>
```

These clusters compare very well to the true class labels again.

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.out = kmeans(scale(x), 3, nstart=20)
> table(km.out$cluster, c(rep(1,20), rep(2,20), rep(3,20)))

    1  2  3
1   8  4 12
2  12  0  7
3   0 16  1
>
```

These results are worse than the results from (b). By scaling the observations, the distance between them got worse.