BIOSTAT 285 Spring 2020 Homework 3

Due 11 PM 5/29/2020

Nan Liu

Remark. For **Computational Part**, please complete your answer in the **RMarkdown** file and summit it with the generated PDF file to CCLE.

Computational Part

1. (SVM, *20 pt*) 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.

```
library(ISLR)
data(Auto)
#mpgadd <- Auto$mpg
```

(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(tidyverse)
```

```
## -- Attaching packages --------------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.0     v purrr   0.3.4
## v tibble  3.0.1     v dplyr   0.8.5
## v tidyr   1.1.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0

## Warning: package 'tibble' was built under R version 3.6.2

## Warning: package 'tidyr' was built under R version 3.6.2

## Warning: package 'purrr' was built under R version 3.6.2

## -- Conflicts ------------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
Auto <- Auto %>%
  mutate (mpg01 = as.factor(ifelse(mpg > median(mpg), 1, 0)))
```

(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.

From the homework assignment 2 we know 'cylinders', 'displacement', 'horsepower' and 'weight' seem mostly likely to be useful in predicting `mpg01`. So we select these four variables in our model.

```
#install.packages("e1071")
set.seed(1)
library(e1071)
library(caret)
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```
tunelinear <- tune(svm, mpg01 ~ cylinders + displacement +
                     horsepower + weight, data = Auto,
                     kernel = "linear",
                     ranges = list(cost = c(0.01, 0.1, 1, 5, 10, 100)))
summary(tunelinear)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##  0.01
##
## - best performance: 0.09955128
##
## - Detailed performance results:
##    cost      error dispersion
## 1 1e-02 0.09955128 0.04275859
## 2 1e-01 0.09955128 0.04275859
## 3 1e+00 0.09955128 0.04275859
## 4 5e+00 0.09955128 0.04275859
## 5 1e+01 0.09955128 0.04275859
## 6 1e+02 0.09955128 0.04275859
```

The optimal value of cost is 0.01 and the cross-validation error is 0.09955128 at this time. The cross-validation error is relatively small, so we conclude the model can predict whether a car gets high or low gas mileage pretty well.

(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.

```r
#radial kernal
set.seed(1)
tuneradial <- tune(svm, mpg01 ~ cylinders + displacement +
                     horsepower + weight,
                   data = Auto, kernel = "radial",
                 ranges = list(cost = c(0.1, 1, 10, 100),
                               gamma = c(0.01, 0.5, 1, 2, 3, 4)))
summary(tuneradial)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##    10     1
##
## - best performance: 0.07403846
##
## - Detailed performance results:
##     cost gamma      error dispersion
## 1    0.1  0.01 0.13269231 0.05514394
## 2    1.0  0.01 0.09955128 0.04275859
## 3   10.0  0.01 0.09955128 0.04275859
## 4  100.0  0.01 0.09948718 0.03713378
## 5    0.1  0.50 0.09955128 0.04275859
## 6    1.0  0.50 0.10205128 0.04828663
## 7   10.0  0.50 0.08929487 0.04561858
## 8  100.0  0.50 0.08166667 0.03969554
## 9    0.1  1.00 0.09955128 0.04275859
## 10   1.0  1.00 0.10205128 0.05122308
## 11  10.0  1.00 0.07403846 0.04097358
## 12 100.0  1.00 0.09192308 0.04229712
## 13   0.1  2.00 0.09435897 0.04514346
## 14   1.0  2.00 0.08679487 0.05015153
## 15  10.0  2.00 0.08173077 0.03799005
## 16 100.0  2.00 0.08423077 0.03636273
## 17   0.1  3.00 0.09192308 0.04569973
## 18   1.0  3.00 0.07910256 0.04084741
## 19  10.0  3.00 0.08166667 0.03148532
## 20 100.0  3.00 0.08166667 0.02907271
## 21   0.1  4.00 0.09705128 0.04345463
## 22   1.0  4.00 0.08160256 0.03961477
## 23  10.0  4.00 0.08166667 0.03148532
## 24 100.0  4.00 0.10217949 0.05143158
```

```r
#polynomial
set.seed(1)
tunepoly <- tune(svm, mpg01 ~ cylinders + displacement +
                     horsepower + weight,
                 data = Auto, kernel = "polynomial",
                 ranges = list(cost = c(0.1, 1, 10, 100, 1000),
                               #gamma = c(0.5, 1, 2, 3, 4),
                               degree = c(2, 3, 4)))
summary(tunepoly)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost degree
##   100      3
##
## - best performance: 0.09455128
##
## - Detailed performance results:
##     cost degree      error dispersion
## 1  1e-01      2 0.25551282 0.09508100
## 2  1e+00      2 0.24782051 0.09348880
## 3  1e+01      2 0.25794872 0.09527097
## 4  1e+02      2 0.24275641 0.08735844
## 5  1e+03      2 0.23512821 0.08609059
## 6  1e-01      3 0.23243590 0.11154427
## 7  1e+00      3 0.12512821 0.03511666
## 8  1e+01      3 0.10217949 0.04001059
## 9  1e+02      3 0.09455128 0.04553363
## 10 1e+03      3 0.10211538 0.04201319
## 11 1e-01      4 0.26064103 0.10166774
## 12 1e+00      4 0.24782051 0.09348880
## 13 1e+01      4 0.24782051 0.09348880
## 14 1e+02      4 0.25032051 0.08433285
## 15 1e+03      4 0.24756410 0.07683328
```

When using the SVM with radial basis kernel, the optiaml value of `gamma` is 1 and the optimal value of `cost` is 10. The cross-validation error is 0.07403846 at this time. When using the SVM with polynomial basis kernel, the optiaml value of `degree` is 3 and the optimal value of `cost` is 100.The cross-validation error is 0.09455128 at this time.

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

Hint: In the lab, we used the `plot()` function for `svm` objects only in cases with $p = 2$. When $p > 2$, you can use the `plot()` function to create plots displaying pairs of variables at a time. Essentially, instead of typing

```
> plot(svmfit, dat)
```

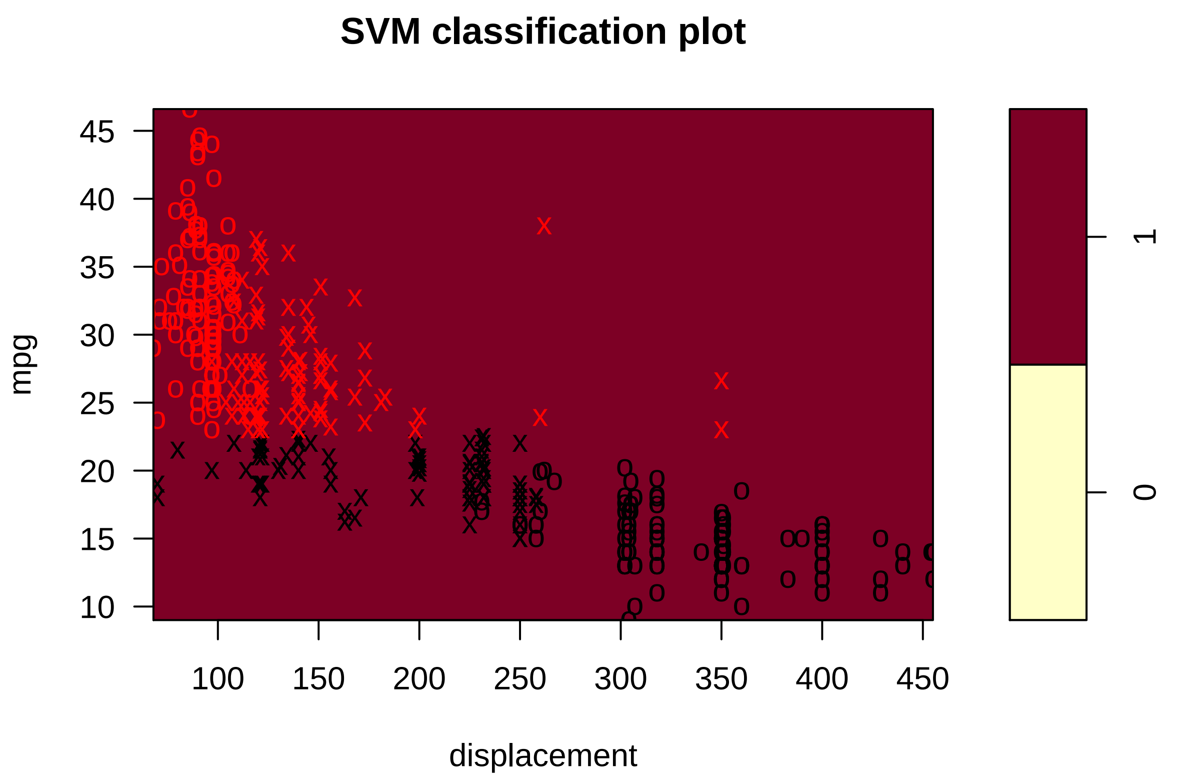
where svmfit contains your fitted model and dat is a data frame containing your data, you can type

```
> plot(svmfit, dat, x1~x4)
```

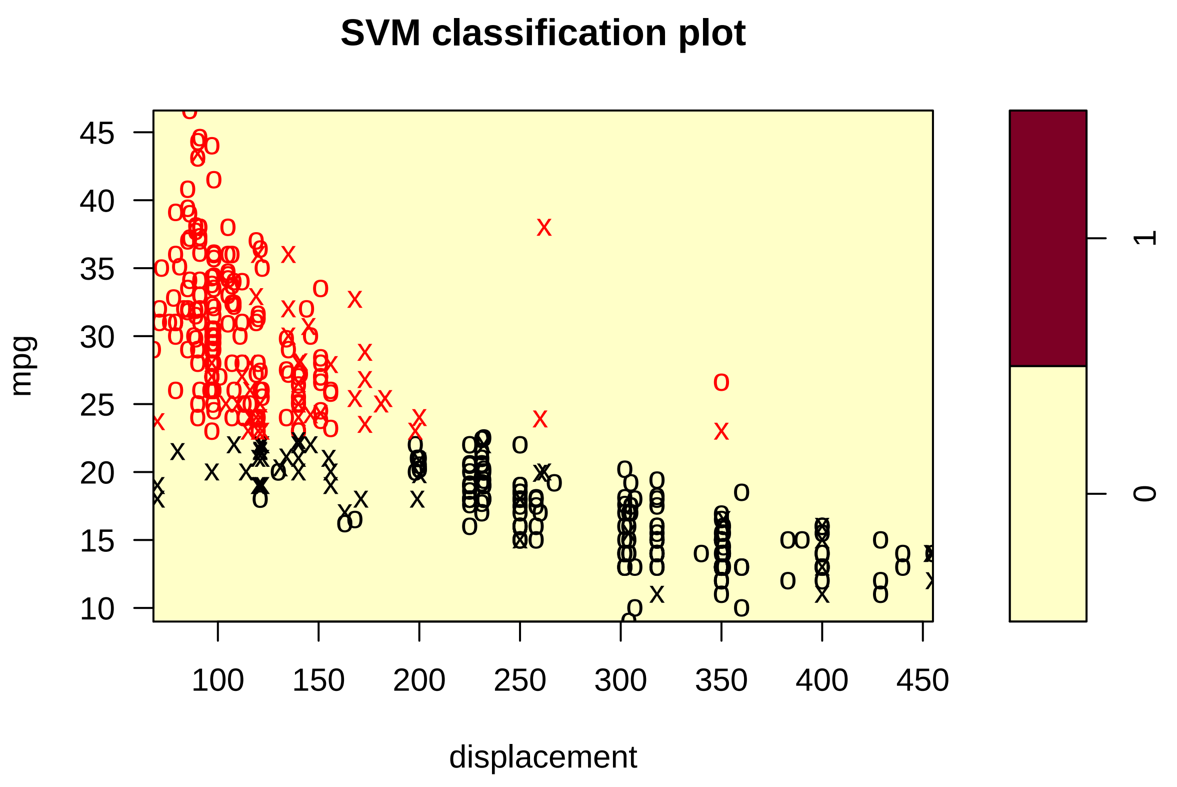
in order to plot just the first and fourth variables. However, you must replace `x1` and `x4` with the correct variable names. To find out more, type `?plot.svm`.

```
svm.linear <- svm(mpg01 ~ cylinders + displacement +
                      horsepower + weight,
                  data = Auto, kernel = "linear", cost = 0.01)
svm.radial <- svm(mpg01 ~ cylinders + displacement +
                      horsepower + weight,
                  data = Auto, kernel = "radial",
                  cost = 10, gamma = 1)
svm.poly <- svm(mpg01 ~ cylinders + displacement +
                      horsepower + weight,
                data = Auto, kernel = "polynomial",
                  cost = 100, degree = 3, gamma = 1)
```

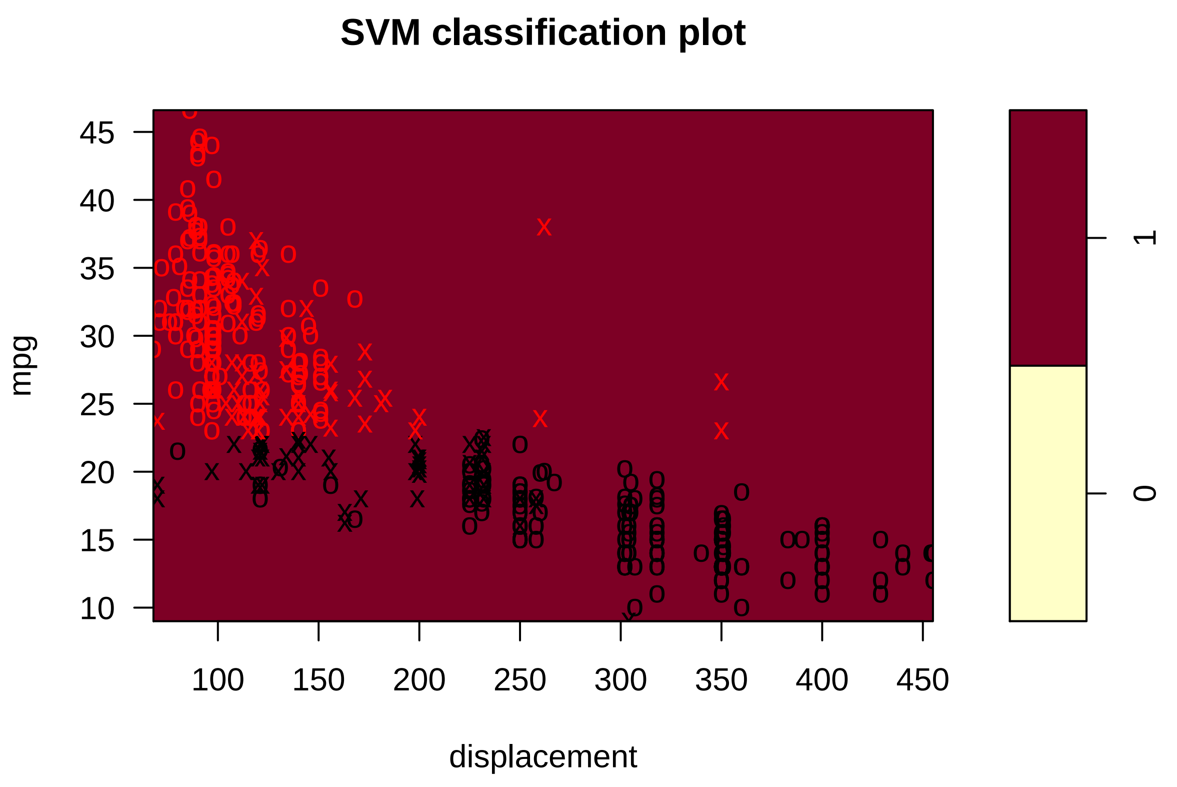
```
plot(svm.linear, Auto, as.formula(mpg ~ displacement))
```



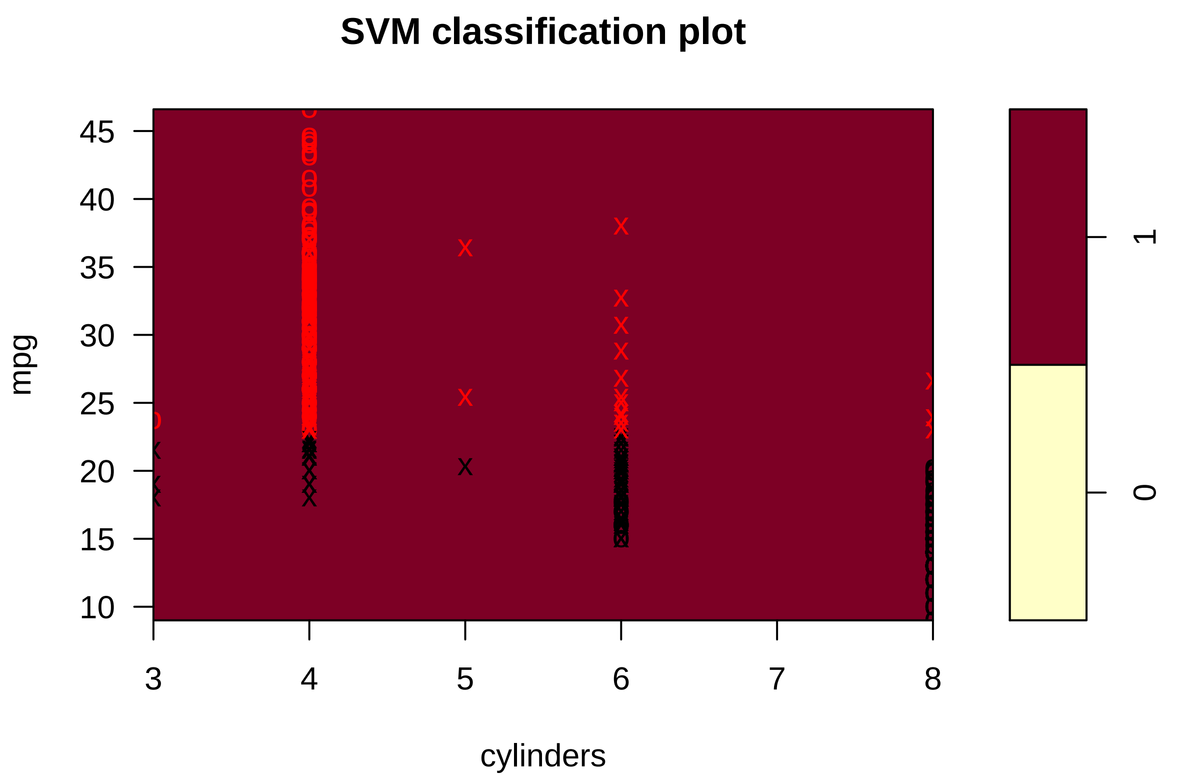
```
plot(svm.radial, Auto, as.formula(mpg ~ displacement))
```



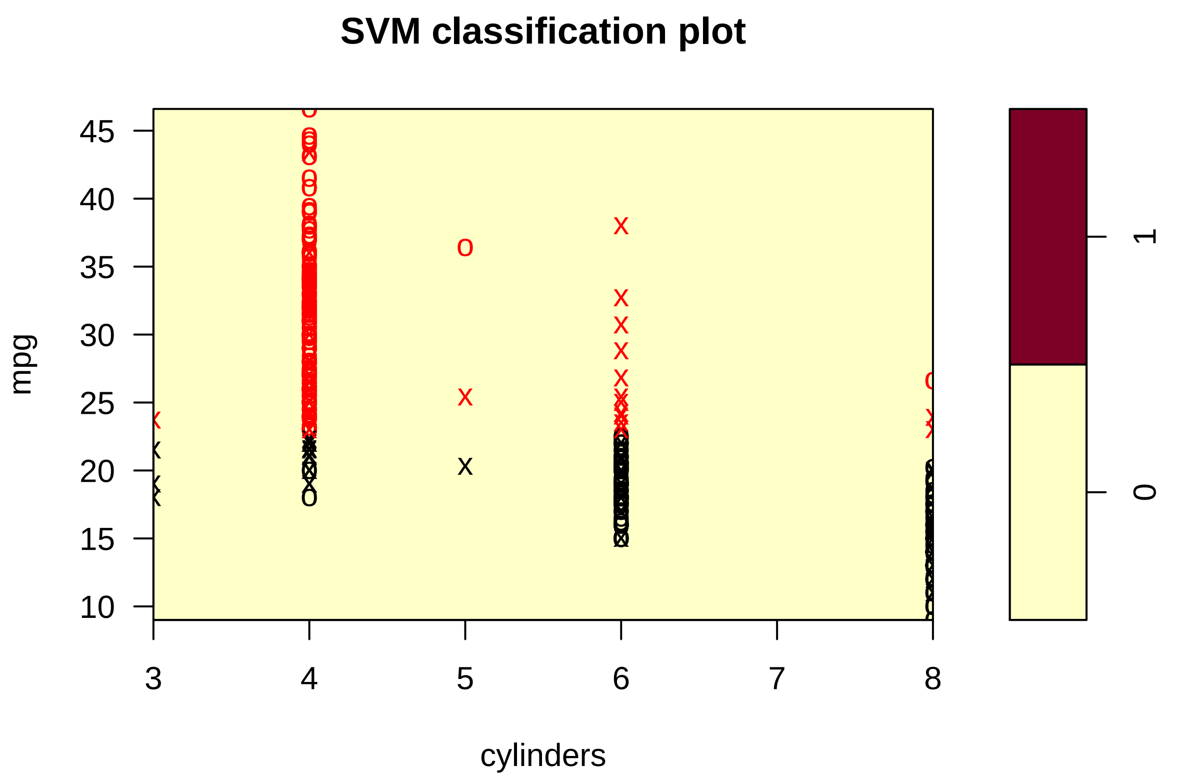
```r
plot(svm.poly, Auto, as.formula(mpg ~ displacement))
```



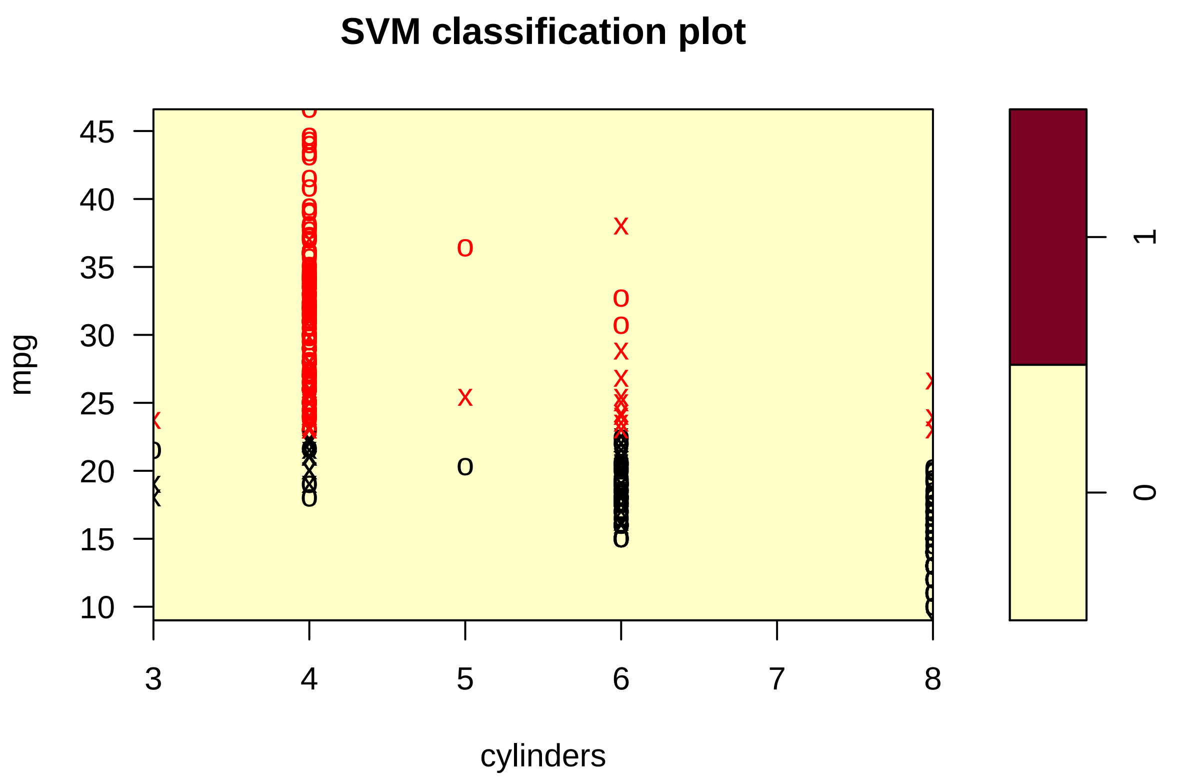
```r
plot(svm.linear, Auto, as.formula(mpg ~ cylinders))
```



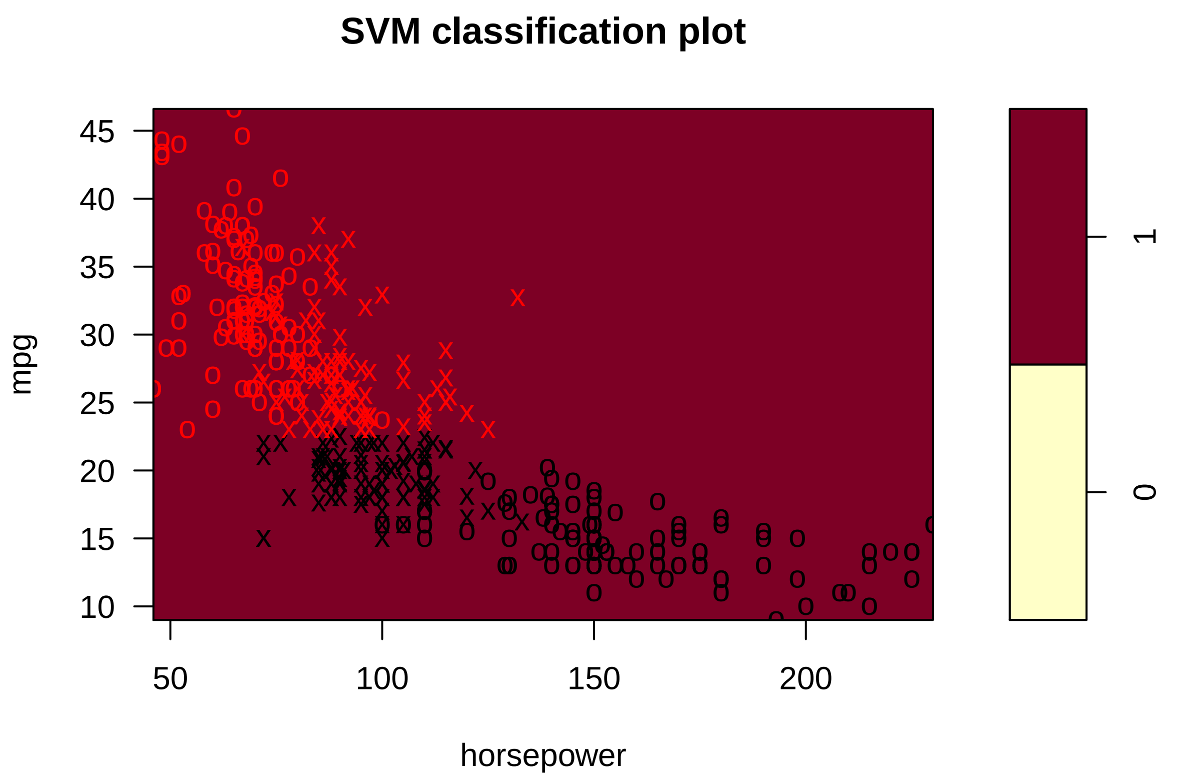
```r
plot(svm.radial, Auto, as.formula(mpg ~ cylinders))
```



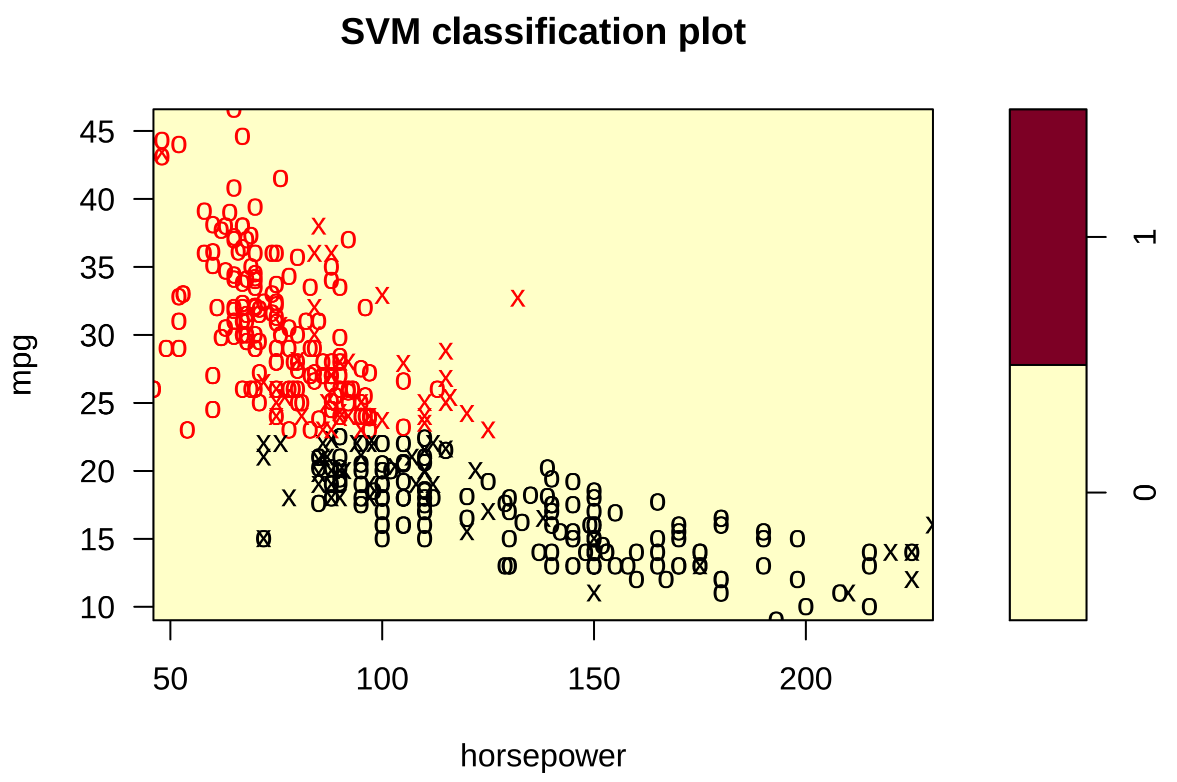
```r
plot(svm.poly, Auto, as.formula(mpg ~ cylinders))
```



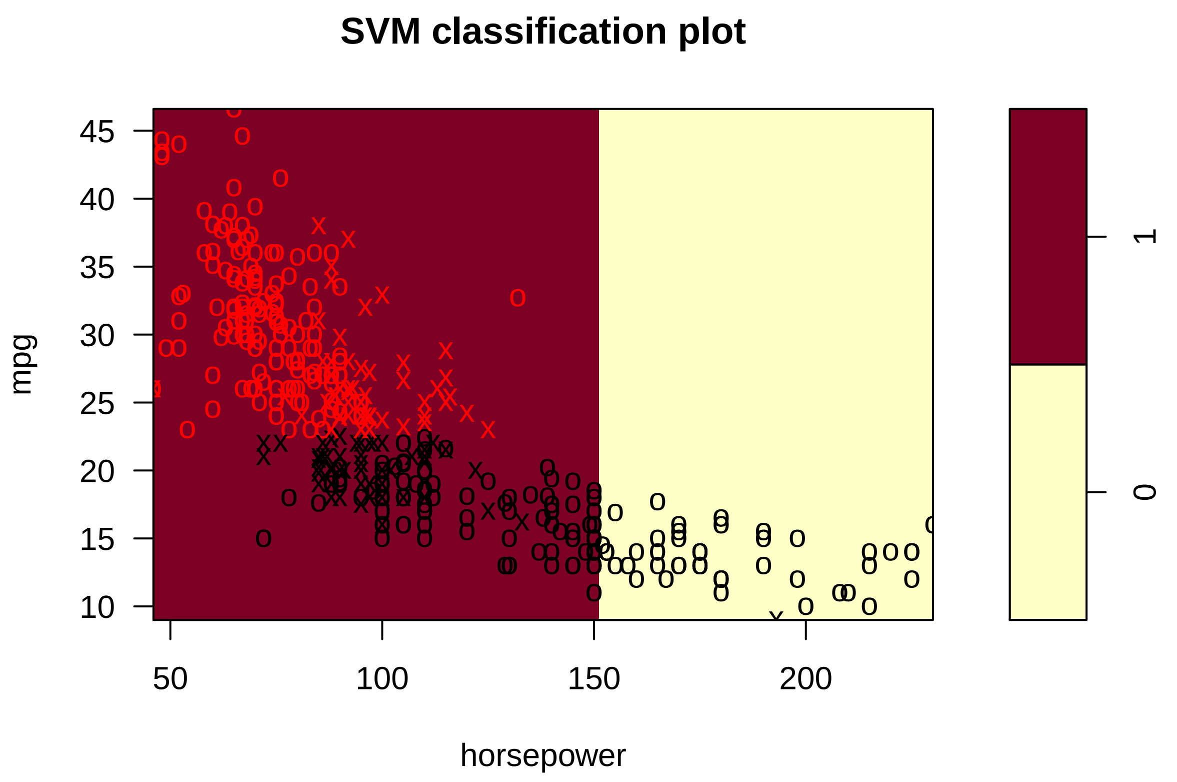
```r
plot(svm.linear, Auto, as.formula(mpg ~ horsepower))
```



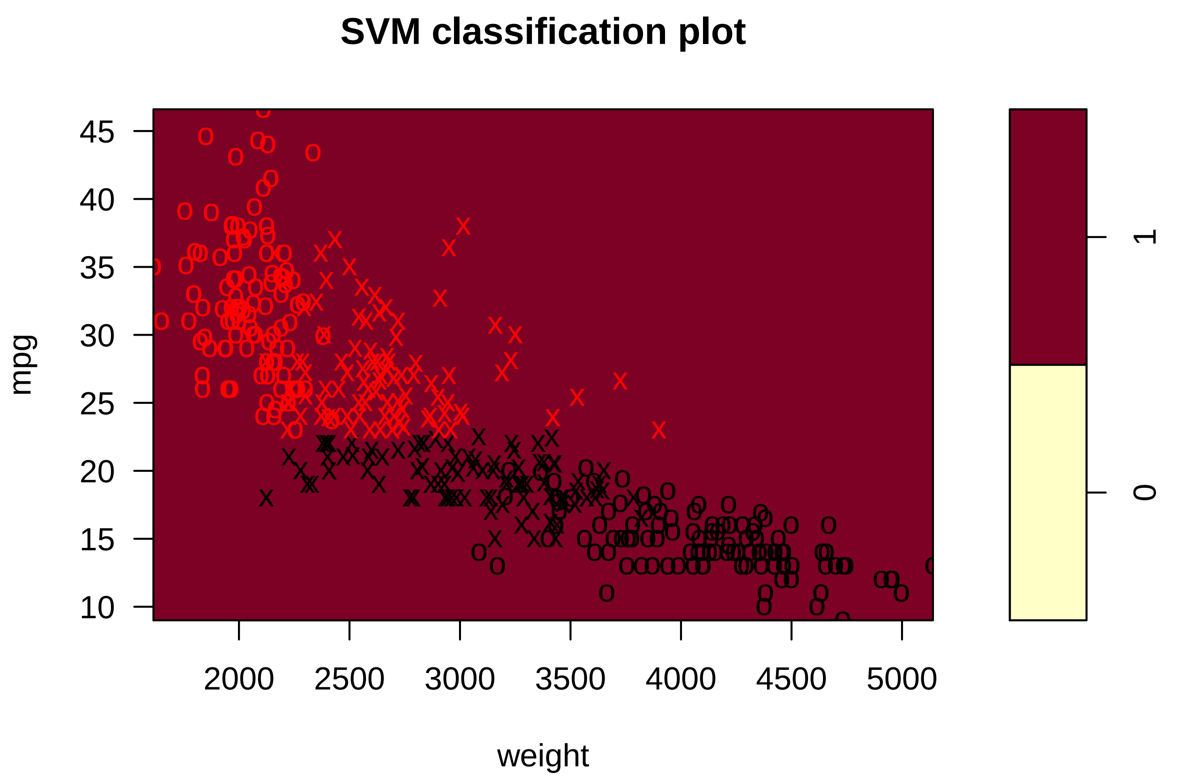
```r
plot(svm.radial, Auto, as.formula(mpg ~ horsepower))
```



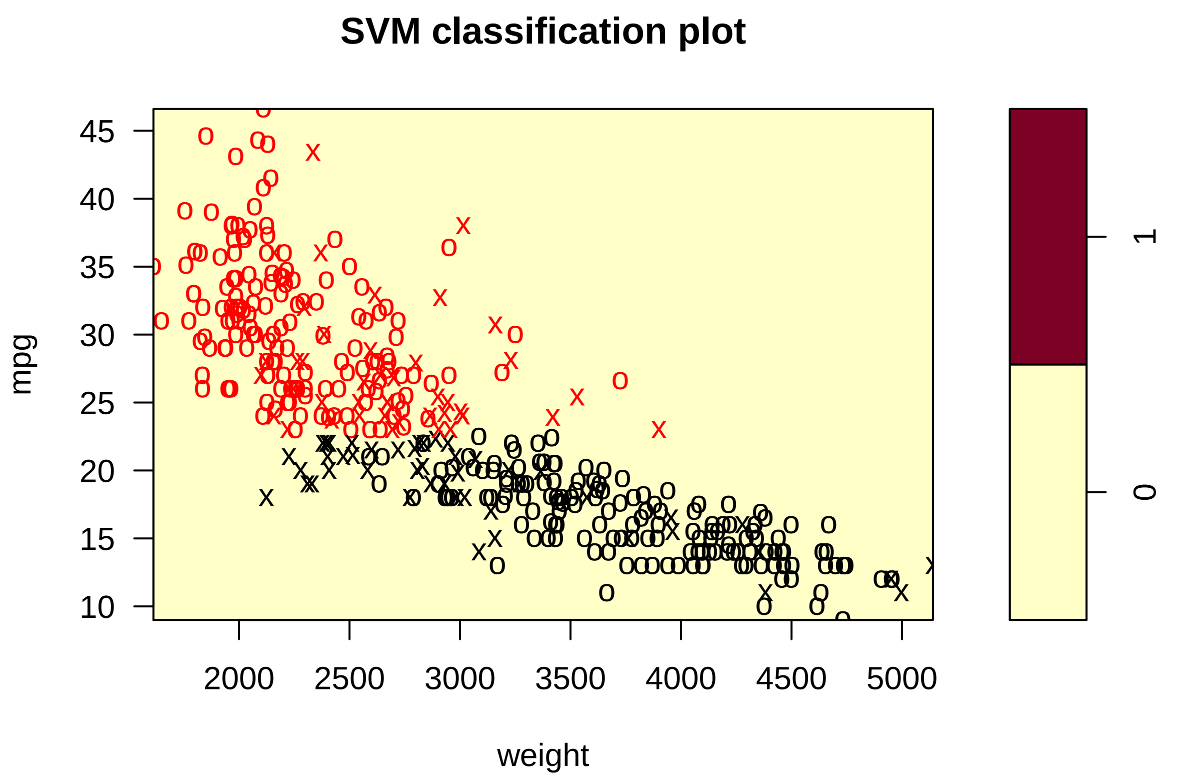
```r
plot(svm.poly, Auto, as.formula(mpg ~ horsepower))
```



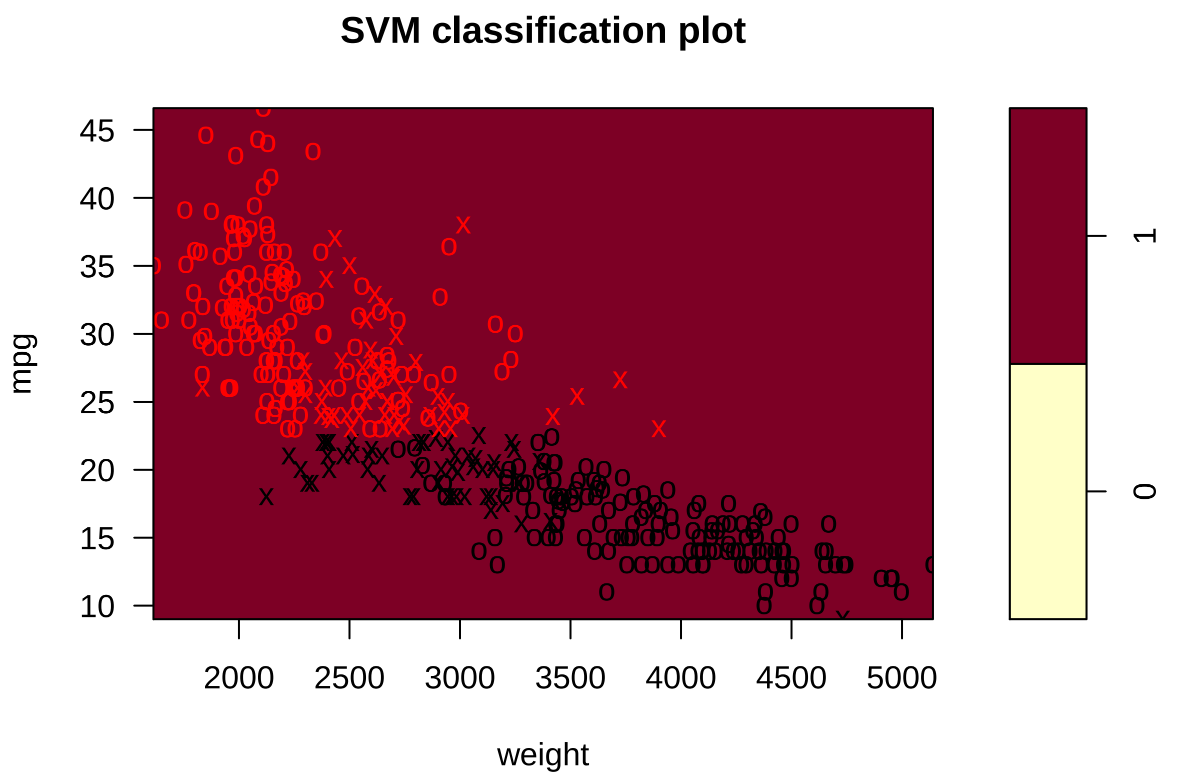
```r
plot(svm.linear, Auto, as.formula(mpg ~ weight))
```



```r
plot(svm.radial, Auto, as.formula(mpg ~ weight))
```



```r
plot(svm.poly, Auto, as.formula(mpg ~ weight))
```

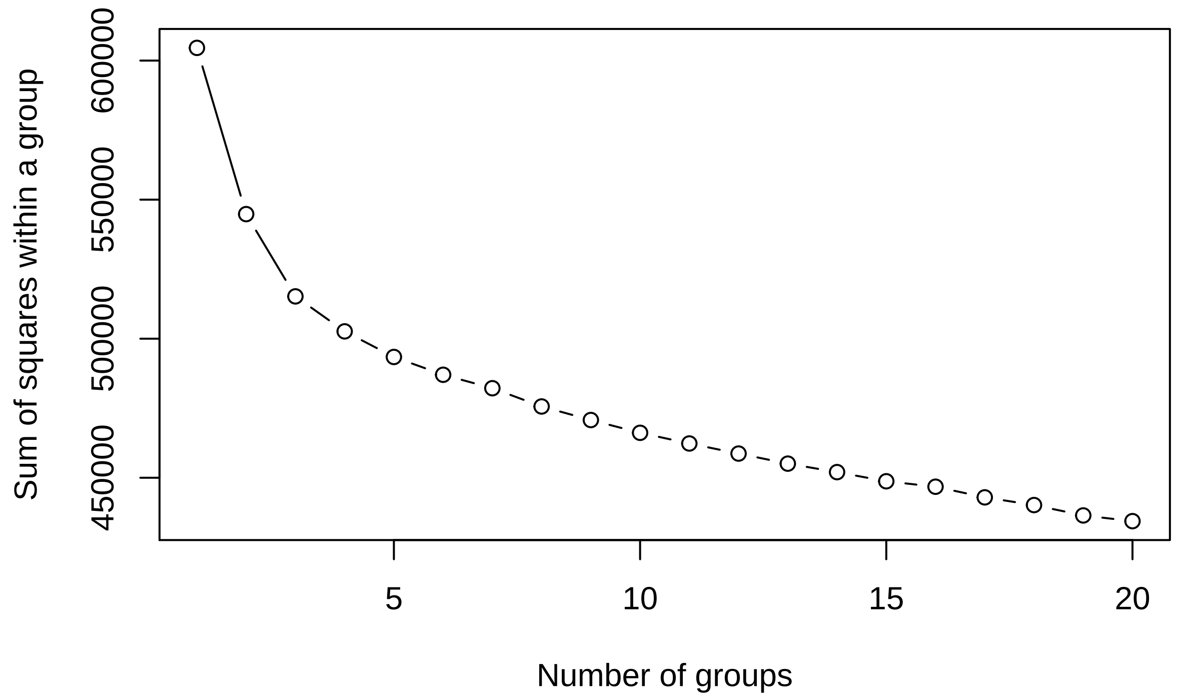


2. (K-Means Clustering, PCA and MDS, *40 pt*) The following codes read in a gene expression data from the TCGA project, which contains the expression of a random sample of 2000 genes for 563 patients from three cancer subtypes: Basal (`Basal`), Luminal A (`LumA`), and Luminal B (`LumB`). Suppose we are only interested in distinguishing Luminal A samples from Luminal B - but alas, we also have Basal samples, and we don't know which is which. Write a data analysis report to address the following problems.

```r
TCGA <- read.csv("TCGA_sample_2.txt", header = TRUE)

# Store the subtypes of tissue and the gene expression data
Subtypes <- TCGA[ ,1]
Gene <- as.matrix(TCGA[,-1])
```

(a) Run K-means for K from 1 to 20 and plot the associated within cluster sum of squares (WSSs). Comment the WSS at $K = 3$.

```r
wssplot <- function(data, nc=20, seed=123){
               wss <- (nrow(data)-1)*sum(apply(data,2,var))
               for (i in 2:nc){
                    set.seed(seed)
                    wss[i] <- sum(kmeans(data, centers=i)$withinss)}
                plot(1:nc, wss, type="b", xlab="Number of groups",
                     ylab="Sum of squares within a group")}

wssplot(Gene, nc = 20)
```

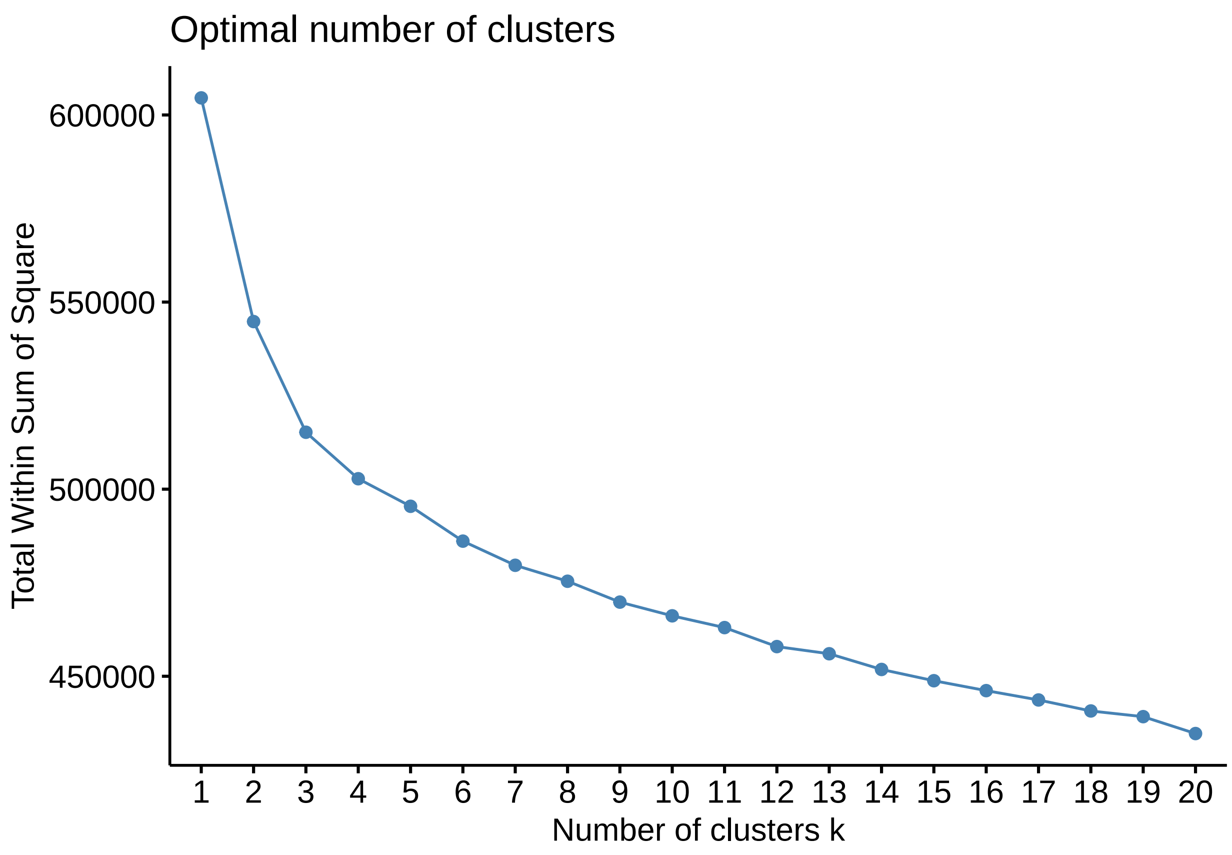


```r
#can also use the fviz_nbclust function
library(factoextra)
```

```
## Warning: package 'factoextra' was built under R version 3.6.2
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```r
fviz_nbclust(Gene, kmeans, k.max = 20, method = "wss")
```



When $K = 3$, the within cluster sum of squares is about 515213.5. The decresing rate is smaller after $K = 3$

(b) Apply K-means with $K = 3$ to the `Gene` dataset. What percentage of `Basal`, `LumA`, and `LumB` type samples are in each of the 3 resulting clusters? Did we do a good job distinguishing `LumA` from `LumB`? Confusion matrix of clusters versus subtypes might be helpful.

```
set.seed(1)
new.result <- kmeans(Gene,3)
#confusion matrix
table(true=Subtypes,pred=new.result$cluster)
```

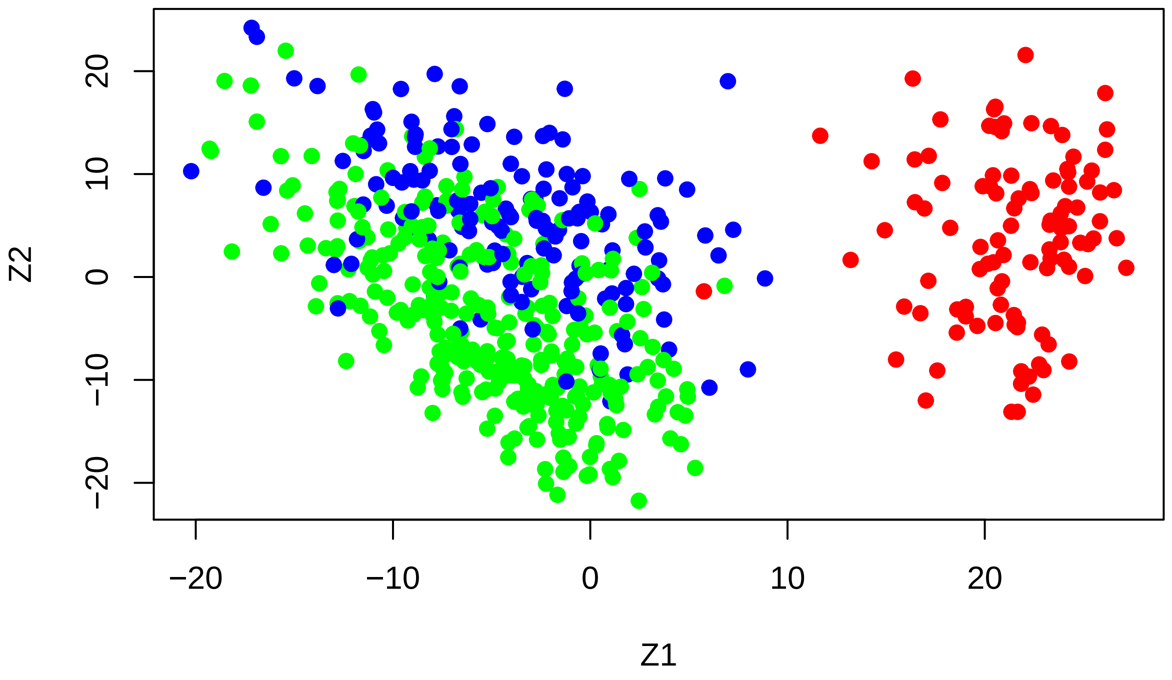
```
##        pred
## true      1   2   3
##   Basal 101   1   0
##   LumA    0 192 117
##   LumB    0  27 125
```

The first cluster only includes `Basal` type samples. The second resulting cluster includes 0.4% `Basal` type, 12.3% `LumB` type and 87.3% `LumA` type. The third resulting type includes 48.3% `LumA` type and 51.7% `LumB` type.

We might conclude that we did not do a good job distinguishing `LumA` from `LumB`

(c) Now apply PCA to the `Gene` dataset. Plot the data in the first two PCs colored by `Subtypes`. Does this plot appear to separate the cancer subtypes well?

```r
pr.out <- prcomp(Gene, scale=FALSE)
Cols=function(vec){
  cols=rainbow(length(unique(vec)))
  return(cols[as.numeric(as.factor(vec))])
}
plot(pr.out$x[,1:2], col=Cols(Subtypes), pch=19,xlab="Z1",ylab="Z2")
```

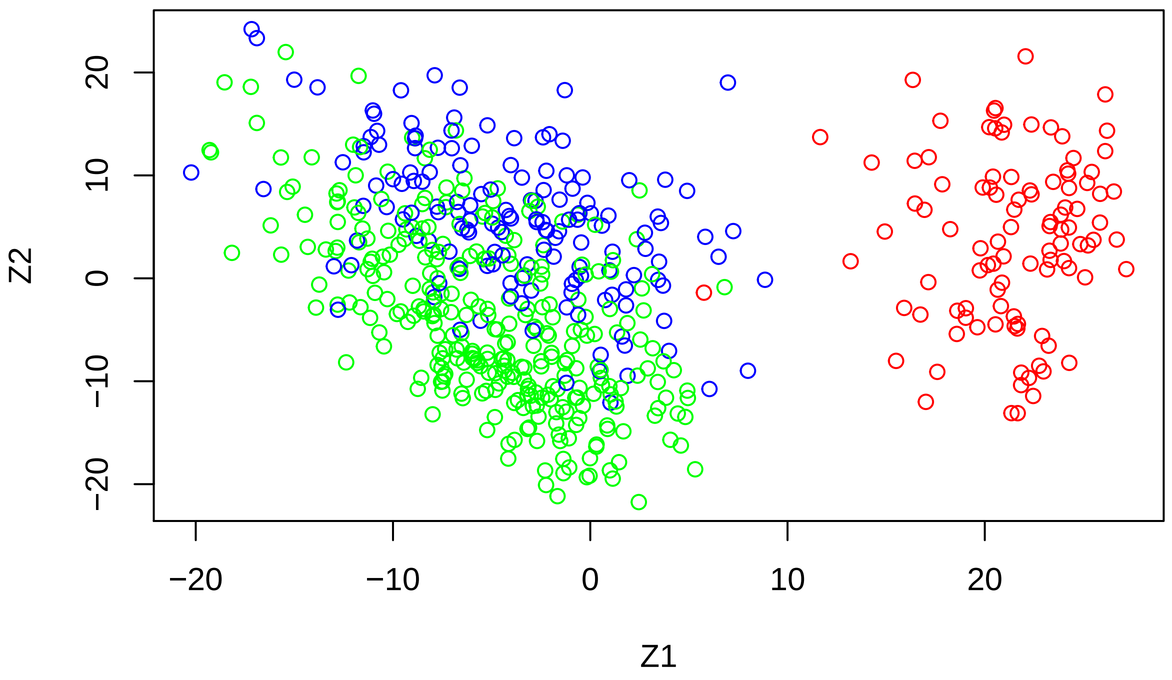


This plot does not appear to separate the cancer subtypes well because the blue dots and greens dots overlap a lot.

(d) Try plotting some more PC combinations. Can you find a pair of PCs that appear to separate all three subtypes well? Report the scatterplot of the data for pair of PCs that you think best separates all three types.

After several try of different combinations, I assume the three subtypes cannot be separated with clear bound. I think the best pair of PCs that can separate all three types is the first two, just the same as that in (c).

```r
plot(pr.out$x[,c(1, 2)], col=Cols(Subtypes), pch=1,xlab="Z1",ylab="Z2")
```



(e) Perform K-means with $K = 3$ on the pair of PCs identified in (d). Report the confusion matrix and make some comments.

```
set.seed(1)
km.out=kmeans(pr.out$x[,c(1, 2)], 3, nstart=20)
km.clusters=km.out$cluster

table(km.clusters,Subtypes)
```

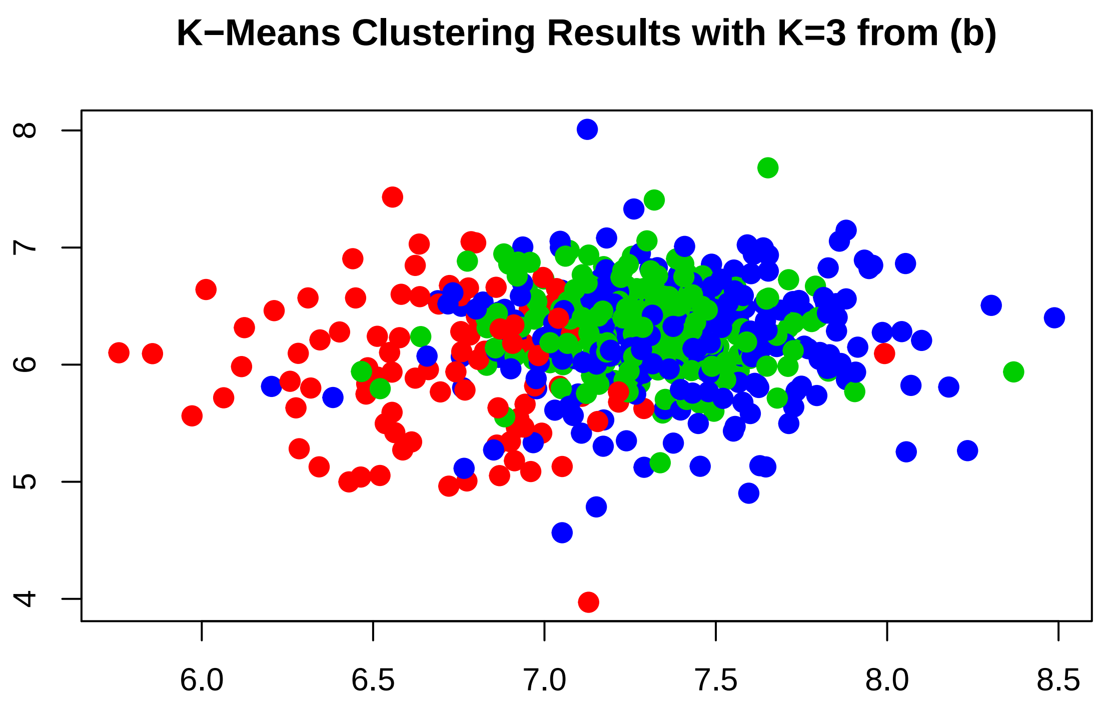
```
##            Subtypes
## km.clusters Basal LumA LumB
##           1   101    0    2
##           2     1  195   33
##           3     0  114  117
```

The 101 Basal type samples are separated to cluster1, while the other 1 is separated to cluster2. So we conclude that Basal type can be separated from the other 2 types clearly.

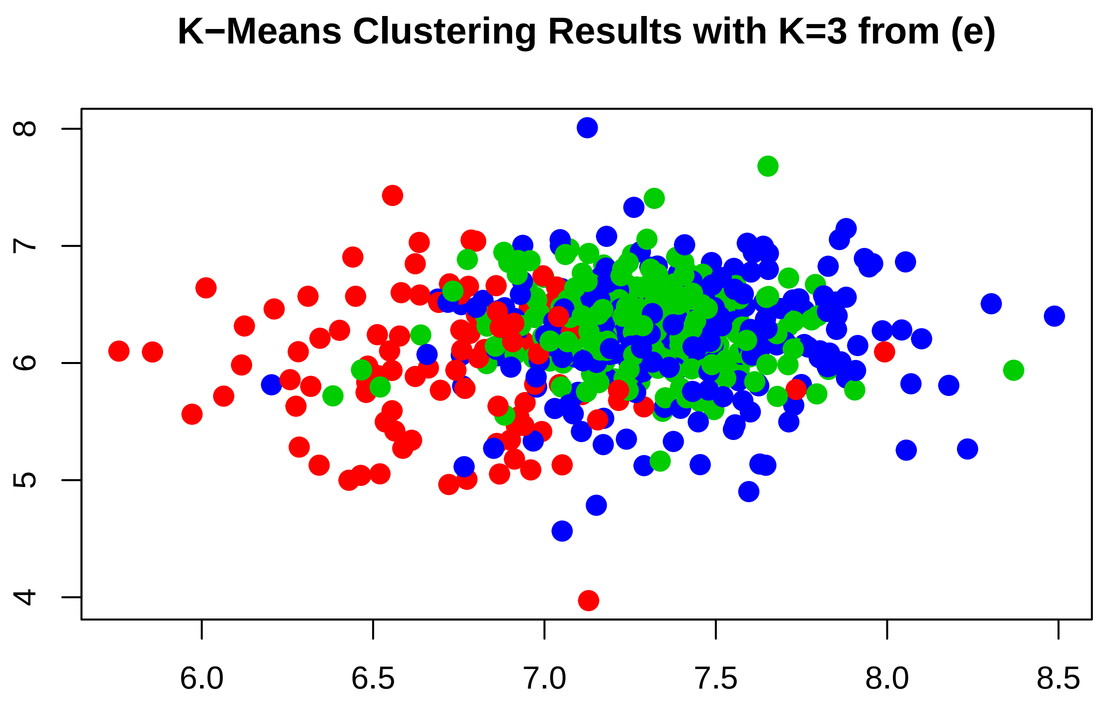
The 195 LumA type samples are separated to cluster2, while the other 114 are separated to cluster3. The 2 LumB type samples are separated to cluster1, the 33 LumB type samples are separated to cluster 2, while the other 117 are separated to cluster3. We still cannot distinguish LumA type samples from LumB type samples.

(f) Create two plots colored by the clusters found in (b) and in (e) respectively. Do they look similarly or differently? Explain why using PCA to reduce the number of dimensions from 2000 to 2 did not significantly change the results of K-means.

```
#from (b)
set.seed(1)
plot(Gene, col = (new.result$cluster+1), main = "K-Means Clustering Results with K=3 from (b)",
     xlab="", ylab="", pch=20, cex=2)
```



```r
#from (e)
plot(Gene, col=(km.out$cluster+1), main="K-Means Clustering Results with K=3 from (e)",
     xlab="", ylab="", pch=20, cex=2)
```



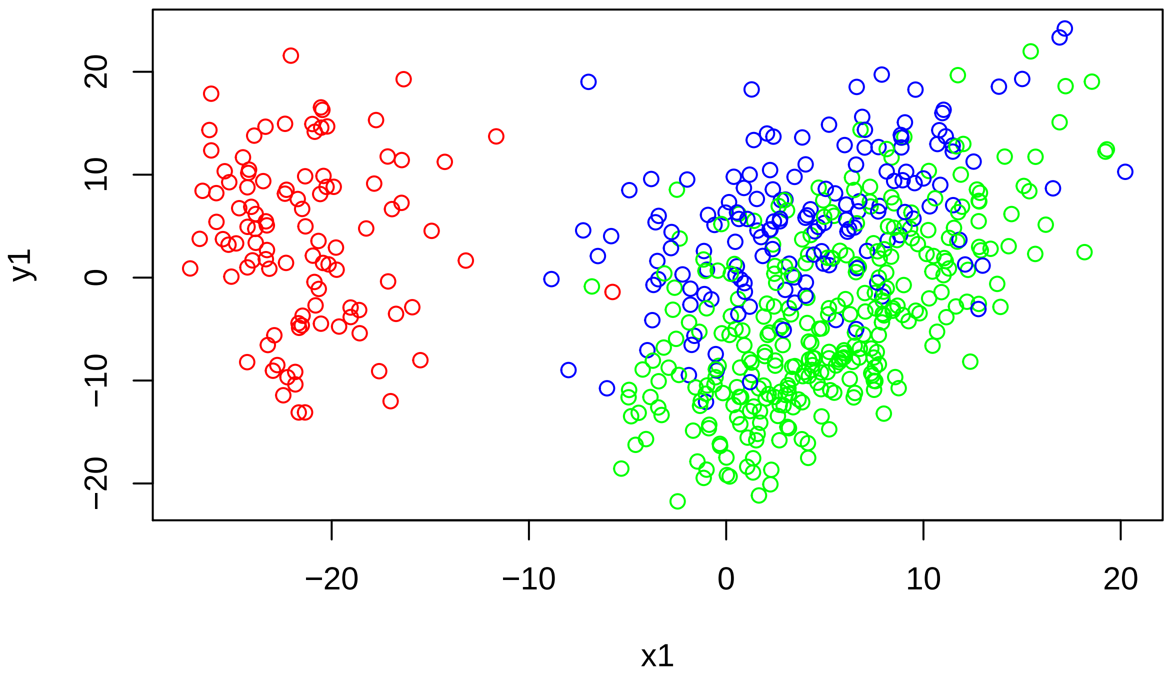
The two plots look similarly. It is because using PCA can only help us discard noise dimensions while still can reflect important information of the original data. K-means and PCA maximize the same objective function, with the only difference being that K-means has additional "categorical" constraint.

(g) Now apply MDS with various metrics and non-metric MDS to `Gene` to obtain 2-dimensional representations. Does any of them provide better separated scatterplot as compared to that from (d)? Notice that the Euclidean metric in MDS gives the same representation as PCA does.

```
library(MASS)
```

```
## Warning: package 'MASS' was built under R version 3.6.2

##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

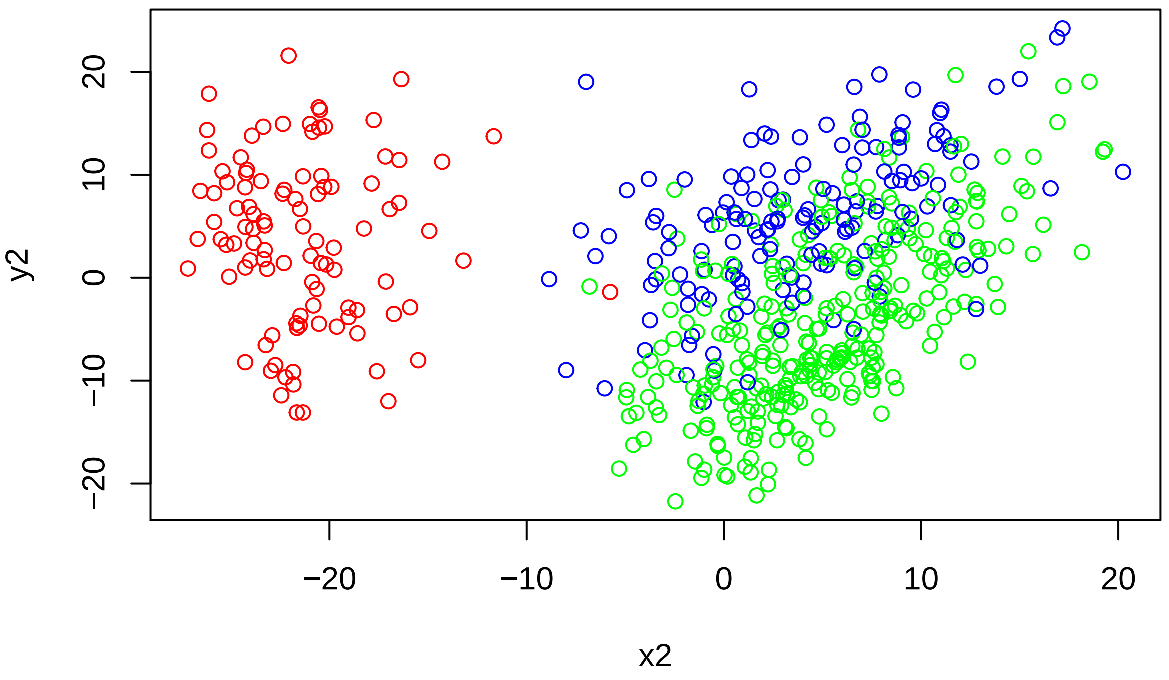
```
#various metrics
d <- dist(Gene)
fit1 <- cmdscale(d, k = 2)

# plot solution
x1 <- fit1[,1]
y1 <- fit1[,2]
plot(x1, y1, col = Cols(Subtypes))
```



```
#non-metric MDS
fit2 <- isoMDS(d, k = 2)
```

```
## initial  value 33.268145
## final  value 33.250852
## converged
```

```r
x2 <- fit2$points[,1]
y2 <- fit2$points[,2]
plot(x2, y2, col = Cols(Subtypes))
```



None of them provide better separated scatterplot as compared to that from (d). Actually, they have similar performance.

(h) Perform K-means with $K = 3$ on the new representations from (g) and report the confusion matrices. Compare them with that from (e).

```r
set.seed(1)
#various metrics
km.out=kmeans(fit1, 3, nstart=20)
km.clusters=km.out$cluster
table(true=Subtypes,pred=km.clusters)
```

```
##        pred
## true      1   2   3
##   Basal 101   1   0
##   LumA    0 195 114
##   LumB    2  33 117
```

```r
#non-metric MDS
set.seed(1)
km.out=kmeans(fit2$points, 3, nstart=20)
km.clusters=km.out$cluster
table(true=Subtypes,pred=km.clusters)
```

```
##        pred
## true      1   2   3
##   Basal 101   1   0
##   LumA    0 195 114
##   LumB    2  33 117
```

The 2 confusion matrixes in this question are the same. And they are the same as the confusion matrix from (e).

(i) Suppose we might know that the first PC contains information we aren't interested in. Apply K-means with $K = 3$ to `Gene` dataset **subtracting the approximation from the first PC**. Report the confusion matrix and make some comments.

```
set.seed(1)
km.out = kmeans(pr.out$x[,1] - pr.out$x[,c(1:563)], 3, nstart = 20)
table(true=Subtypes, pred=km.out$cluster)
```

```
##        pred
## true      1   2   3
##   Basal   1 101   0
##   LumA  158   0 151
##   LumB   89   0  63
```

In the first resulting cluster, there is 1 Basal type sample, 158 LumA type samples and 89 LumB type samples. The second resulting cluster only include 101 Basal type samples. The third resulting cluster includes 151 LumA type samples and 63 LumB type samples. So we still cannot separate LumA type from LumB pretty well.

Suppose we just drop the first PC, the confusion matrix is:

```
set.seed(1)
km.out = kmeans(pr.out$x[,-1], 3, nstart = 20)
table(true=Subtypes, pred=km.out$cluster)
```

```
##        pred
## true      1   2   3
##   Basal  26  56  20
##   LumA  172  52  85
##   LumB   12  60  80
```

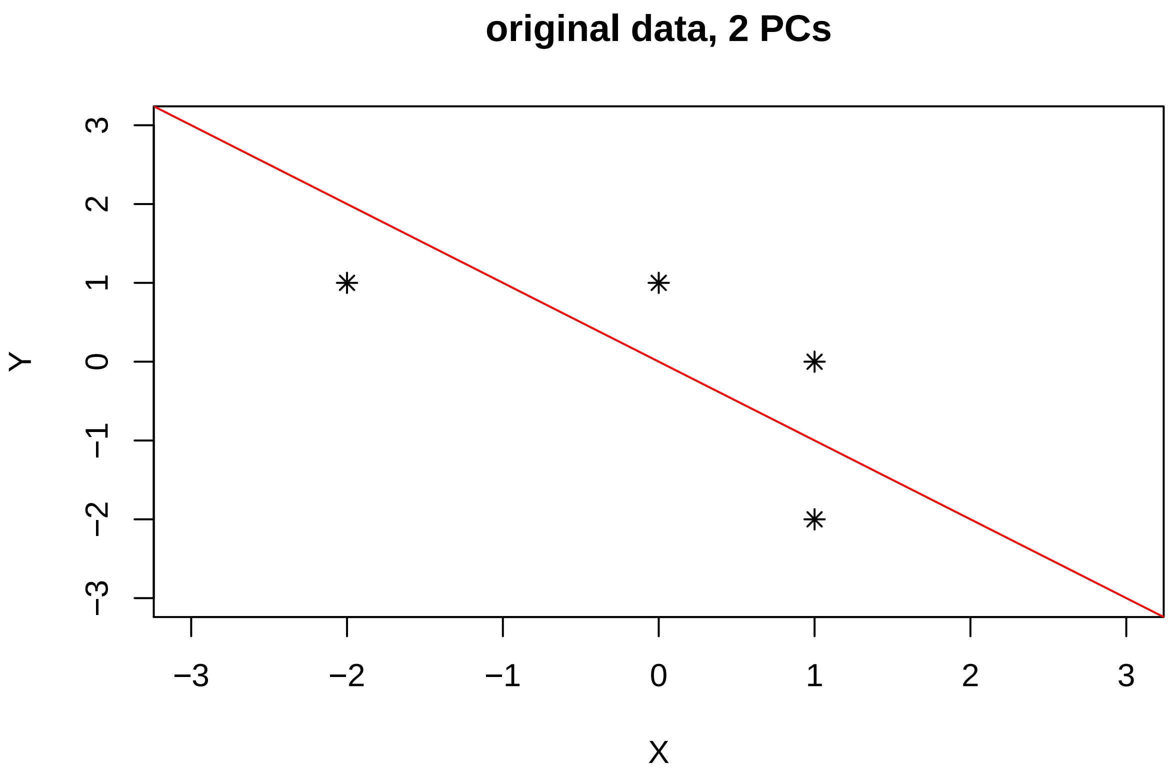
The classification has really bad performance.

CODE FROM THEORY PART:

(d)

```
X = c(0, -2, 1, 1)
Y = c(1, 1,-2, 0)
covariance = cov(data.frame(X,Y))

# grab the eigenvalues
lambda1 = eigen(covariance)$values[1]
lambda2 = eigen(covariance)$values[2]
#lambda1/(lambda1 + lambda2)
```

```r
# grab the eigenvectors
v1 = eigen(covariance)$vectors[,1]
v2 = eigen(covariance)$vectors[,2]

r = seq(-4,4,0.1)
# plot projection of original data into new basis
dim1 = t(v1)%*%t(as.matrix(data.frame(X,Y)))
dim2 = t(v2)%*%t(as.matrix(data.frame(X,Y)))
projection = cbind(t(dim1),t(dim2))

plot(data.frame(X,Y),xlim=c(-3,3),ylim=c(-3,3),pch=8,main='original data, 2 PCs')
lines(y=v1[2]/v1[1]*r,x=r,type='l',col='red')
```

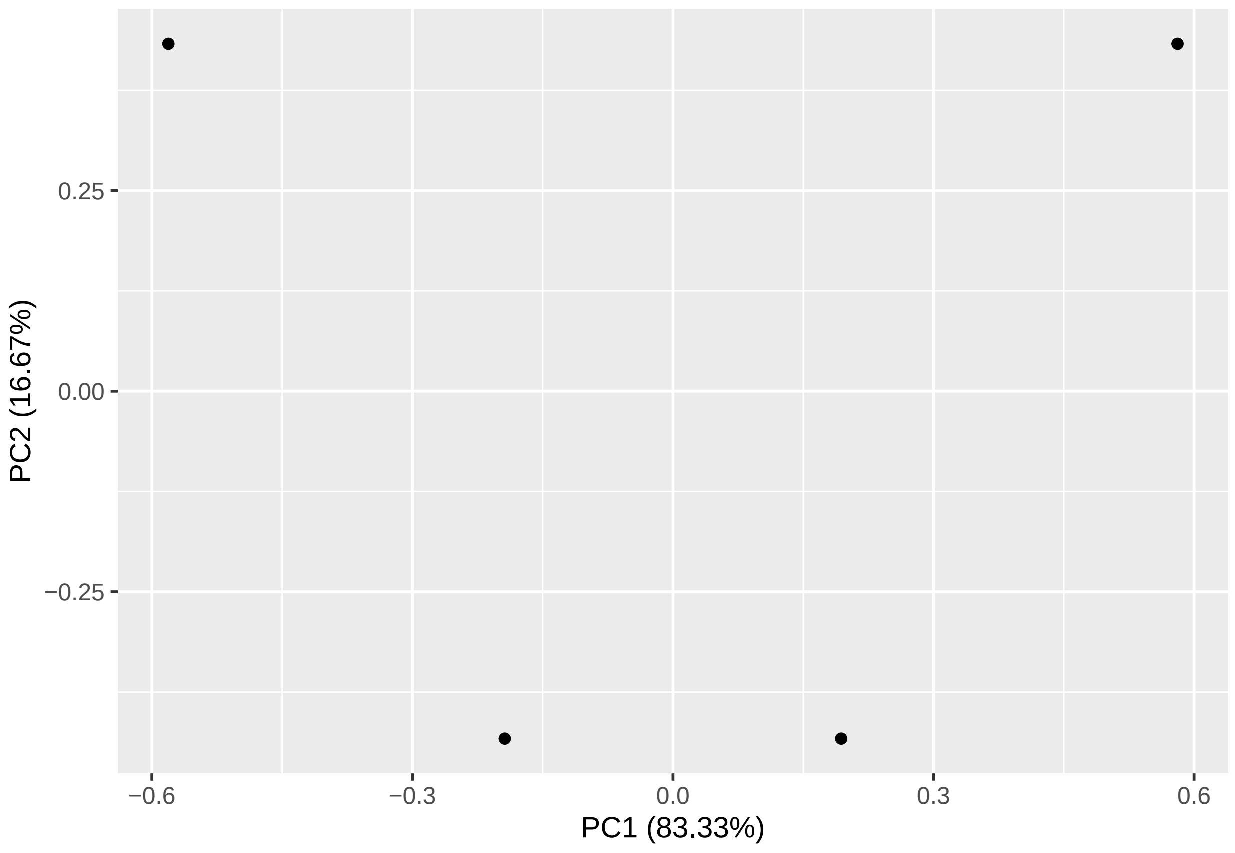


(e)

```r
#install.packages("ggfortify")
library(ggfortify)
```

```
## Warning: package 'ggfortify' was built under R version 3.6.2
```

```r
df <- data.frame(X,Y)
pca_res <- prcomp(df, scale. = TRUE)
autoplot(pca_res)
```



(g)

```
X = c(0, -2, 1, 1)
Y = c(10, 10,-20, 0)
covariance = cov(data.frame(X,Y))

# grab the eigenvalues
lambda1 = eigen(covariance)$values[1]
lambda2 = eigen(covariance)$values[2]

lambda1/(lambda1 + lambda2)
```

```
## [1] 0.9945239
```

```
# grab the eigenvectors
(v1 = eigen(covariance)$vectors[,1])
```

```
## [1] -0.06688731  0.99776054
```

```
(v2 = eigen(covariance)$vectors[,2])
```

```
## [1] -0.99776054 -0.06688731
```

```r
b <- cbind(c(0,-2,1,1),c(10,10,-20,-0))
#install.packages("factoextra")
library(factoextra)
(res.pca <- prcomp(b, scale = FALSE))
```

```
## Standard deviations (1, .., p=2):
## [1] 14.173702  1.051745
##
## Rotation (n x k) = (2 x 2):
##              PC1        PC2
## [1,]  0.06688731 0.99776054
## [2,] -0.99776054 0.06688731
```