PSTAT131_HW5

Niveditha Lakshminarayanan, Celeste Herrera

12/14/2020

```
#install.packages("ggridges")
#install.packages("dendextend")

library(tidyverse)
library(ROCR)
library(ggridges)
library(dendextend)
library(e1071)
```

1. Clustering and dimension reduction of for gene expression

This problem involves the analysis of gene expession data from 327 subjects from Yeoh *et al* (2002). The data set includes abundance levels for 3141 genes and a class label indicating one of 7 leukemia subtypes the patient was diagnosed with.

```
setwd("/Users/Nivi/Documents/UCSB 2020-21/Fall 2020/PSTAT 131")
leukemia_data <- read_csv("leukemia_data.csv")
```

(a) The class of the first column of `leukemia_data`, `Type` is set to `character` by default. Convert the `Type` column to a factor using the `mutate` function. Use the `table` command to print the number of patients with each leukemia subtype. Which leukemia subtype occurs the least in the data?

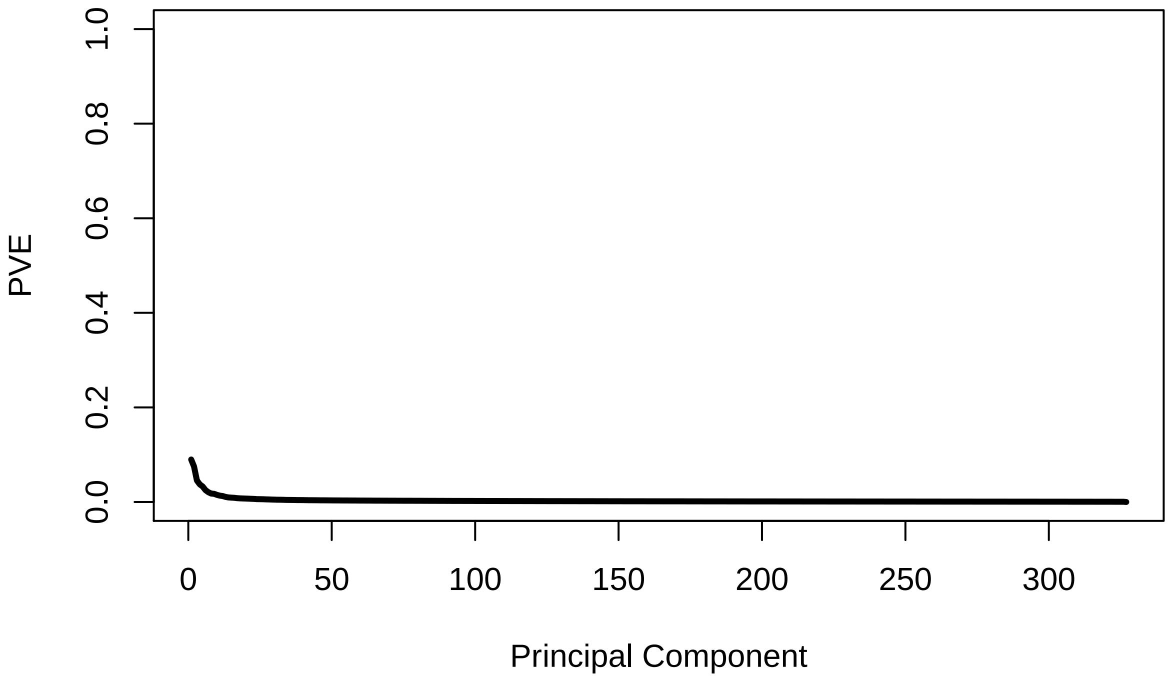
```
leukemia_data <- leukemia_data %>%
  mutate(Type=factor(Type))
table(leukemia_data$Type)
```

```
##
##    BCR-ABL   E2A-PBX1 Hyperdip50        MLL     OTHERS      T-ALL   TEL-AML1
##         15         27         64         20         79         43         79
```

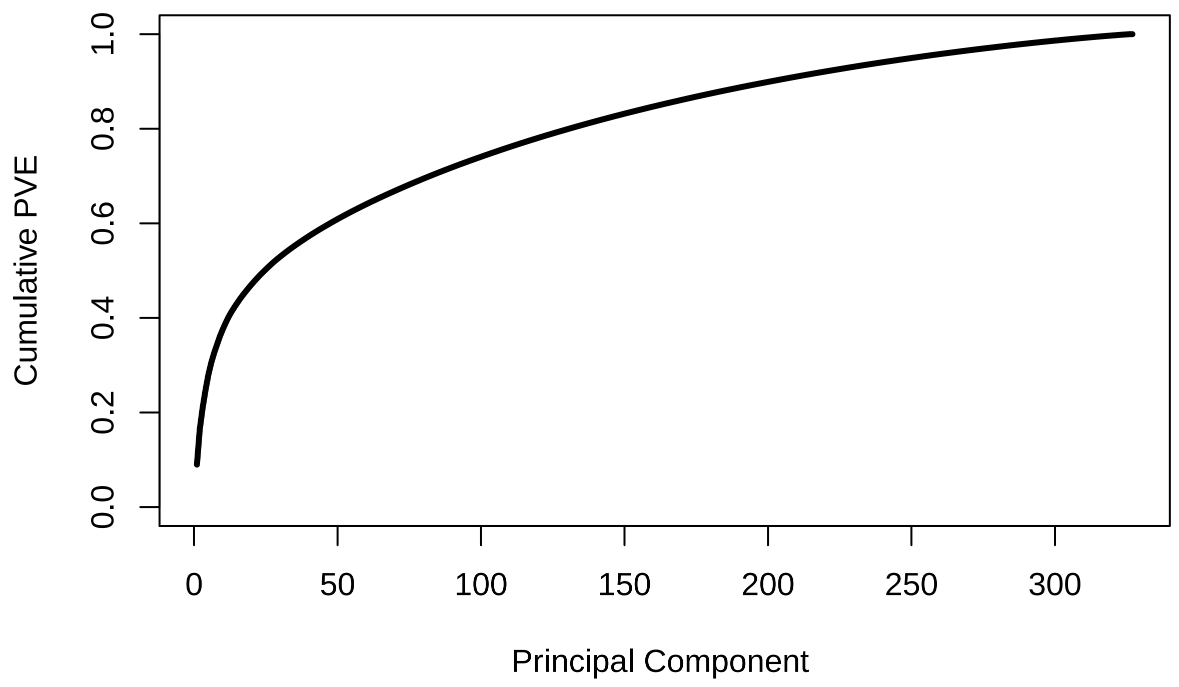
We can see that the BCR-ABL leukemia subtype occurs the least in the data.

(b) Run PCA on the leukemia data using `prcomp` function with `scale=TRUE` and `center=TRUE` (this scales each gene to have mean 0 and variance 1). Make sure you exclude the `Type` column when you run the PCA function (we are only interested in the unsupervised learning regime for now, where our focus is on reducing the dimension of the gene expression values). Plot the proportion of variance explained by each principal component (PVE) and the cumulative PVE. How many PC's do we need in order to explain 90% of the total variation in the data?

```
#PCA - without Type column
pr.out=prcomp(subset(leukemia_data, select = -c(Type)), scale=TRUE, center=TRUE)

#PVE
pr.var=(pr.out$sdev)^2
pve=pr.var/sum(pr.var)

#plot PVE
plot(pve, type="l", lwd=3, xlab="Principal Component", ylab="PVE", ylim=c(0,1))
```



```
#cumulative PVE
cumulative_pve <- cumsum(pve)

#plot cumulative PVE
plot(cumulative_pve, type="l", lwd=3, xlab="Principal Component ", ylab=" Cumulative PVE ", ylim=c(0,1)
```



```r
which(cumulative_pve >= 0.9)[1]
```

```
## [1] 201
```

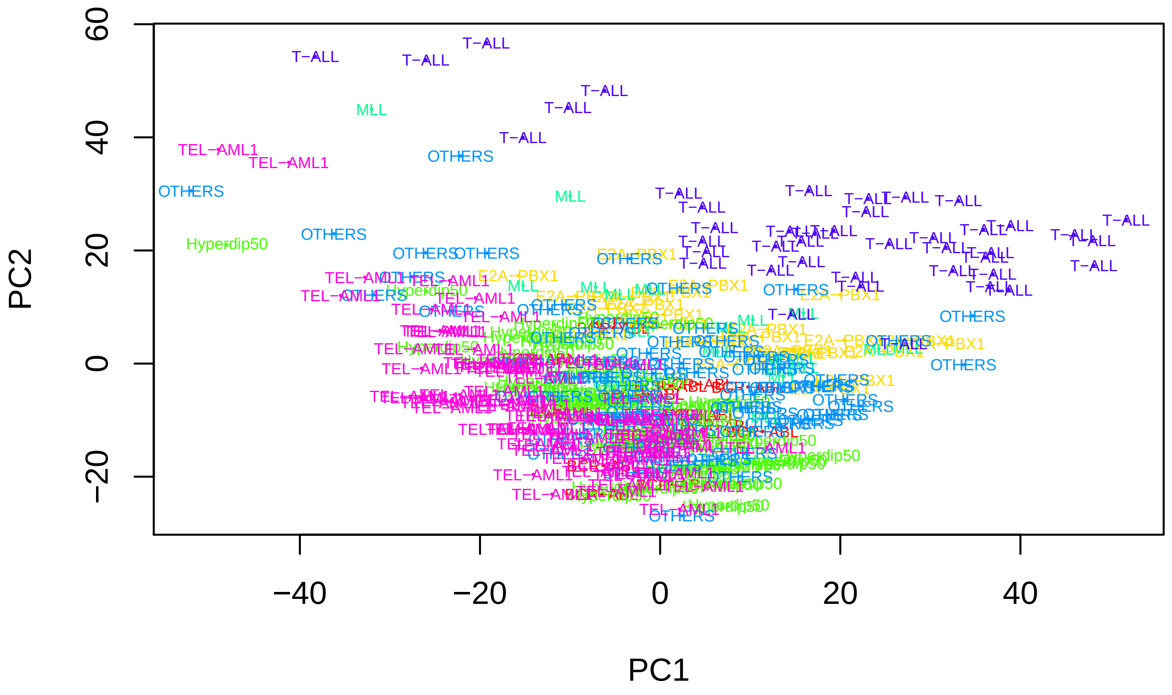
Since we have 327 principal components, it is difficult to tell from visual analysis of the graph how many PC's we need to explain 90% of the total variation in the data. So, we used the `which` function to find the first principal component with a cumulative PVE value closest to 0.9, and we yield that we need a minimum of 201 PC's to explain 90% of the total variation in the data.

(c) Use the results of PCA to project the data into the first 2 principal component dimensions, i.e., compute the scores vectors for the first two PCs. Recall in Lab 9, `prcomp` returns this dimension reduced data in the columns of `x`. Plot the data as a scatter plot using `plot` function with `col=plot_colors` where `plot_colors` is defined.

```r
rainbow_colors <- rainbow(7)
plot_colors <- rainbow_colors[leukemia_data$Type]
```

This will color the points according to the leukemia subtype. Add the leukemia type labels to the plot using `text` with `labels` argument set to the leukemia type and the col to `plot_colors` (it may help legibility to make the points on the plot very small by setting `cex` to a small number). Which group is most clearly separated from the others along the PC2 axis? Which genes have the highest absolute loadings for PC1 (the genes that have the largest (absolute) weights in the weighted average used to create the new variable PC1)? You can find these by taking the absolute values of the first principal component loadings and sorting them. Print the first 6 genes in this sorted vector using the `head` function.

```r
#plot
plot(pr.out$x[,1:2], col = plot_colors, cex=0.1)
text(pr.out$x[,1:2], labels = leukemia_data$Type, col = plot_colors, cex = 0.5)
```



```r
#genes w/ highest absolute loadings
head(sort(abs(pr.out$rotation[,1]), decreasing=TRUE))
```

```
##     SEMA3F       CCT2       LDHB      COX6C     SNRPD2       ELK3
## 0.04517148 0.04323818 0.04231619 0.04183480 0.04179822 0.04155821
```

The TEL-AML1 group seems most clearly separated along the PC2 axis, and the TEL-AML1, T-ALL, and OTHERS groups are most clearly separated along the PC1 axis. We can also see that the SEMA3F, CCT2, LDHB, COX6C, SNRPD2, and ELK3 genes have the highest absolute loadings for PC1.

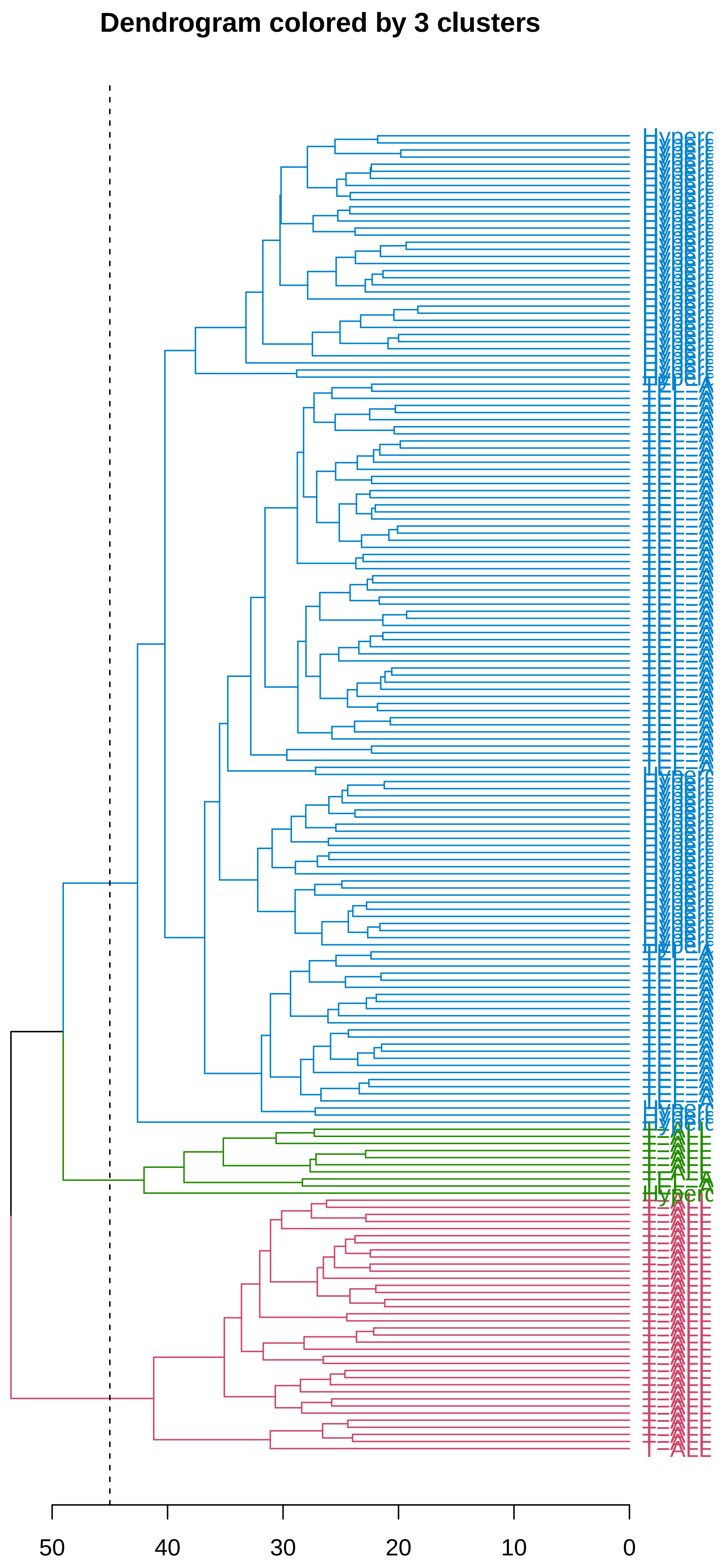
(f) Use the `filter` command to create a new dataframe (or a tibble if you'd like) `leukemia_subset` by subsetting to include only rows for which `Type` is either `T-ALL`, `TEL-AM1`, or `Hyperdip50`. Compute a euclidean distance matrix between the subjects using the `dist` function and then run hierarchical clustering using complete linkage. Plot 2 dendrograms based on the hierarchical clustering result. In the first plot, force 3 leukemia types to be the labels of the terminal nodes on the right. In the second plot, do all the same things except that this time color all the branches and labels to have 5 groups. Please make sure library `dendextend` is installed. (Hint: see Lab 8).

```r
#create leukemia_subset
leukemia_subset <- data.frame(filter(leukemia_data,
                                     Type == 'T-ALL' | Type == 'TEL-AML1' | Type == 'Hyperdip50'))

#euclidean distance matrix
leukDist <- dist(leukemia_subset)
```

```
## Warning in dist(leukemia_subset): NAs introduced by coercion
```

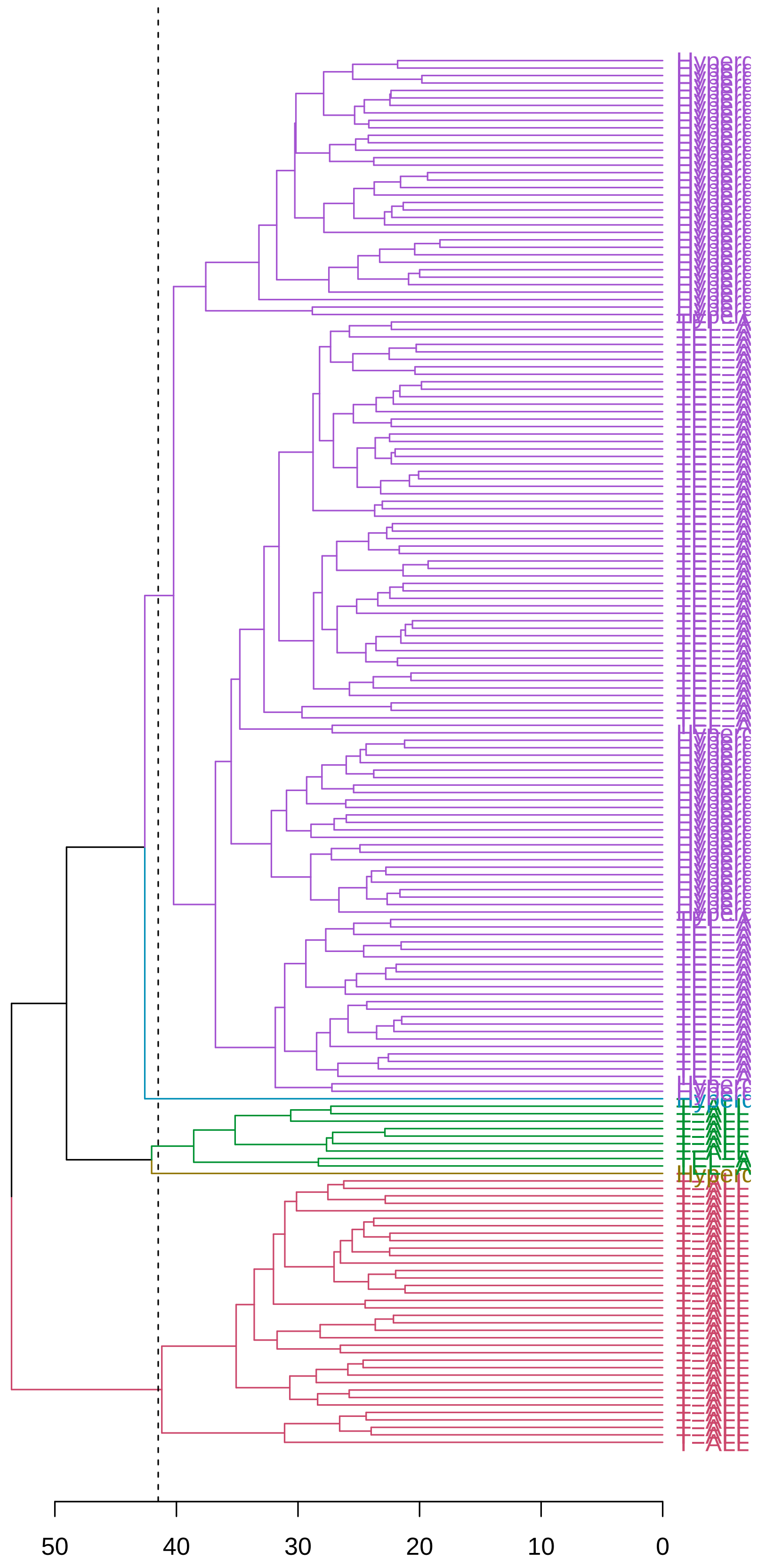
```r
#hierarchical clustering w/ complete linkage
set.seed(123)
leukHclust <- hclust(leukDist)
```

```r
library(dendextend)
dend <- as.dendrogram(leukHclust)
dend <- color_branches(dend, k = 3)
dend <- color_labels(dend, k = 3)
dend = set_labels(dend, labels=leukemia_subset$Type[order.dendrogram(dend)])
plot(dend, horiz = TRUE, main="Dendrogram colored by 3 clusters", cex=0.3)
abline(v = 45, lty=2)
```


Dendrogram colored by 3 clusters
50
40
30
20
10
0

```r
dend2 <- as.dendrogram(leukHclust)
dend2 <- color_branches(dend2, k = 5)
dend2 <- color_labels(dend2, k=5)
dend2 = set_labels(dend2, labels=leukemia_subset$Type[order.dendrogram(dend)])
plot(dend2, horiz = TRUE, main="Dendrogram colored by 5 clusters", cex=0.3)
abline(v = 41.5, lty=2)
```

Dendrogram colored by 5 clusters



```r
cutree(dend2, k=5)
```

```
## 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6
## 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 4 4 4 4 5 4 4 4 4 4 5 5 4 5 4 4
## 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 7 7 7 7 7 7 7 7 7 7 7 7 7 7
## 4 4 4 4 5 4 4 5 4 4 4 4 5 4 4 4 4 4 4 4 4 4 4 4 4 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
## 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
## 1 5 1 1 1 1 1 1 1 5 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

2. An SVMs prediction of drug use

```r
setwd("/Users/Nivi/Documents/UCSB 2020-21/Fall 2020/PSTAT 131")
drug <- read.csv('drug.csv', header = FALSE,
col.names=c('ID','Age','Gender','Education','Country',
'Ethnicity','Nscore',
'Escore','Oscore','Ascore','Cscore',
'Impulsive','SS','Alcohol','Amphetamines','Amylnitrite','Benzodiazepine',
'Caffeine','Cannabis', 'Chocolate','Cocaine','Crack','Ecstasy',
'Heroin','Ketamine','LegalHighs','LSD','Methadone',
'Mushrooms','Nicotine','Semeron','VolatileSubstanceAbuse'))
```

(a) Split the data into training and test data. Use a random sample of 1000 observations for the training data and the rest as test data. Use a support vector machine to predict `recent_cannabis_use` using only the subset of predictors between `Age` and `SS` variables as in homework 4. Use a "radial" kernel and a cost of 1. Generate and print the confusion matrix of the predictions against the test data.

```r
library(e1071)
set.seed(123)

#add recent_cannabis_use variable
drug = drug %>% mutate(recent_cannabis_use = factor(ifelse((Cannabis == "CL0" |
        Cannabis == "CL1" | Cannabis == "CL2"), "No", "Yes"), levels = c("No", "Yes")))

#subset predictors
drug2 = drug %>% select(-ID) %>% select(!(Alcohol:VolatileSubstanceAbuse))

#training set
train = sample(1:nrow(drug2), 1000)
drug.train = drug2[train,]

#test set
drug.test = drug2[-train,]

#SVM
drug_svm = svm(recent_cannabis_use ~., data=drug.train, kernel='radial',cost=1,scale=FALSE)
```

```r
#predict on test set
ypred = predict(drug_svm, drug.test)

#confusion matrix
table(predict=ypred, truth=drug.test$recent_cannabis_use)
```

```
##        truth
## predict  No Yes
##     No  338  80
##     Yes  85 382
```

(b) Use the `tune` function to perform cross validation over the set of cost parameters: `cost=c(0.001,0.01,0.1,1,10,100)`. What is the optimal cost and corresponding cross validated training error for this model? Print the confusion matrix for the best model. The best model can be found in the `best.model` variable returned by `tune`.

```r
#cross-validation with tune
set.seed(123)
drug_tune = tune(svm, recent_cannabis_use ~.,
                 data=drug.train, kernel='radial',
            ranges=list(cost=c(0.001,0.01,0.1,1,10,100)))
summary(drug_tune)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##   0.1
##
## - best performance: 0.188
##
## - Detailed performance results:
##    cost error dispersion
## 1 1e-03 0.463 0.05292552
## 2 1e-02 0.377 0.07528465
## 3 1e-01 0.188 0.04211096
## 4 1e+00 0.190 0.02748737
## 5 1e+01 0.205 0.03341656
## 6 1e+02 0.246 0.03272783
```

We can see that the optimal cost is 0.1, since that results in the lowest cross-validated training error rate of 0.188. So, we know that this is the cost associated with the best model, and can now obtain its resulting confusion matrix.

```r
#best model
best.model=drug_tune$best.model
summary(best.model)
```

```
##
## Call:
## best.tune(method = svm, train.x = recent_cannabis_use ~ ., data = drug.train,
##     ranges = list(cost = c(0.001, 0.01, 0.1, 1, 10, 100)), kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  0.1
##
## Number of Support Vectors:  630
##
##  ( 315 315 )
##
##
## Number of Classes:  2
##
## Levels:
##  No Yes
```

```
#best model confusion matrix
ypred_best = predict(best.model, drug.test)
table(predict=ypred_best, truth=drug.test$recent_cannabis_use)
```

```
##        truth
## predict  No Yes
##     No  340  74
##     Yes  83 388
```

We can see that this is a much stronger confusion matrix than the one from part (a), as the number of false positives and false negatives are both significantly lower with the best model.