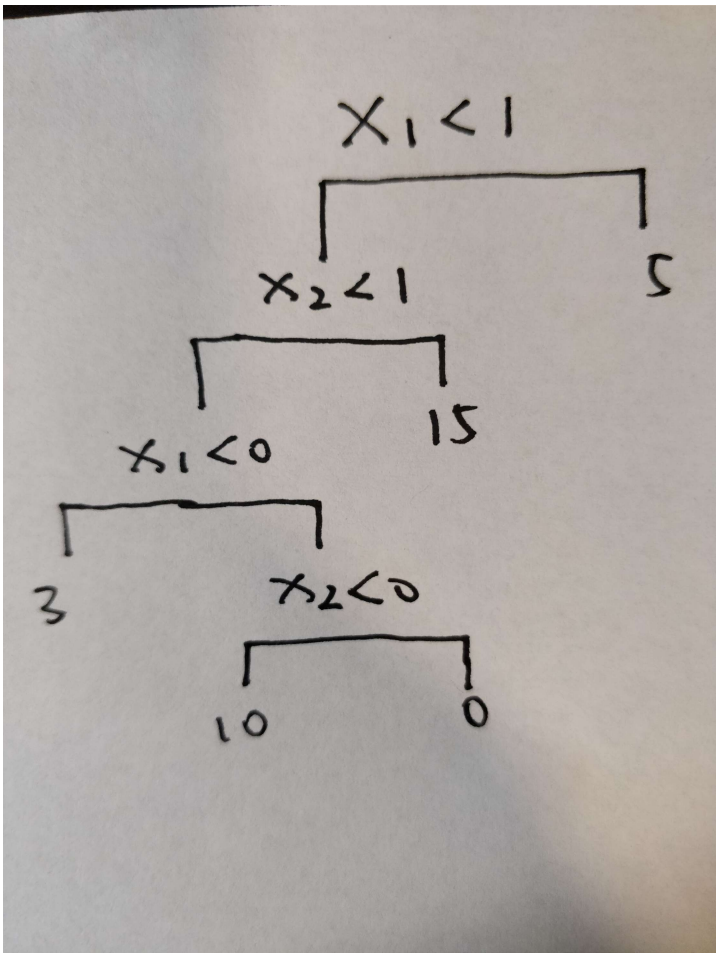
STATS 216: HW4

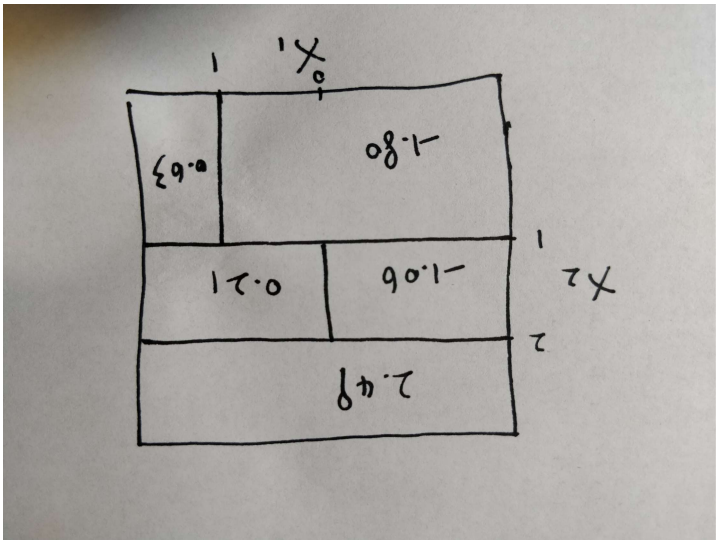
Xiangpeng Li

1.

a.



b.



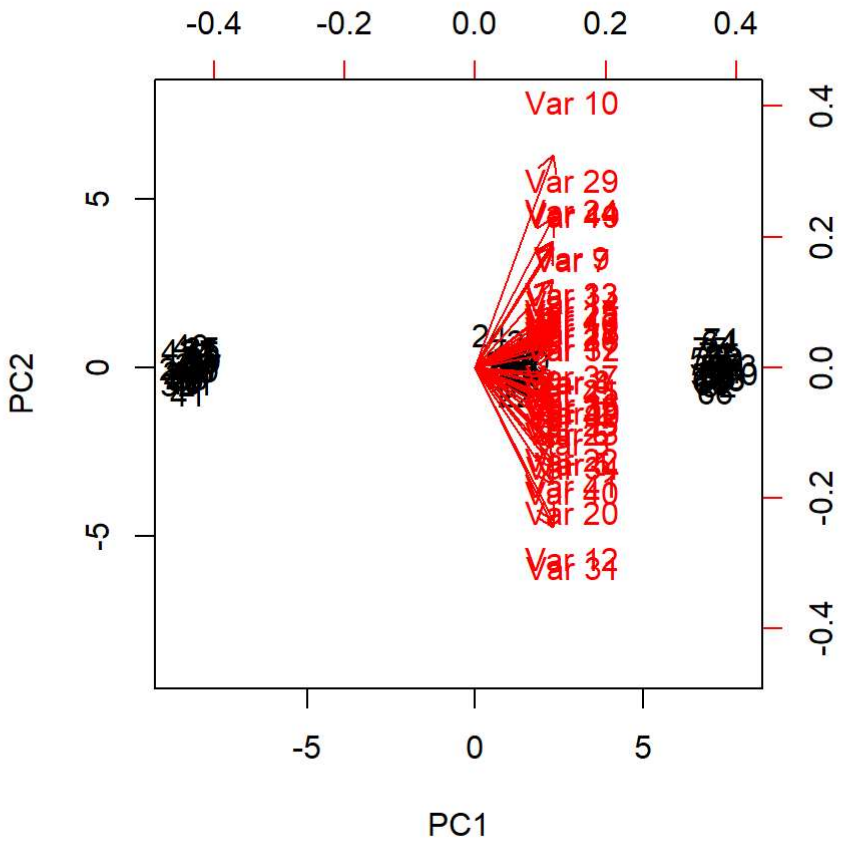
2.

a.

```
set.seed(1)
x = matrix(rnorm(25 * 3 * 45, mean = 0), ncol = 45)

# shift mean
x[1:25, 1:45] = x[1:25, 1:45] + 3
x[26:50, 1:45] = x[26:50, 1:45] - 3
x[51:75, 1:45] = x[51:75, 1:45] + 7
```

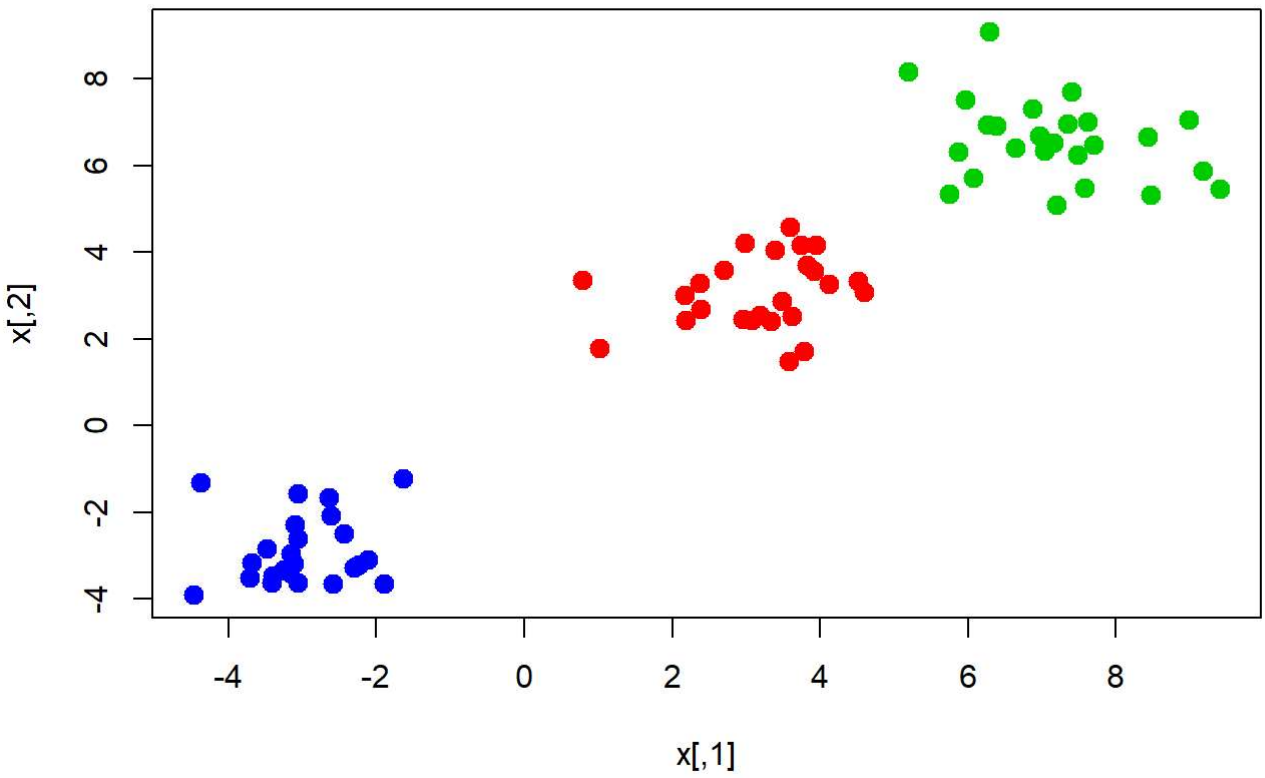
b.

```
# perform PCA
pca.fit = prcomp(x, scale = TRUE)
biplot(pca.fit, scale = 0)
```



c.

```
# perform K-means
kmeans3 = kmeans(x, 3, nstart = 20)

plot(x, col = (kmeans3$cluster + 1), pch = 20, cex = 2)
```



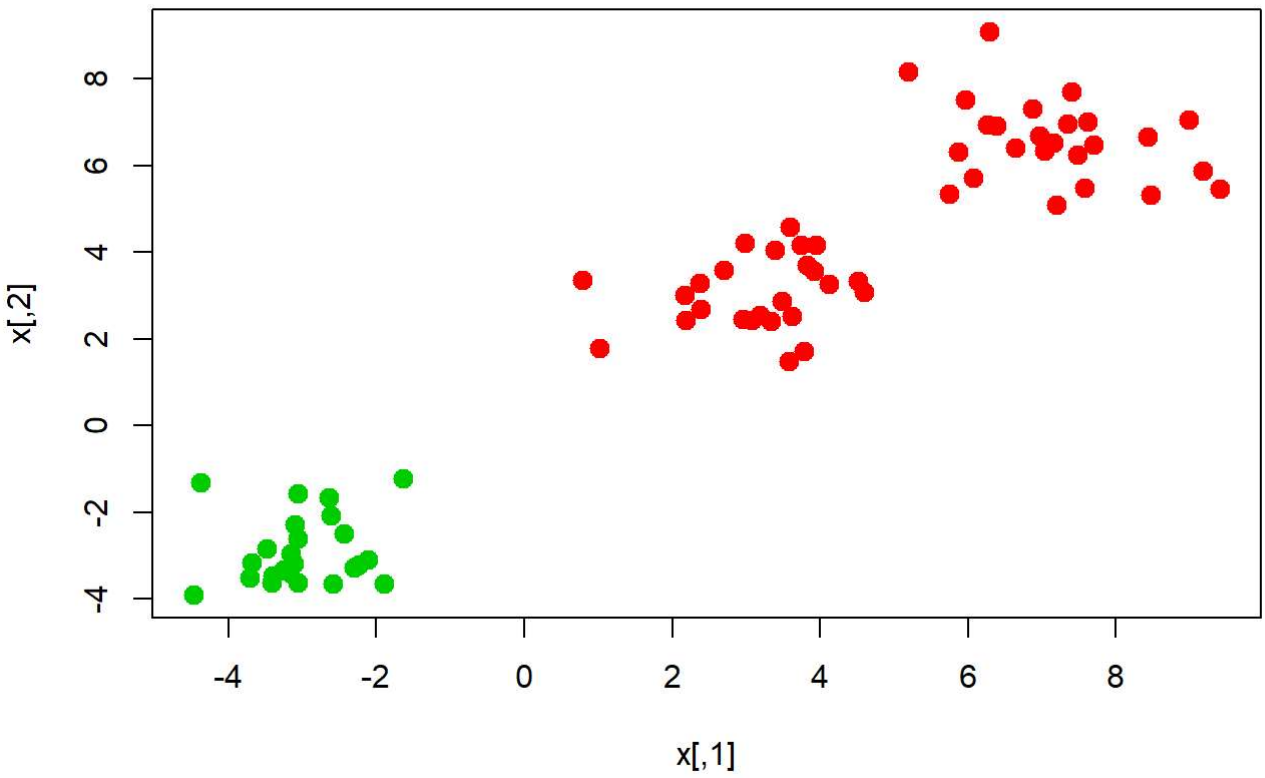

```
table(kmeans3$cluster, c(rep(2, 25), rep(1, 25), rep(3, 25)))
```

```
##
##      1  2  3
##   1  0 25  0
##   2  0  0 25
##   3 25  0  0
```

Since K-means will arbitrarily number the cluster, the label may not be correct, but we can clearly see the predictors are correctly clustered,

d.

```
kmeans2 = kmeans(x, 2, nstart = 20)

plot(x, col = (kmeans2$cluster + 1), pch = 20, cex = 2)
```



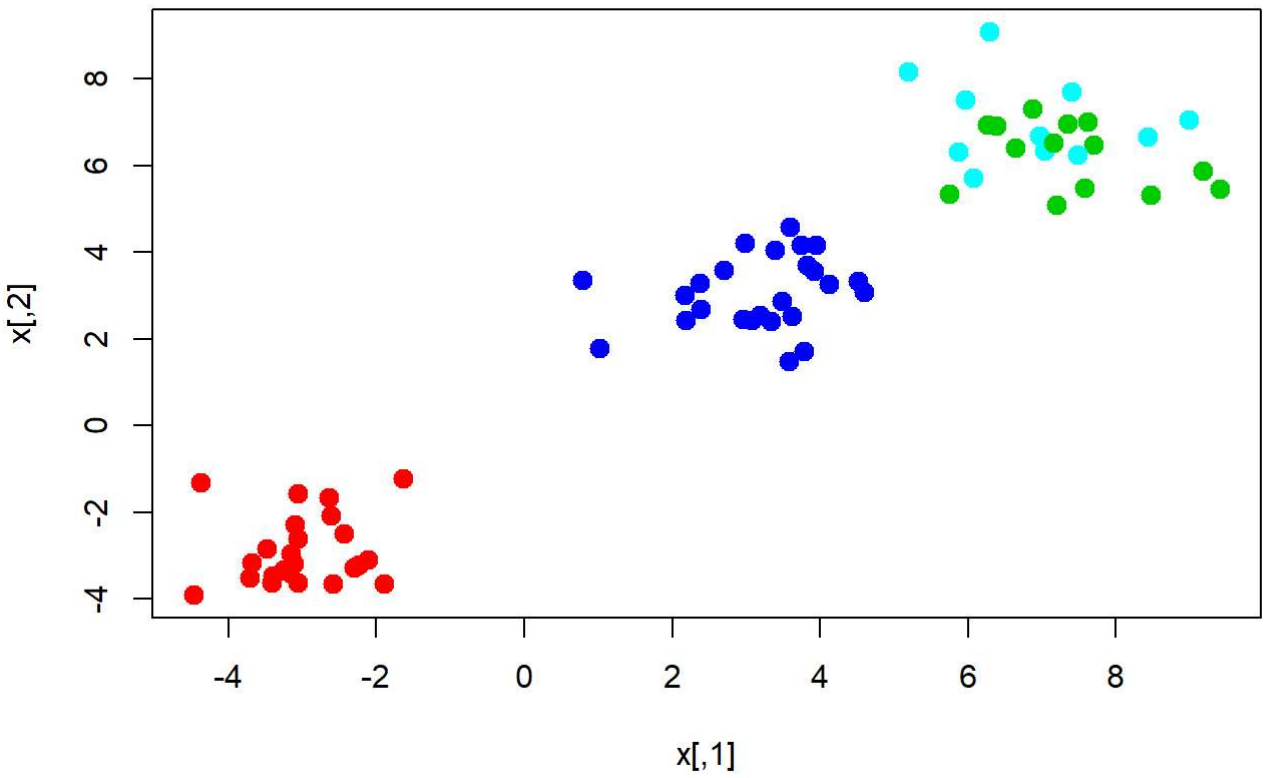

```
table(kmeans2$cluster, c(rep(2, 25), rep(1, 25), rep(3, 25)))
```

```
##
##      1  2  3
##   1  0 25 25
##   2 25  0  0
```

As we can see in the plot and table, all the observations in one group are fully labeled as another group.

e.

```
kmeans4 = kmeans(x, 4, nstart = 20)

plot(x, col = (kmeans4$cluster + 1), pch = 20, cex = 2)
```



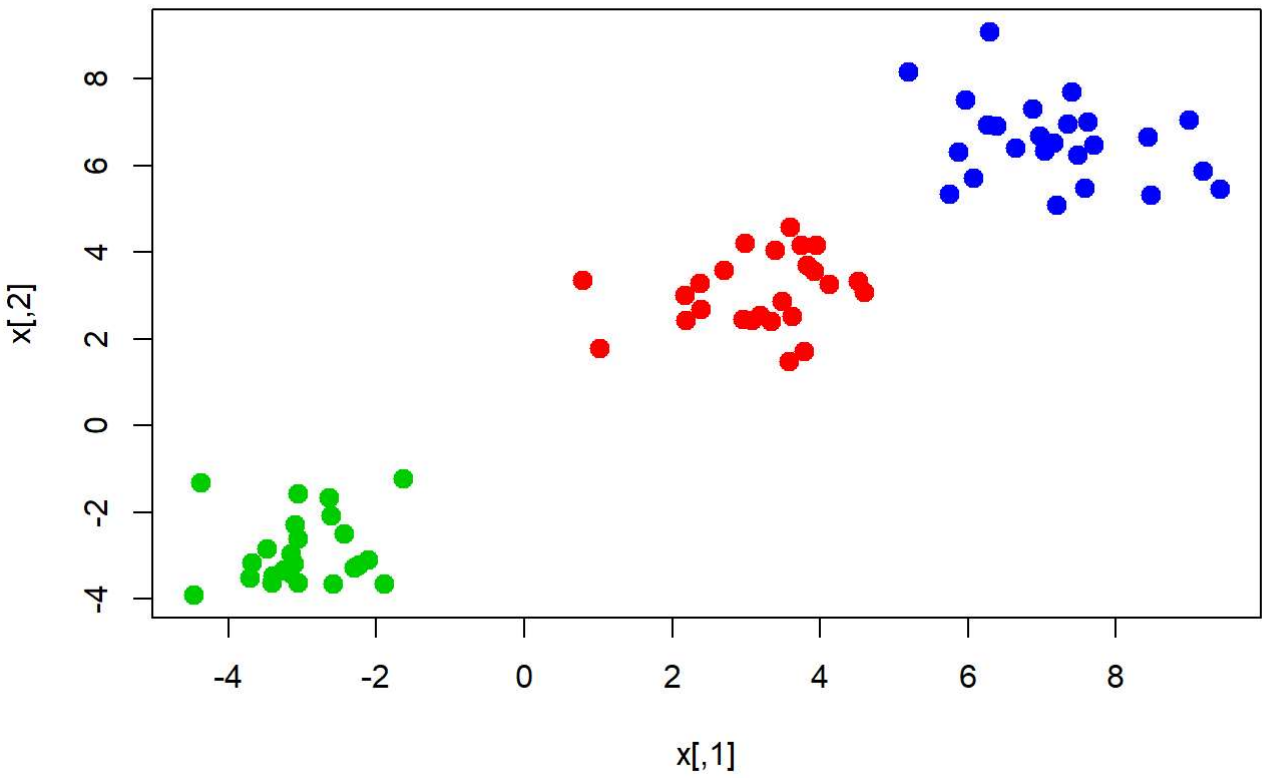

```
table(kmeans4$cluster, c(rep(2, 25), rep(1, 25), rep(3, 25)))
```

```
##
##      1  2  3
##   1 25  0  0
##   2  0  0 14
##   3  0 25  0
##   4  0  0 11
```

In this fit, observations from top right are divided into two clusters.

f.

```
kmeans.pca.fit = kmeans(pca.fit$x[1:75, 1:2], 3, nstart = 20)

plot(x, col = (kmeans.pca.fit$cluster + 1), pch = 20, cex = 2)
```



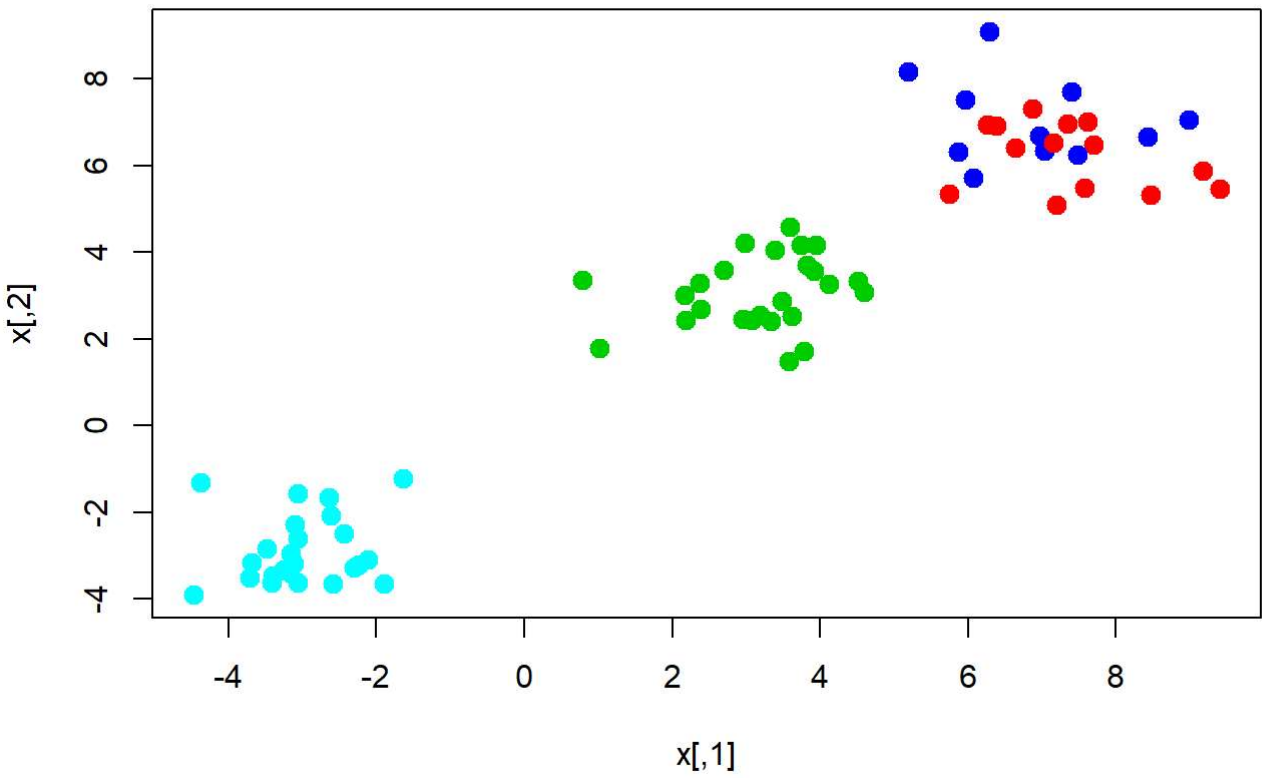

```
table(kmeans.pca.fit$cluster, c(rep(2, 25), rep(1, 25), rep(3, 25)))
```

```
##
##      1  2  3
##   1  0 25  0
##   2 25  0  0
##   3  0  0 25
```

All the observations are correctly labeled. It means that first and second components explain the majority of the observations.

g.

```
kmeansScale = kmeans(scale(x), 4, nstart = 20)

plot(x, col = (kmeansScale$cluster + 1), pch = 20, cex = 2)
```




```
table(kmeansScale$cluster, c(rep(2, 25), rep(1, 25), rep(3, 25)))
```

```
##
##      1  2  3
##   1  0  0 14
##   2  0 25  0
##   3  0  0 11
##   4 25  0  0
```

Some observations are not labeled correctly. Because each pairwise distance will be affected by the scale function. If the measure units are already the same, we should not scale it again.

3.

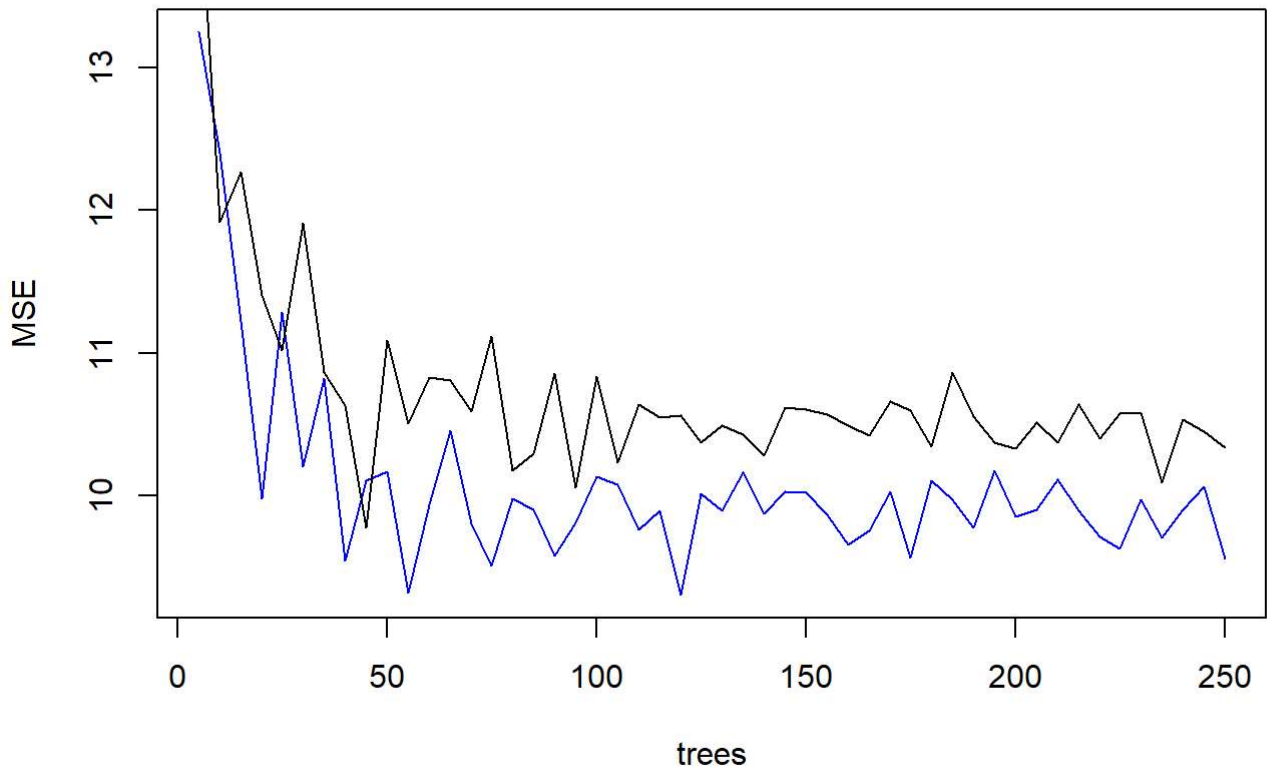
a.

```
# setup data
setwd("D:/One Drive/OneDrive/Document/Study/Stanford/Introduction to Statistical Learning/homewo
rk/hw4")
load("body.RData")
set.seed(1)
traingIndex = sample(seq_len(nrow(X)), size = 307)
bag.mse = rep(0, 50)
rf.mse = rep(0, 50)
index = c(1: 50)
training = X[traingIndex, ]
trainRes = Y[traingIndex, ]
test = X[-traingIndex, ]
testRes = Y[-traingIndex, ]


# fit the model
library(randomForest)
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
for (i in index) {
  bag.fit = randomForest(trainRes$Weight ~ ., data = training, ntree = i * 5, mtry = ncol(traini
ng), importance = TRUE)
  rf.fit = randomForest(trainRes$Weight ~ ., data = training, ntree = i * 5, mtry = ncol(trainin
g) / 3, importance = TRUE)
  bag.pred = predict(bag.fit, newdata = test)
  rf.pred = predict(rf.fit, newdata = test)
  bag.mse[i] = mean((bag.pred - testRes$Weight)^2)
  rf.mse[i] = mean((rf.pred - testRes$Weight)^2)
}
plot(index * 5, rf.mse, xlab = "trees", ylab = "MSE", col = "blue", type = "l")
lines(index * 5, bag.mse)
```



b.

```
bag.fit = randomForest(trainRes$Weight ~ ., data = training, mtry = ncol(training), importance =
 TRUE)
rf.fit = randomForest(trainRes$Weight ~ ., data = training, mtry = ncol(training) / 3, importanc
e = TRUE)
importance(bag.fit)
```

```
##                       %IncMSE IncNodePurity
## Wrist.Diam           6.213179      114.1910
## Wrist.Girth          9.723407      379.4467
## Forearm.Girth       12.681253     2516.7483
## Elbow.Diam          11.525439      254.7523
## Bicep.Girth          8.369623     2642.5175
## Shoulder.Girth      18.262922    12025.7172
## Biacromial.Diam     11.526198      200.3680
## Chest.Depth         10.039241      222.6029
## Chest.Diam           9.353270      272.9933
## Chest.Girth         16.180858     7039.8064
## Navel.Girth         10.955272      435.7336
## Waist.Girth         26.585136    19675.2072
## Pelvic.Breadth      11.617856      194.3229
## Bitrochanteric.Diam  7.456843      145.4929
## Hip.Girth           28.822096     3708.7290
## Thigh.Girth         19.839332      590.1507
## Knee.Diam           14.643604      573.4334
## Knee.Girth          24.639385     1806.0981
## Calf.Girth          12.704152      433.7513
## Ankle.Diam          11.393109      304.3159
## Ankle.Girth         11.363095      368.3032
```

```
importance(rf.fit)
```

```
##                       %IncMSE IncNodePurity
## Wrist.Diam           7.201574      208.1603
## Wrist.Girth          7.818100     1110.1029
## Forearm.Girth       12.456342     4601.8990
## Elbow.Diam           7.166250      715.3144
## Bicep.Girth         11.155830     5283.4467
## Shoulder.Girth      15.591795     9125.2871
## Biacromial.Diam      8.289636      202.4015
## Chest.Depth          8.285154      867.3976
## Chest.Diam           8.263369     1161.6197
## Chest.Girth         16.310453     8456.0094
## Navel.Girth         12.133828      803.9854
## Waist.Girth         20.073316    12823.6658
## Pelvic.Breadth      10.513057      279.3287
## Bitrochanteric.Diam  9.606671      333.0635
## Hip.Girth           24.636410     2982.1815
## Thigh.Girth         17.786278      867.2484
## Knee.Diam           13.262410      636.4748
## Knee.Girth          20.754587     1833.2621
## Calf.Girth          13.824790      693.6542
## Ankle.Diam          10.897502      378.8536
## Ankle.Girth          9.305935      597.5383
```

For Bagging, Waist.Girth has the largest %IncMSE and IncNodePurity, so it's the most important variable. Fot Random forest, Hip.Girth and Waist.Girth are the most importance variables, they have the largest %IncMSE and IncNodePurity.

c.

```
rf.fit = randomForest(trainRes$Weight ~ ., data = training, mtry = ncol(training) / 3, importanc
e = TRUE, ntree = 500)
rf.pred = predict(rf.fit, newdata = test)
rf.mse = mean((rf.pred - testRes$Weight)^2)
rf.mse1 = mean((rf.pred - testRes)^2)
rf.mse
```

```
## [1] 9.65916
```

```
rf.mse1
```

```
## [1] 4269.325
```

In homework 3(f) I'm using the whole test variables instead of Weight only to calculate MSE, so the numbers are off. But still we can use the same apporoach to calculate the MSE so we can have a basic idea of their performance.

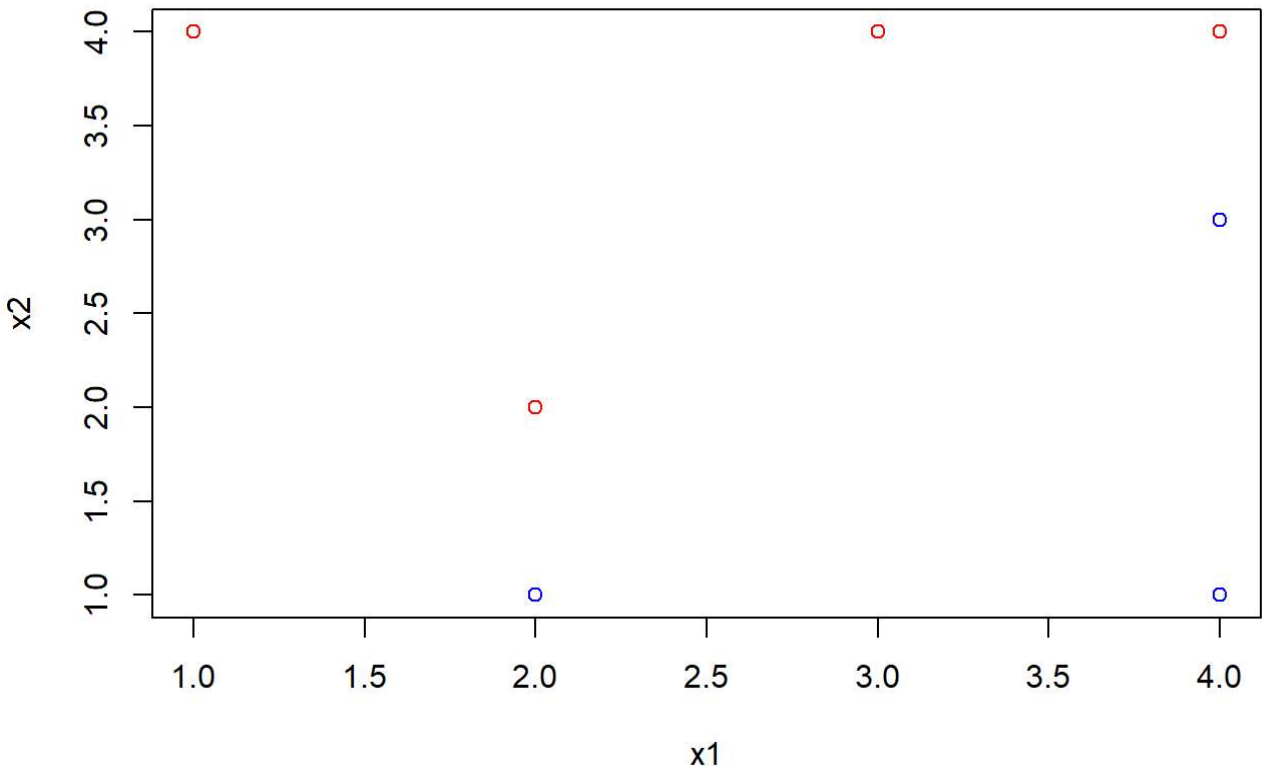
PCR: 4281.326, PLS: 4279.822, LASSO: 4280.238, RF: 4268.863. Random forest have the lowest MSE so it has the best performance.

d. As we can see from the plot in (a), both Bagging and random forest don't have much difference when the numbers of trees are above 100, so the default setting is enough for us in this scenario.

4

a.

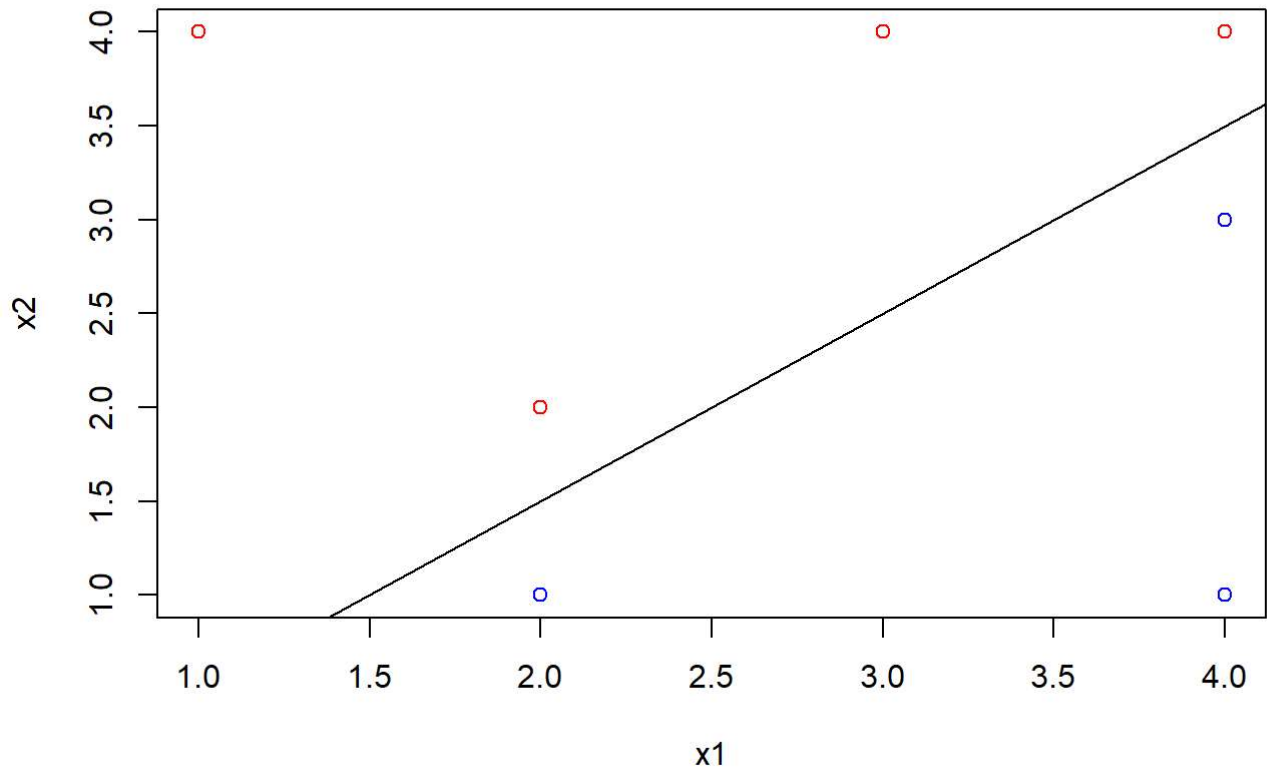
```
x1 = c(3, 2, 4, 1, 2, 4, 4)
x2 = c(4, 2, 4, 4, 1, 3, 1)
y = c("red", "red", "red", "red", "blue", "blue", "blue")

plot(x1, x2, col = y, type = "p")
```



b. As we can see from above figure, points $(2, 1), (2, 2), (4, 4), (4, 3)$ are the supporting vector, the boundary has to be within that region. In order to get the maximum margin, we can conclude the line has to pass through $(2, 1.5), (4, 3.5)$. So the equation will be

$$-0.5 + x1 - x2 = 0$$

```
plot(x1, x2, col = y, type = "p")
abline(a = -0.5, b = 1)
```

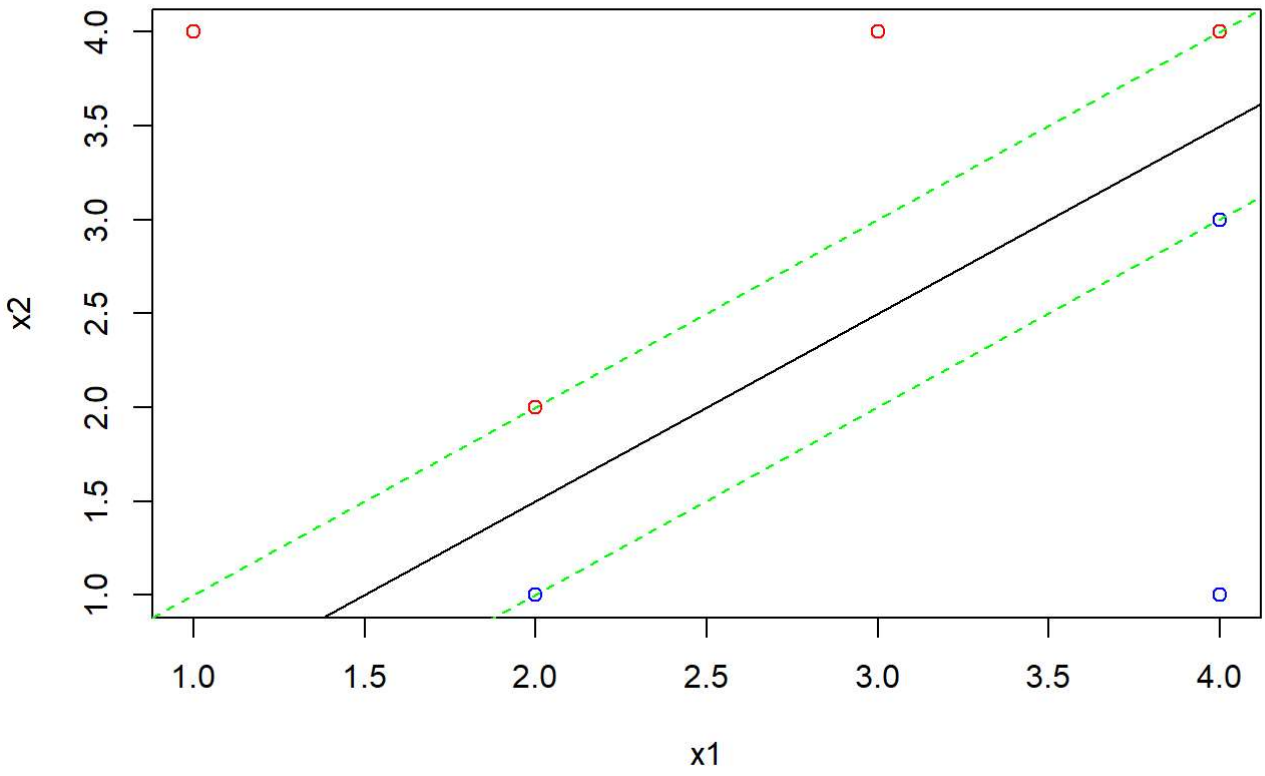


c.

$\beta_0 = -0.5, \beta_1 = 1, \beta_2 = -1$ Classify to Red if $-0.5 + x1 - x2 = 0 > 0$ and classify to blue otherwise

d.

```
plot(x1, x2, col = y, type = "p")
abline(a = -1, b = 1, col = "green", lty = 2)
abline(a = 0, b = 1, col = "green", lty = 2)
abline(a= -0.5, b = 1)
```



Then

we calculate the distance between the solid black line and the dotted green line $0.5 - x1 - x2 = 0$ and $-1 + x1 - x2 = 0$. Through the distance formala we get the distance is $\sqrt{2}/4$

e.

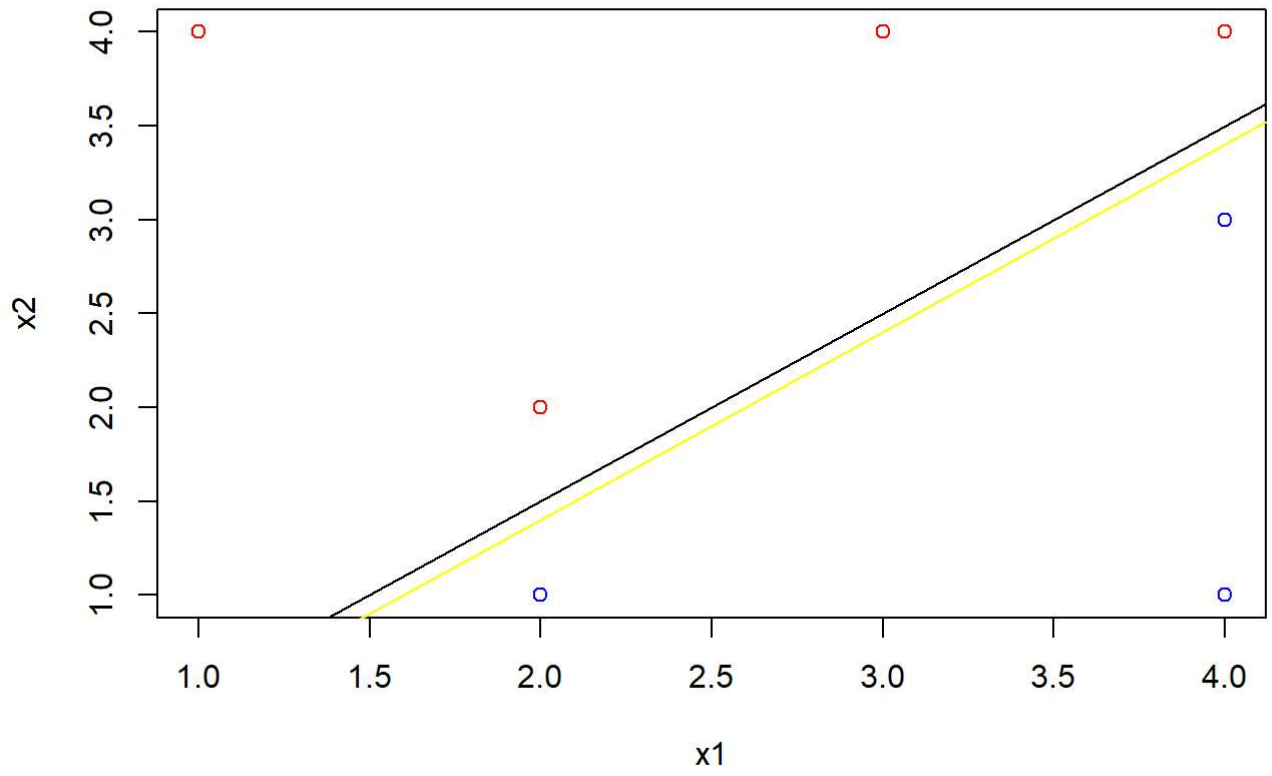
points $(2, 1), (2, 2), (4, 4), (4, 3)$ are the supporting vectors

f.

It will not affect the maximal margin hyperplane. As we can see from the above figure, point $(4, 1)$ is far from the maximal margin classifier line. As long as it's not moving into the margin, it has no affect on the hyperplane.

g.

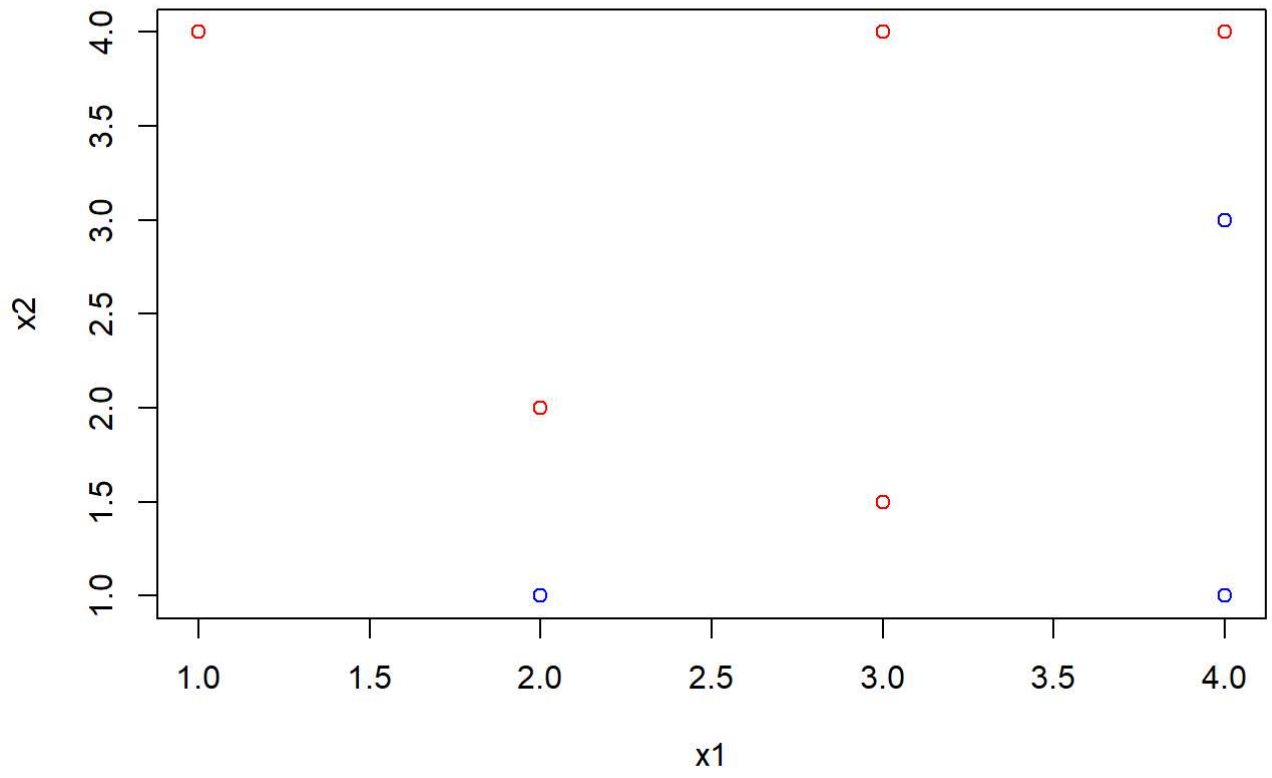
```
plot(x1, x2, col = y, type = "p")
abline(a= -0.5, b = 1)
abline(a= -0.6, b = 1, col = "yellow")
```



I drawed a yellow line with different intercept(0.6), the equation is: $-0.6 + x1 - x2 = 0 > 0$

h.

```
plot(x1, x2, col = y, type = "p")
points(3, 1.5, col = "red")
```



I drawed a point at (3, 1.5) in red. In this case they cannot be separated by a hyperplane.