Jin Kweon - 3032235207 - hw 5

Jin Kweon

11/20/2017

Problem 1

Data import

```
getwd()
```

```
## [1] "/Users/yjkweon24/Desktop/Cal/2017 Fall/PB HLTH 245/HW/HW5"
```

```
cancer <- read.table("Data-HW5-breastcancer.dat", header = T)
dim(cancer)
```

```
## [1] 569  31
```

```
#str(cancer)
#summary(cancer)
table(cancer$y)
```

```
##
##   0   1
## 357 212
```

Comment:

It contains information on 569 FNAs. There are two diagnoses (classes): 212 malignant and 357 benign. So, "y = 0" stands for benign and "y = 1" stands for malignant.

Part a) Partition the full data set into a training set of 400 patients, and a testing set of 169 patients. Please use the following command set.seed(1000) to set the random seed of your partition, so that you can reproduce all your analysis results.

```
set.seed(1000)
samplesize <- 169
testid <- sample(seq_len(nrow(cancer)), samplesize, replace = F)

train <- cancer[-testid, ]
test <- cancer[testid, ]
dim(train)
```

```
## [1] 400  31
```

```
dim(test)
```

```
## [1] 169  31
```

```r
#NA check
na <- c()
for(i in 1:ncol(cancer)){
  na[i] <- sum(is.na(cancer[,i]))
}
na
```

```
##  [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Comment:

If I sample 400 for training IDs, I will get different answers.

I randomly split into the 400 training and 169 testing sets. (I included the for loop of checking NA existence in our original data, just in case - good practice whenever you have a large data set)

Part b) Fit LDA, QDA, MDA (with number of subclasses equal to (5, 5)), Nearest Neighbor (with k = 5), and CART. Report the misclassification error rate on the testing data set.

```r
set.seed(1000)
#LDA
ldas <- lda(train[,-1], train[,1]) #or, lda(y ~., data = cancer[trainid,])

trainpredict <- predict(ldas, train[,-1])$class #predict on the training set
#trainpredict

testpredict <- predict(ldas, test[,-1])$class #predict on the testing set
#testpredict


table(trainpredict, train[,1]) #confusion matrix of training
```

```
##
## trainpredict   0   1
##            0 249   9
##            1   2 140
```

```r
table(testpredict, test[,1]) #confusion matrix of testing
```

```
##
## testpredict   0   1
##           0 106   8
##           1   0  55
```

```r
ldatrainmis <- sum(trainpredict != train[,1]) / length(train[,1])
ldatestmis <- sum(testpredict != test[,1]) / length(test[,1])

paste("The misclassfication error rate on the training set for LDA is", round(ldatrainmis, 5))
```

```
## [1] "The misclassfication error rate on the training set for LDA is 0.0275"
```

```r
paste("The misclassfication error rate on the testing set for LDA is", round(ldatestmis, 5))
```

```
## [1] "The misclassfication error rate on the testing set for LDA is 0.04734"
```

```r
#QDA
qdas <- qda(train[,-1], train[,1]) #or, qda(y ~., data = cancer[trainid,])

trainpredict2 <- predict(qdas, train[,-1])$class #predict on the training set
#trainpredict2

testpredict2 <- predict(qdas, test[,-1])$class #predict on the testing set
#testpredict2


table(trainpredict2, train[,1]) #confusion matrix of training
```

```
##
## trainpredict2   0   1
##             0 248   5
##             1   3 144
```

```r
table(testpredict2, test[,1]) #confusion matrix of testing
```

```
##
## testpredict2   0   1
##            0 102   4
##            1   4  59
```

```r
qdatrainmis <- sum(trainpredict2 != train[,1]) / length(train[,1])
qdatestmis <- sum(testpredict2 != test[,1]) / length(test[,1])

paste("The misclassfication error rate on the training set for QDA is", round(qdatrainmis, 5))
```

```
## [1] "The misclassfication error rate on the training set for QDA is 0.02"
```

```r
paste("The misclassfication error rate on the testing set for QDA is", round(qdatestmis, 5))
```

```
## [1] "The misclassfication error rate on the testing set for QDA is 0.04734"
```

```r
#MDA
mdas <- mda(y~., data = train, subclasses = c(5, 5))

trainpredict3 <- predict(mdas, train[,-1], type = "class") #predict on the training set
#trainpredict3

testpredict3 <- predict(mdas, test[,-1], type = "class") #predict on the testing set
#testpredict3


table(trainpredict3, train[,1]) #confusion matrix of training
```

```
##
## trainpredict3   0   1
##             0 248   5
##             1   3 144
```

```r
table(testpredict3, test[,1]) #confusion matrix of testing
```

```
##
## testpredict3   0   1
##            0 105   6
##            1   1  57
```

```r
mdatrainmis <- sum(trainpredict3 != train[,1]) / length(train[,1])
mdatestmis <- sum(testpredict3 != test[,1]) / length(test[,1])

paste("The misclassfication error rate on the training set for MDA is", round(mdatrainmis, 5))
```

```
## [1] "The misclassfication error rate on the training set for MDA is 0.02"
```

```r
paste("The misclassfication error rate on the testing set for MDA is", round(mdatestmis, 5))
```

```
## [1] "The misclassfication error rate on the testing set for MDA is 0.04142"
```

```r
#KNN with k = 5
k <- 5
trainpredict4 <- knn(train[,-1], train[,-1], train[,1], k = k)
#trainpredict4

testpredict4 <- knn(train[,-1], test[,-1], train[,1], k = k)
#testpredict4

table(trainpredict4, train[,1]) #confusion matrix of training
```

```
##
## trainpredict4   0   1
##             0 244  15
##             1   7 134
```

```r
table(testpredict4, test[,1]) #confusion matrix of testing
```

```
##
## testpredict4   0   1
##            0 100   6
##            1   6  57
```

```r
knntrainmis <- sum(trainpredict4 != train[,1]) / length(train[,1])
knntestmis <- sum(testpredict4 != test[,1]) / length(test[,1])

paste("The misclassfication error rate on the training set for KNN is", round(knntrainmis, 5))
```

```
## [1] "The misclassfication error rate on the training set for KNN is 0.055"
```

```r
paste("The misclassfication error rate on the testing set for KNN is", round(knntestmis, 5))
```

```
## [1] "The misclassfication error rate on the testing set for KNN is 0.07101"
```

```r
#CART (Classification And Regression Tree)
trees <- tree(as.factor(y) ~., data = train) #or, tree(as.factor(y) ~., data = cancer, subset = trainid
trees
```

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##  1) root 400 528.200 0 ( 0.627500 0.372500 )
##    2) x23 < 105.15 240  83.140 0 ( 0.958333 0.041667 )
##      4) x28 < 0.13505 223  12.810 0 ( 0.995516 0.004484 )
```

```
##        8) x11 < 0.5761 218   0.000 0 ( 1.000000 0.000000 ) *
##        9) x11 > 0.5761 5   5.004 0 ( 0.800000 0.200000 ) *
##      5) x28 > 0.13505 17  23.510 1 ( 0.470588 0.529412 )
##       10) x22 < 25.89 7   0.000 0 ( 1.000000 0.000000 ) *
##       11) x22 > 25.89 10   6.502 1 ( 0.100000 0.900000 ) *
##    3) x23 > 105.15 160 124.400 1 ( 0.131250 0.868750 )
##      6) x28 < 0.1434 41  56.810 1 ( 0.487805 0.512195 )
##       12) x21 < 16.805 12   0.000 0 ( 1.000000 0.000000 ) *
##       13) x21 > 16.805 29  34.160 1 ( 0.275862 0.724138 )
##         26) x2 < 16.375 6   0.000 0 ( 1.000000 0.000000 ) *
##         27) x2 > 16.375 23  13.590 1 ( 0.086957 0.913043 )
##           54) x27 < 0.2006 5   6.730 1 ( 0.400000 0.600000 ) *
##           55) x27 > 0.2006 18   0.000 1 ( 0.000000 1.000000 ) *
##      7) x28 > 0.1434 119  11.550 1 ( 0.008403 0.991597 )
##       14) x7 < 0.085885 5   5.004 1 ( 0.200000 0.800000 ) *
##       15) x7 > 0.085885 114   0.000 1 ( 0.000000 1.000000 ) *
```

```r
trainpredict5 <- predict(trees, newdata = train[,-1], type = "class")
#trainpredict5

testpredict5 <- predict(trees, newdata = test[,-1], type = "class")
#testpredict5

table(trainpredict5, train[,1]) #confusion matrix of training
```

```
##
## trainpredict5   0   1
##             0 247   1
##             1   4 148
```

```r
table(testpredict5, test[,1]) #confusion matrix of testing
```

```
##
## testpredict5   0   1
##            0 102   7
##            1   4  56
```

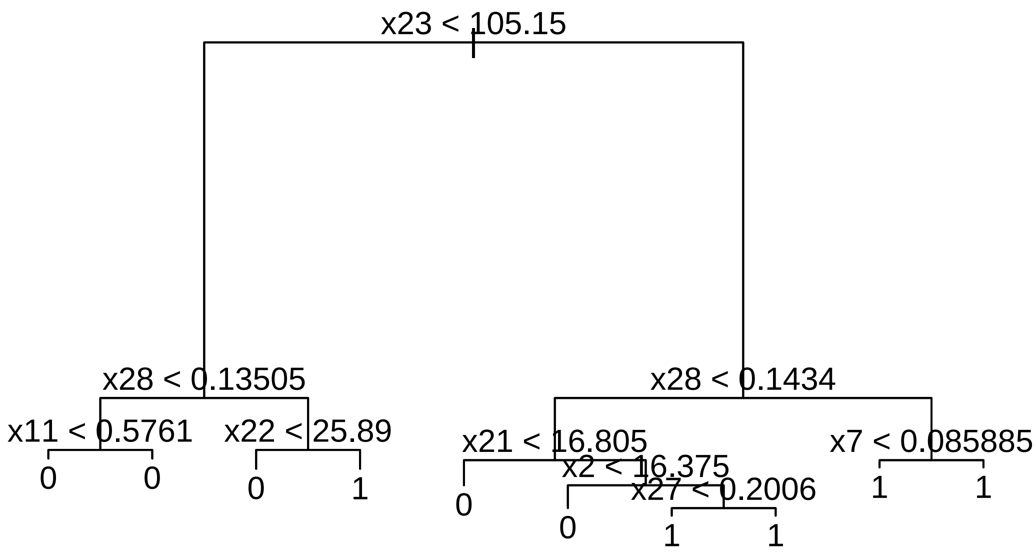
```r
treetrainmis <- sum(trainpredict5 != train[,1]) / length(train[,1])
treetestmis <- sum(testpredict5 != test[,1]) / length(test[,1])

paste("The misclassfication error rate on the training set for KNN is", round(treetrainmis, 5))
```

```
## [1] "The misclassfication error rate on the training set for KNN is 0.0125"
```

```r
paste("The misclassfication error rate on the testing set for KNN is", round(treetestmis, 5))
```

```
## [1] "The misclassfication error rate on the testing set for KNN is 0.06509"
```

```r
plot(trees)
text(trees, pretty = 0)
```



Comment:

I again set the seed with 1000.

I reported the misclassification error rate both on the training and testing data set above. Also, please be aware that misclassification error rate on testing set is what is important. Training and testing on the same data set is not a good practice to do…

For tree, we need to go for classification tree, as the y is categorial consisted with 1 and 0 only.

Just for the readers, I made confusion matrixes for each method. (So, you can actually see the sensitivity and specificity)

As they asked me to report the misclassification error rate on the testing data set, I printed out the test error rate for each method.

Part c) Fit MDA, with number of subclasses equal to (1, 1), (5, 5), and (10, 10), respectively. Report the misclassification error rate on both the training and the testing data. Please describe the pattern, and see if it agrees with your expectation.

```
set.seed(1000)
#(1,1)
mda1 <- mda(y~., data = train, subclasses = c(1, 1))

trainpredictmda1 <- predict(mda1, train[,-1], type = "class") #predict on the training set
#trainpredictmda1

testpredictmda1 <- predict(mda1, test[,-1], type = "class") #predict on the testing set
#testpredictmda1


table(trainpredictmda1, train[,1]) #confusion matrix of training
```

```
##
## trainpredictmda1   0   1
##                0 249   9
##                1   2 140
```

```r
table(testpredictmda1, test[,1]) #confusion matrix of testing
```

```
##
## testpredictmda1   0   1
##               0 106   8
##               1   0  55
```

```r
mdatrainmis1 <- sum(trainpredictmda1 != train[,1]) / length(train[,1])
mdatestmis1 <- sum(testpredictmda1 != test[,1]) / length(test[,1])

paste("The misclassfication error rate on the training set for MDA is", round(mdatrainmis1, 5))
```

```
## [1] "The misclassfication error rate on the training set for MDA is 0.0275"
```

```r
paste("The misclassfication error rate on the testing set for MDA is", round(mdatestmis1, 5))
```

```
## [1] "The misclassfication error rate on the testing set for MDA is 0.04734"
```

```r
#(3,3)
mda2 <- mda(y~., data = train, subclasses = c(3, 3))

trainpredictmda2 <- predict(mda2, train[,-1], type = "class") #predict on the training set
#trainpredictmda2

testpredictmda2 <- predict(mda2, test[,-1], type = "class") #predict on the testing set
#testpredictmda2


table(trainpredictmda2, train[,1]) #confusion matrix of training
```

```
##
## trainpredictmda2   0   1
##                0 249   6
##                1   2 143
```

```r
table(testpredictmda2, test[,1]) #confusion matrix of testing
```

```
##
## testpredictmda2   0   1
##               0 106   7
##               1   0  56
```

```r
mdatrainmis2 <- sum(trainpredictmda2 != train[,1]) / length(train[,1])
mdatestmis2 <- sum(testpredictmda2 != test[,1]) / length(test[,1])

paste("The misclassfication error rate on the training set for MDA is", round(mdatrainmis2, 5))
```

```
## [1] "The misclassfication error rate on the training set for MDA is 0.02"
```

```r
paste("The misclassfication error rate on the testing set for MDA is", round(mdatestmis2, 5))
```

```
## [1] "The misclassfication error rate on the testing set for MDA is 0.04142"
```

```r
#(5,5)
mda3 <- mda(y~., data = train, subclasses = c(5, 5))

trainpredictmda3 <- predict(mda3, train[,-1], type = "class") #predict on the training set
#trainpredictmda3

testpredictmda3 <- predict(mda3, test[,-1], type = "class") #predict on the testing set
#testpredictmda3


table(trainpredictmda3, train[,1]) #confusion matrix of training
```

```
##
## trainpredictmda3   0   1
##                0 249   6
##                1   2 143
```

```r
table(testpredictmda3, test[,1]) #confusion matrix of testing
```

```
##
## testpredictmda3   0   1
##               0 106   7
##               1   0  56
```

```r
mdatrainmis3 <- sum(trainpredictmda3 != train[,1]) / length(train[,1])
mdatestmis3 <- sum(testpredictmda3 != test[,1]) / length(test[,1])

paste("The misclassfication error rate on the training set for MDA is", round(mdatrainmis3, 5))
```

```
## [1] "The misclassfication error rate on the training set for MDA is 0.02"
```

```r
paste("The misclassfication error rate on the testing set for MDA is", round(mdatestmis3, 5))
```

```
## [1] "The misclassfication error rate on the testing set for MDA is 0.04142"
```

```r
#(10,10)
mda5 <- mda(y~., data = train, subclasses = c(10, 10))

trainpredictmda5 <- predict(mda5, train[,-1], type = "class") #predict on the training set
#trainpredictmda5

testpredictmda5 <- predict(mda5, test[,-1], type = "class") #predict on the testing set
#testpredictmda5


table(trainpredictmda5, train[,1]) #confusion matrix of training
```

```
##
## trainpredictmda5   0   1
##                0 250   7
##                1   1 142
```

```r
table(testpredictmda5, test[,1]) #confusion matrix of testing
```

```
##
## testpredictmda5   0   1
##               0 105   6
```

```
##                1   1  57
```

```r
mdatrainmis5 <- sum(trainpredictmda5 != train[,1]) / length(train[,1])
mdatestmis5 <- sum(testpredictmda5 != test[,1]) / length(test[,1])

paste("The misclassfication error rate on the training set for MDA is", round(mdatrainmis5, 5))
```

```
## [1] "The misclassfication error rate on the training set for MDA is 0.02"
```

```r
paste("The misclassfication error rate on the testing set for MDA is", round(mdatestmis5, 5))
```

```
## [1] "The misclassfication error rate on the testing set for MDA is 0.04142"
```

```r
#(13,13)
mda6 <- mda(y~., data = train, subclasses = c(13, 13))

trainpredictmda6 <- predict(mda6, train[,-1], type = "class") #predict on the training set
#trainpredictmda6

testpredictmda6 <- predict(mda6, test[,-1], type = "class") #predict on the testing set
#testpredictmda6


table(trainpredictmda6, train[,1]) #confusion matrix of training
```

```
##
## trainpredictmda6   0   1
##                 0 250   4
##                 1   1 145
```

```r
table(testpredictmda6, test[,1]) #confusion matrix of testing
```

```
##
## testpredictmda6   0   1
##                0 105   7
##                1   1  56
```

```r
mdatrainmis6 <- sum(trainpredictmda6 != train[,1]) / length(train[,1])
mdatestmis6 <- sum(testpredictmda6 != test[,1]) / length(test[,1])

paste("The misclassfication error rate on the training set for MDA is", round(mdatrainmis6, 5))
```

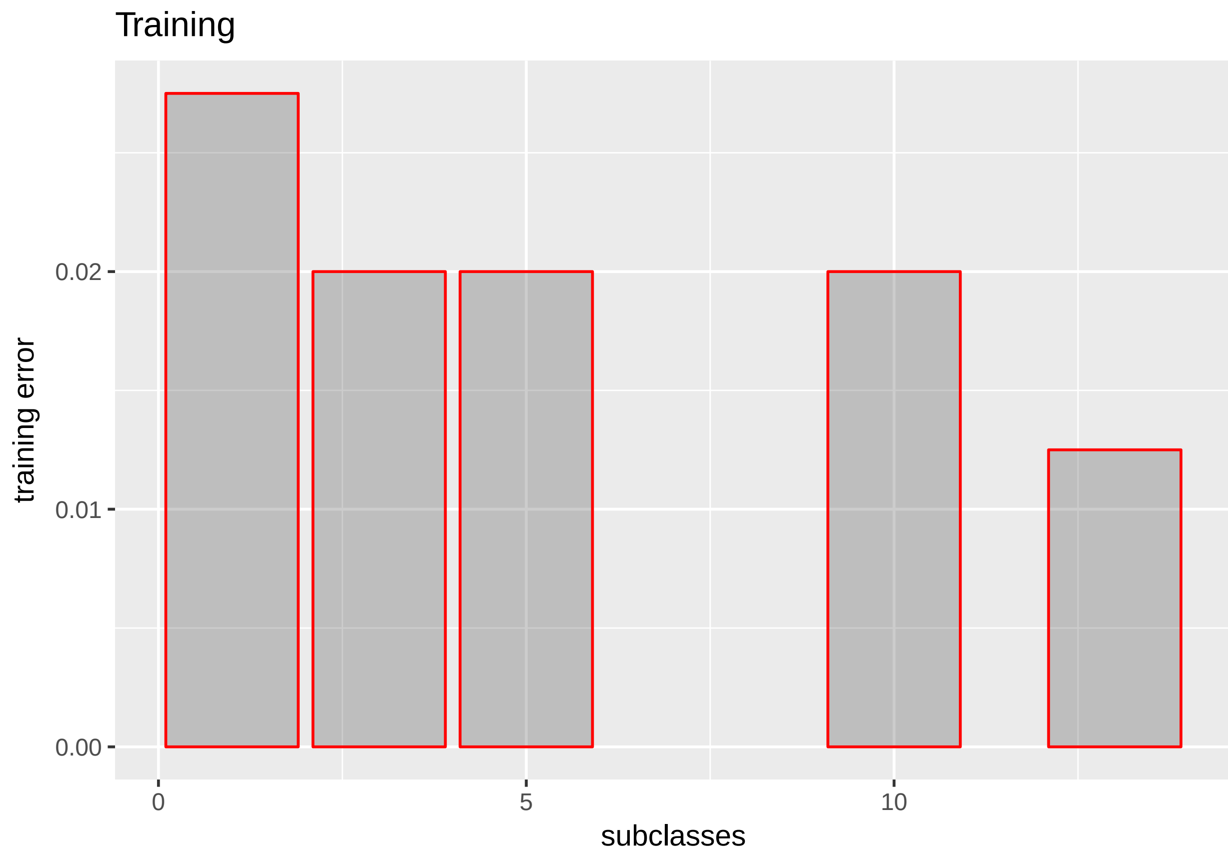
```
## [1] "The misclassfication error rate on the training set for MDA is 0.0125"
```

```r
paste("The misclassfication error rate on the testing set for MDA is", round(mdatestmis6, 5))
```

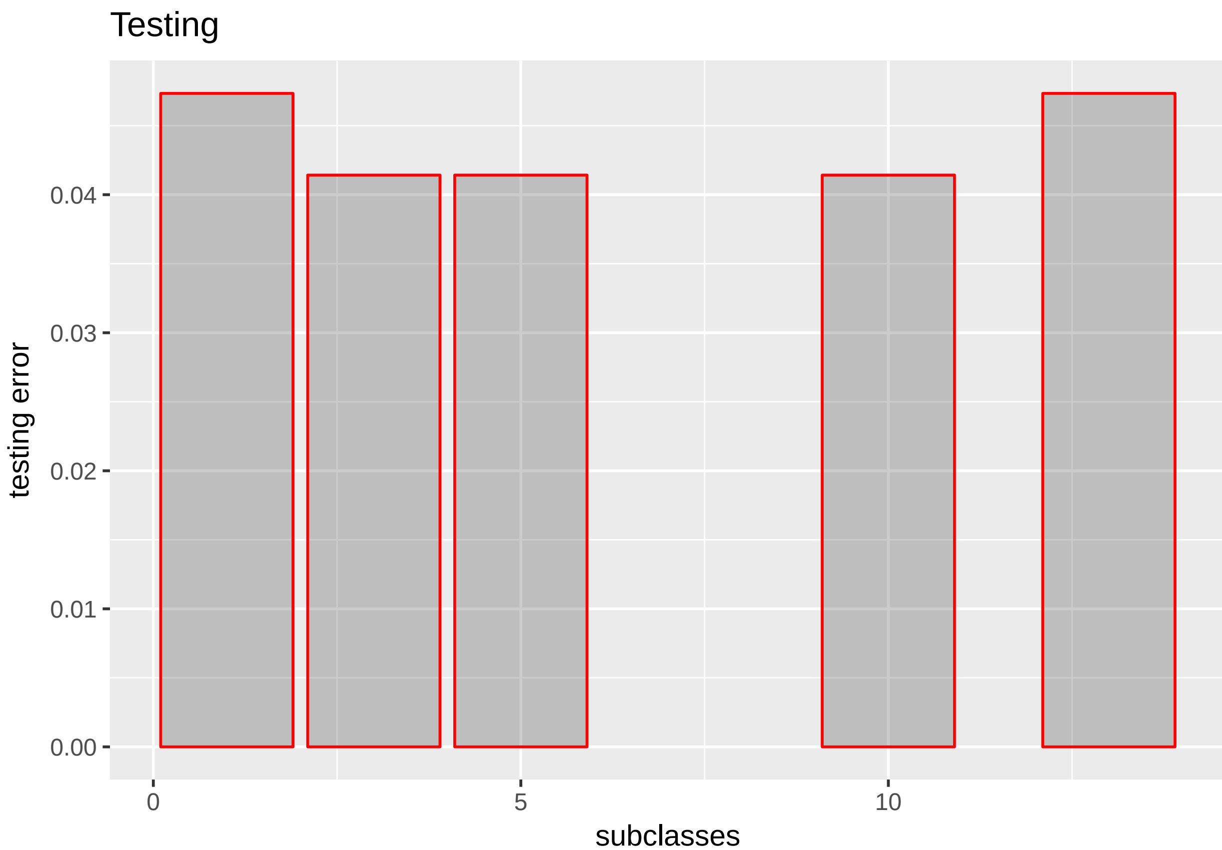
```
## [1] "The misclassfication error rate on the testing set for MDA is 0.04734"
```

```r
training <- c()
training <- c(mdatrainmis1, mdatrainmis2, mdatrainmis3, mdatrainmis5, mdatrainmis6)

testing <- c()
testing <- c(mdatestmis1, mdatestmis2, mdatestmis3, mdatestmis5, mdatestmis6)

mdarate <- data.frame(train = training, test = testing)

graph <- ggplot(as.data.frame(mdarate), aes(x = c(1,3,5,10,13), y = train))
graph <- graph + geom_bar(stat = "identity", alpha = 0.3, color = "red") +
labs(title = "Training", x = "subclasses", y = "training error")
```

```
graph
```




```
graph <- ggplot(as.data.frame(mdarate), aes(x = c(1,3,5,10,13), y = test))
graph <- graph + geom_bar(stat = "identity", alpha = 0.3, color = "red") +
labs(title = "Testing", x = "subclasses", y = "testing error")
graph
```



Comment:

I again set the seed with 1000.

As number of subclasses go up, the model becomes more flexible!!! (meaning more variance, but less bias)

I reported/printed out the misclassification error rate both on the training and testing data set above. Also, please be aware that misclassification error rate on testing set is what is important. Training and testing on the same data set is not a good practice to do…

I decided to make two more extra subclasses: (3,3) and (13,13) to see the pattern more clearly. As I could see from the misclassification rates, **the misclassification rate for training set generally goes down (as it randomly generates, sometimes it might not be monotonically decreasing) as sub classes increase because the model beocomes more flexible; however, misclassification rate for testing set goes down and bounce up.**

This pattern makes sense, as the training error generally goes down theoretically as the model becomes more flexiblel (because you train on the training set, and when you fit back onto the training set, as the model becomes more flexible, misclassfication error rate will definitely go down); however, testing error is not the case as being more flexible can increase variance. (so, as you make the model have less bias, the model will have more variance)

Here is a picture from the lecture note, below:

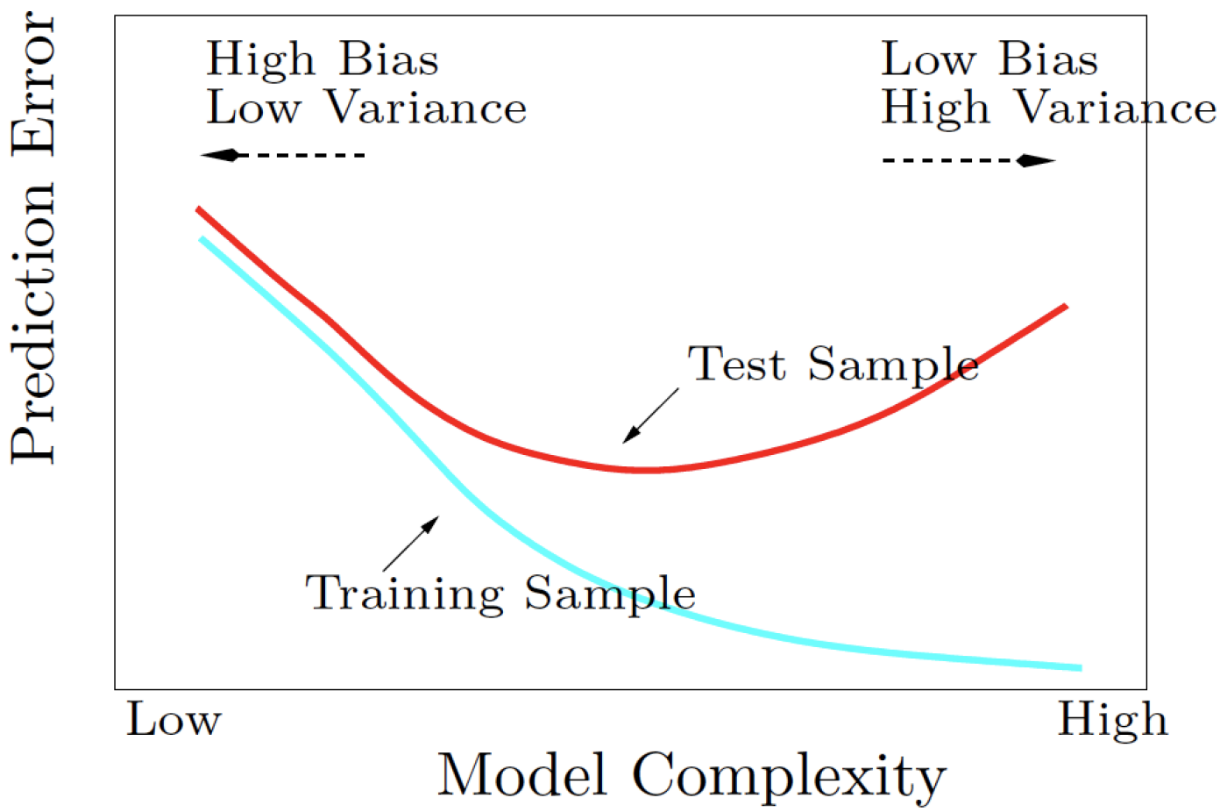


Figure 1: error curve

Part d) Fit Nearest Neighbor, with the number of neighbors k = 1, 5 and 10, respectively. Report the misclassification error rate on both the training and the testing data. Please describe the pattern, and see if it agrees with your expectation.

```r
set.seed(1000)
#k = 1
trainpredictk1 <- knn(train[,-1], train[,-1], train[,1], k = 1)
#trainpredictk1

testpredictk1 <- knn(train[,-1], test[,-1], train[,1], k = 1)
#testpredictk1

table(trainpredictk1, train[,1]) #confusion matrix of training
```

```
##
## trainpredictk1   0   1
##              0 251   0
##              1   0 149
```

```r
table(testpredictk1, test[,1]) #confusion matrix of testing
```

```
##
## testpredictk1  0  1
##             0 97  9
##             1  9 54
```

```r
knn1trainmis <- sum(trainpredictk1 != train[,1]) / length(train[,1])
knn1testmis <- sum(testpredictk1 != test[,1]) / length(test[,1])

paste("The misclassfication error rate on the training set for KNN is", round(knn1trainmis, 5))
```

```
## [1] "The misclassfication error rate on the training set for KNN is 0"
```

```r
paste("The misclassfication error rate on the testing set for KNN is", round(knn1testmis, 5))
```

```
## [1] "The misclassfication error rate on the testing set for KNN is 0.10651"
```

```r
#k = 3
trainpredictk2 <- knn(train[,-1], train[,-1], train[,1], k = 3)
#trainpredictk2

testpredictk2 <- knn(train[,-1], test[,-1], train[,1], k = 3)
#testpredictk2

table(trainpredictk2, train[,1]) #confusion matrix of training
```

```
##
## trainpredictk2   0   1
##              0 247  11
##              1   4 138
```

```r
table(testpredictk2, test[,1]) #confusion matrix of testing
```

```
##
## testpredictk2   0   1
##             0 102   6
##             1   4  57
```

```r
knn2trainmis <- sum(trainpredictk2 != train[,1]) / length(train[,1])
knn2testmis <- sum(testpredictk2 != test[,1]) / length(test[,1])

paste("The misclassfication error rate on the training set for KNN is", round(knn2trainmis, 5))
```

```
## [1] "The misclassfication error rate on the training set for KNN is 0.0375"
```

```r
paste("The misclassfication error rate on the testing set for KNN is", round(knn2testmis, 5))
```

```
## [1] "The misclassfication error rate on the testing set for KNN is 0.05917"
```

```r
#k = 5
trainpredictk3 <- knn(train[,-1], train[,-1], train[,1], k = 5)
#trainpredictk3

testpredictk3 <- knn(train[,-1], test[,-1], train[,1], k = 5)
#testpredictk3

table(trainpredictk3, train[,1]) #confusion matrix of training
```

```
##
## trainpredictk3   0   1
##              0 244  15
##              1   7 134
```

```r
table(testpredictk3, test[,1]) #confusion matrix of testing
```

```
##
## testpredictk3   0   1
##             0 100   6
##             1   6  57
```

```r
knn3trainmis <- sum(trainpredictk3 != train[,1]) / length(train[,1])
knn3testmis <- sum(testpredictk3 != test[,1]) / length(test[,1])

paste("The misclassfication error rate on the training set for KNN is", round(knn3trainmis, 5))
```

```
## [1] "The misclassfication error rate on the training set for KNN is 0.055"
```

```r
paste("The misclassfication error rate on the testing set for KNN is", round(knn3testmis, 5))
```

```
## [1] "The misclassfication error rate on the testing set for KNN is 0.07101"
```

```r
#k = 10
trainpredictk5 <- knn(train[,-1], train[,-1], train[,1], k = 10)
#trainpredictk5

testpredictk5 <- knn(train[,-1], test[,-1], train[,1], k = 10)
#testpredictk5

table(trainpredictk5, train[,1]) #confusion matrix of training
```

```
##
## trainpredictk5   0   1
##              0 244  15
##              1   7 134
```

```r
table(testpredictk5, test[,1]) #confusion matrix of testing
```

```
##
## testpredictk5   0   1
##             0 101   7
##             1   5  56
```

```r
knn5trainmis <- sum(trainpredictk5 != train[,1]) / length(train[,1])
knn5testmis <- sum(testpredictk5 != test[,1]) / length(test[,1])

paste("The misclassfication error rate on the training set for KNN is", round(knn5trainmis, 5))
```

```
## [1] "The misclassfication error rate on the training set for KNN is 0.055"
```

```r
paste("The misclassfication error rate on the testing set for KNN is", round(knn5testmis, 5))
```

```
## [1] "The misclassfication error rate on the testing set for KNN is 0.07101"
```

```r
#k = 13
trainpredictk6 <- knn(train[,-1], train[,-1], train[,1], k = 13)
#trainpredictk6

testpredictk6 <- knn(train[,-1], test[,-1], train[,1], k = 13)
#testpredictk6

table(trainpredictk6, train[,1]) #confusion matrix of training
```

```
##
## trainpredictk6   0   1
##              0 244  19
##              1   7 130
```

```r
table(testpredictk6, test[,1]) #confusion matrix of testing
```

```
##
## testpredictk6   0   1
##             0 102   8
##             1   4  55
```

```r
knn6trainmis <- sum(trainpredictk6 != train[,1]) / length(train[,1])
knn6testmis <- sum(testpredictk6 != test[,1]) / length(test[,1])

paste("The misclassfication error rate on the training set for KNN is", round(knn6trainmis, 5))
```

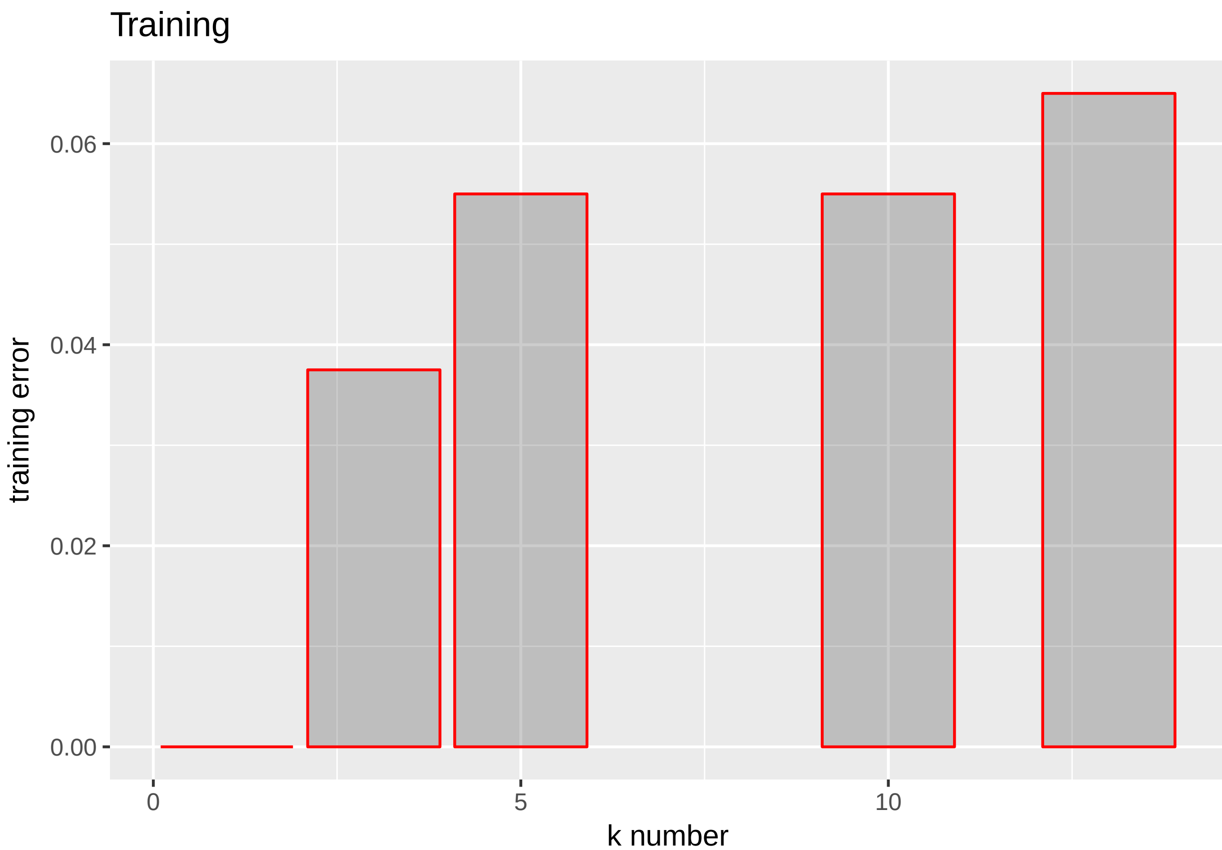
```
## [1] "The misclassfication error rate on the training set for KNN is 0.065"
```

```r
paste("The misclassfication error rate on the testing set for KNN is", round(knn6testmis, 5))
```

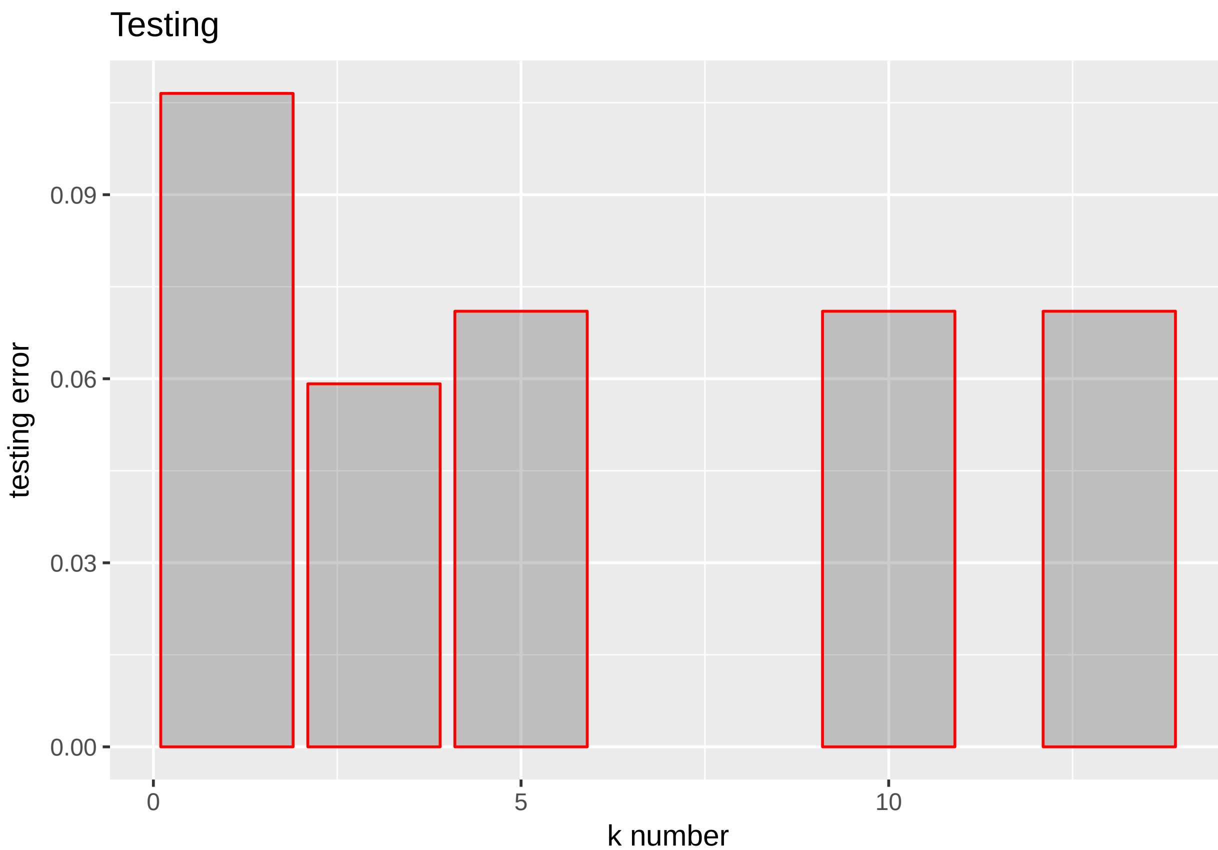
```
## [1] "The misclassfication error rate on the testing set for KNN is 0.07101"
```

```r
training1 <- c()
training1 <- c(knn1trainmis, knn2trainmis, knn3trainmis, knn5trainmis, knn6trainmis)

testing1 <- c()
testing1 <- c(knn1testmis, knn2testmis, knn3testmis, knn5testmis, knn6testmis)

mdarate1 <- data.frame(train = training1, test = testing1)

graph <- ggplot(as.data.frame(mdarate1), aes(x = c(1,3,5,10,13), y = train))
graph <- graph + geom_bar(stat = "identity", alpha = 0.3, color = "red") +
labs(title = "Training", x = "k number", y = "training error")
```

```
graph
```




```
graph <- ggplot(as.data.frame(mdarate1), aes(x = c(1,3,5,10,13), y = test))
graph <- graph + geom_bar(stat = "identity", alpha = 0.3, color = "red") +
labs(title = "Testing", x = "k number", y = "testing error")
graph
```



Comment:

I again set the seed with 1000.

I reported the misclassification error rate both on the training and testing data set above. Also, please be aware that misclassification error rate on testing set is what is important. Training and testing on the same data set is not a good practice to do…

I decided to make two more extra K's: 3 and 13 to see the pattern more clearly.

This is really similar as the previous question. **As we increase the number of K's, the model becomes less flexible (meaning more variance, but less bias); thus, the training error rate always goes up monotonically. However, this is not the case for testing set error. Although small k can make the model being more flexible (so small bias), it can increase the variance as overfitting might occur. So, the test set misclassification error rate can go up or down as k changes.**

Also, remeber that the training error/misclassification rate for k = 1 is always equal to 0, as they are matching with its own, so no misclassficiation can arise here.

It is always better to standardize the data in knn, as they can put more weights on the higher scale. However, since the instruction did not ask me to do it, I will just keep the way it is.

This pattern makes sense, as the training error monotonically goes down theoretically as the model becomes more flexiblel (because you train on the training set, and when you fit back onto the training set, as the model becomes more flexible, misclassfication error rate will definitely go down); however, testing error is not the case as being more flexible can increase variance. (so, as you make the model have less bias, the model will have more variance)

Here is a picture from the lecture note, below:

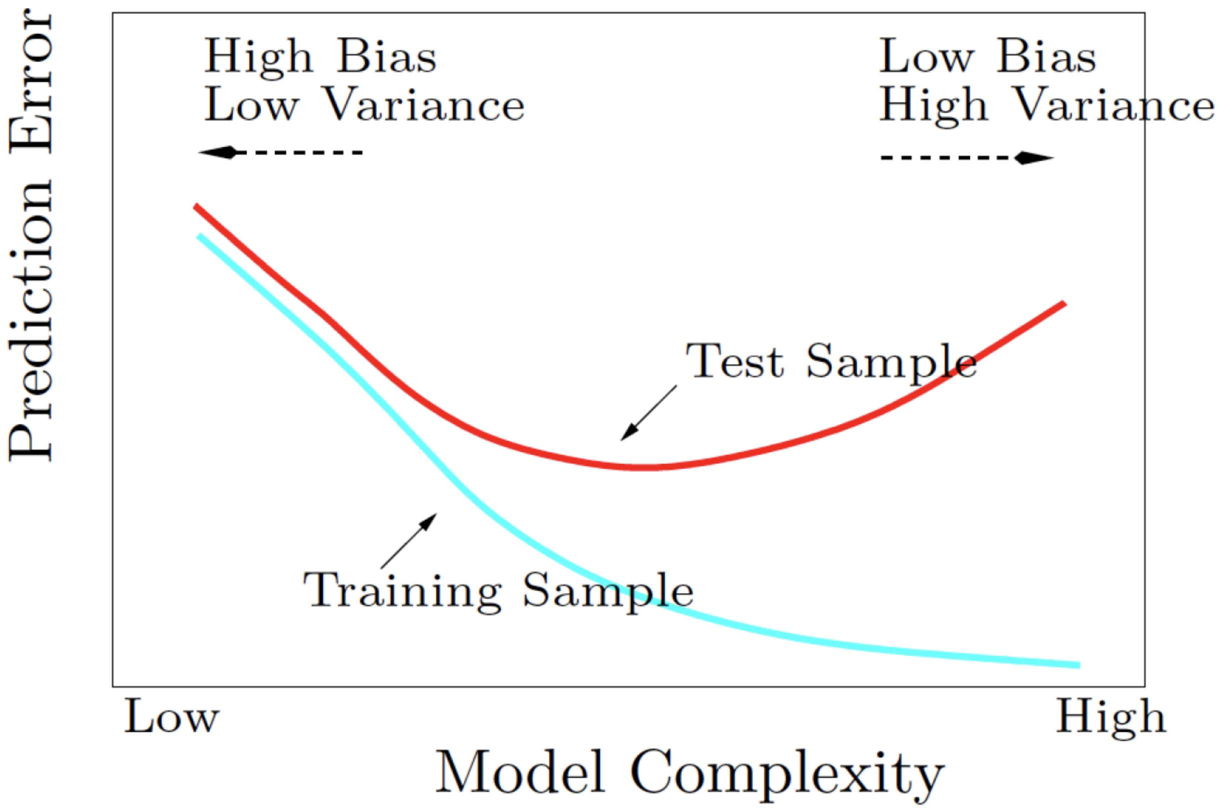


Figure 2: error curve

Problem 2 (Extra credit)

Good explanation, below:

https://www.youtube.com/watch?v=_aWzGGNrcic

Data import

```
univ <- read.table("Data-HW5-university.dat")
dim(univ)
```

```
## [1] 25  7
```

```
str(univ)
```

```
## 'data.frame':    25 obs. of  7 variables:
##  $ V1: Factor w/ 25 levels "Brown","CalTech",..: 9 15 25 17 11 7 2 6 1 10 ...
##  $ V2: num  14 13.8 13.8 13.6 13.8 ...
##  $ V3: int  91 91 95 90 94 90 100 89 89 75 ...
##  $ V4: int  14 14 19 20 30 30 25 23 22 44 ...
##  $ V5: int  11 8 11 12 10 12 6 10 13 7 ...
##  $ V6: num  39.5 30.2 43.5 36.5 34.9 ...
##  $ V7: int  97 95 96 93 91 95 81 95 94 87 ...
```

```
summary(univ)
```

```
##               V1           V2              V3               V4
##  Brown         : 1   Min.   :10.05   Min.   : 28.00   Min.   :14.0
##  CalTech       : 1   1st Qu.:12.40   1st Qu.: 74.00   1st Qu.:24.0
##  CarnegieMellon: 1   Median :12.85   Median : 81.00   Median :36.0
##  Columbia      : 1   Mean   :12.66   Mean   : 76.48   Mean   :39.2
##  Cornell       : 1   3rd Qu.:13.40   3rd Qu.: 90.00   3rd Qu.:50.0
##  Dartmouth     : 1   Max.   :14.15   Max.   :100.00   Max.   :90.0
##  (Other)       :19
##        V5               V6                V7
##  Min.   : 6.00   Min.   : 8.704   Min.   :67.00
##  1st Qu.:11.00   1st Qu.:15.140   1st Qu.:81.00
##  Median :12.00   Median :27.553   Median :90.00
##  Mean   :12.72   Mean   :27.388   Mean   :86.72
##  3rd Qu.:14.00   3rd Qu.:34.870   3rd Qu.:94.00
##  Max.   :25.00   Max.   :63.575   Max.   :97.00
##
```

```r
colnames(univ) <- c("University", "SAT", "top10", "acceptrate", "ratio", "expense", "graduate")

#NA check
na2 <- c()
for(i in 1:ncol(univ)){
  na2[i] <- sum(is.na(univ[,i]))
}
na2
```

```
## [1] 0 0 0 0 0 0 0
```

Comment:

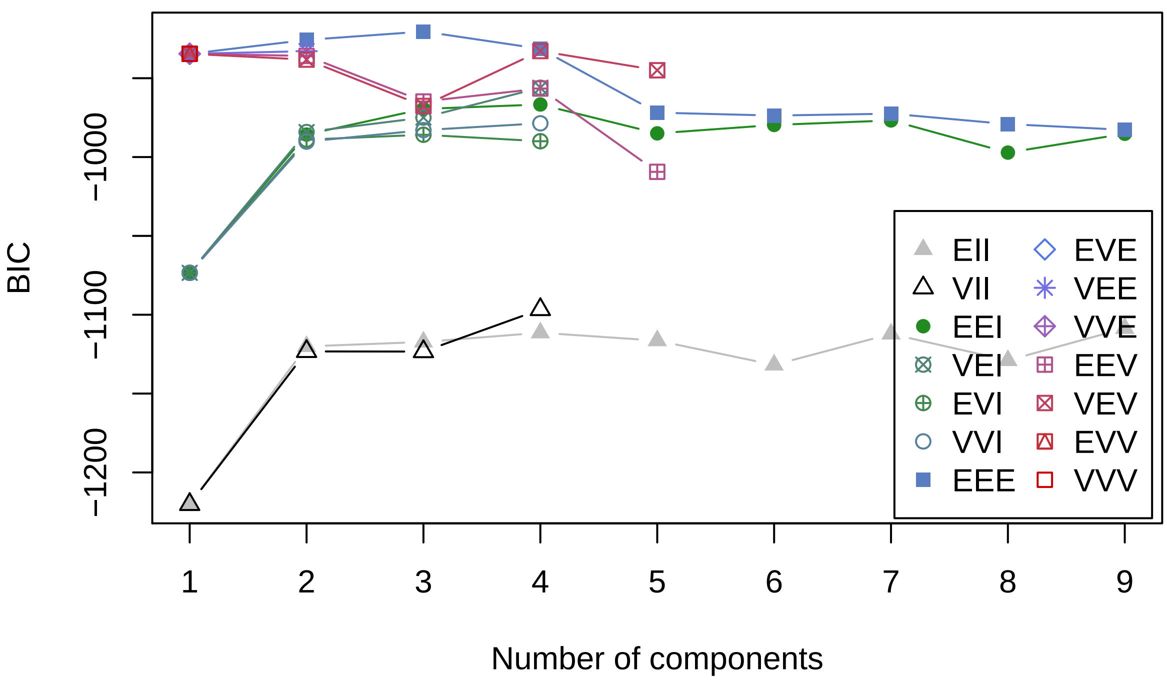
These variables include X1 = average SAT score of new freshmen, X2 = percentage of new freshmen in top 10% of high school class, X3 = percentage of applicants accepted, X4 = student-faculty ratio, X5 = estimated annual expenses, and X6 = graduation rate (%).

I included the for loop of checking NA existence in our original data, just in case.

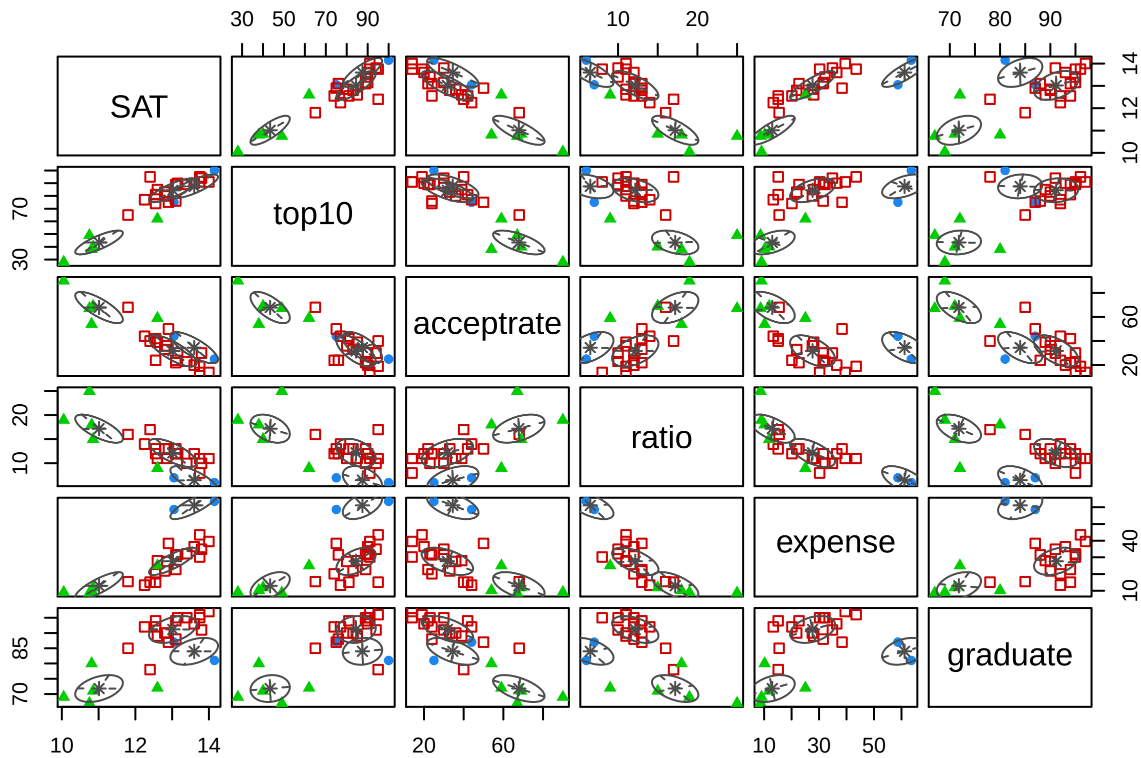
Part a) Use Mclust() in R to analyze this data. Report the best model using BIC.

Good link: https://www.statmethods.net/advstats/cluster.html & https://www.youtube.com/watch?v=5eDqRysaico

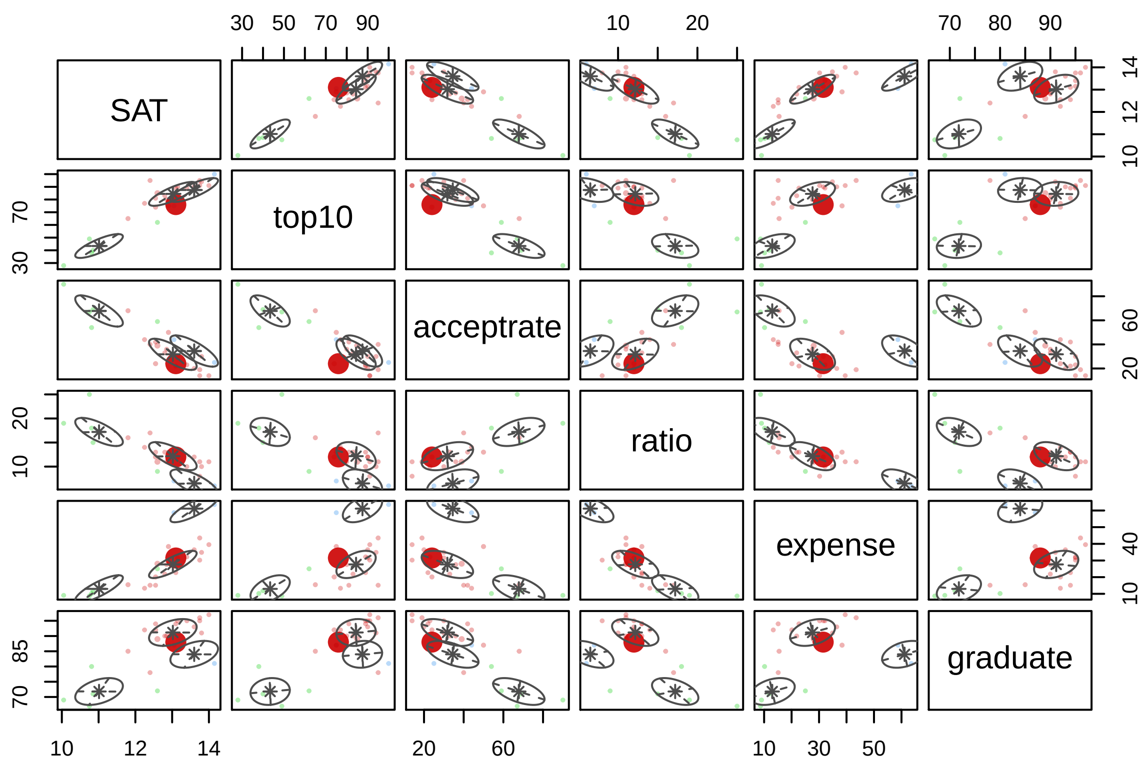
```r
fit <- Mclust(univ[,-1])
plot(mclustBIC(univ[,-1]))
```



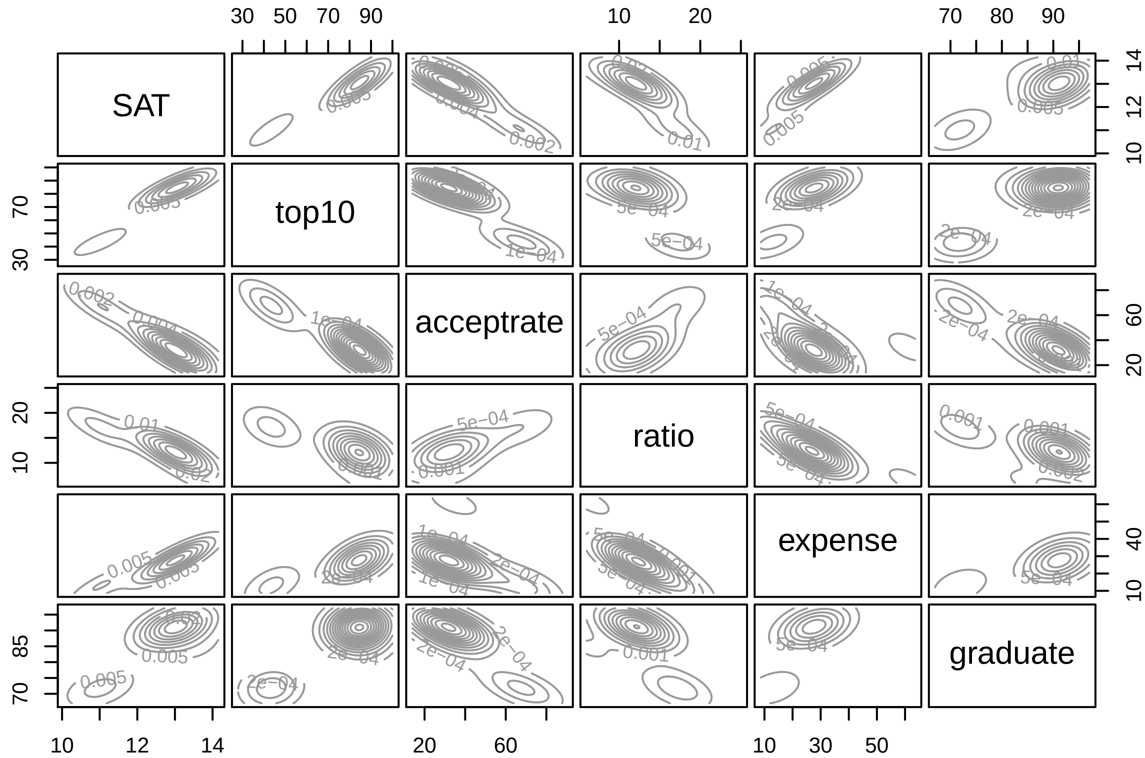

```r
plot(fit, what = "classification")
```




```r
plot(fit, what = "uncertainty")
```




```r
plot(fit, what = "density")
```




```r
summary(fit)
```

```
## ----------------------------------------------------
## Gaussian finite mixture model fitted by EM algorithm
## ----------------------------------------------------
##
## Mclust EEE (ellipsoidal, equal volume, shape and orientation) model with 3 components:
##
##  log.likelihood  n df       BIC       ICL
##        -394.209 25 41 -920.3919 -920.3921
##
## Clustering table:
##  1  2  3
##  2 18  5
```

```r
cat("This is how they are classified:", fit$classification)
```

```
## This is how they are classified: 2 2 2 2 2 2 1 2 2 1 2 2 2 2 2 2 2 2 3 2 2 3 3 3 3
```

```r
cat("The matrix of probability for each observation is:")
```

```
## The matrix of probability for each observation is:
```

```r
round(fit$z, 8)
```

```
##         [,1]      [,2]       [,3]
##  [1,] 0e+00 1.0000000 0.00000000
##  [2,] 0e+00 1.0000000 0.00000000
##  [3,] 0e+00 1.0000000 0.00000000
##  [4,] 0e+00 1.0000000 0.00000000
##  [5,] 0e+00 1.0000000 0.00000000
##  [6,] 0e+00 1.0000000 0.00000000
```

```
##  [7,] 1e+00 0.0000000 0.00000000
##  [8,] 0e+00 1.0000000 0.00000000
##  [9,] 0e+00 1.0000000 0.00000000
## [10,] 1e+00 0.0000000 0.00000000
## [11,] 1e-08 1.0000000 0.00000000
## [12,] 0e+00 1.0000000 0.00000000
## [13,] 0e+00 1.0000000 0.00000000
## [14,] 5e-07 0.9999995 0.00000000
## [15,] 0e+00 0.9998886 0.00011144
## [16,] 0e+00 1.0000000 0.00000000
## [17,] 0e+00 1.0000000 0.00000000
## [18,] 0e+00 1.0000000 0.00000002
## [19,] 0e+00 0.0000000 1.00000000
## [20,] 0e+00 1.0000000 0.00000001
## [21,] 0e+00 1.0000000 0.00000000
## [22,] 0e+00 0.0000000 1.00000000
## [23,] 0e+00 0.0000000 1.00000000
## [24,] 0e+00 0.0000000 1.00000000
## [25,] 0e+00 0.0000000 1.00000000
```

```
cat("The uncertainties for each observation is:", fit$uncertainty)
```

```
## The uncertainties for each observation is: 3.108624e-15 1.032507e-13 4.434197e-09 3.677059e-13 9.547
```

```
fit$BIC #same as mclustBIC(univ[,-1])
```

```
## Bayesian Information Criterion (BIC):
##         EII       VII        EEI        VEI        EVI        VVI
## 1 -1220.293 -1220.293 -1073.4288 -1073.4288 -1073.4288 -1073.4288
## 2 -1120.091 -1123.253  -985.7440  -984.1491  -988.9679  -990.2031
## 3 -1117.264 -1123.357  -969.4683  -974.9564  -985.8617  -982.9120
## 4 -1111.419 -1096.691  -966.6780  -956.0351  -989.9596  -978.6004
## 5 -1116.416        NA  -984.9752         NA         NA         NA
## 6 -1131.823        NA  -979.6441         NA         NA         NA
## 7 -1112.177        NA  -976.7745         NA         NA         NA
## 8 -1128.973        NA  -997.1818         NA         NA         NA
## 9 -1108.603        NA  -985.2352         NA         NA         NA
##         EEE      EVE      VEE      VVE        EEV       VEV      EVV
## 1 -934.5220 -934.522 -934.522 -934.522  -934.5220 -934.5220 -934.522
## 2 -925.7113       NA -932.856       NA  -935.8556 -938.2095       NA
## 3 -920.3919       NA       NA       NA  -964.7729 -967.5965       NA
## 4 -931.3249       NA       NA       NA  -956.5395 -932.7612       NA
## 5 -971.8253       NA       NA       NA -1009.3675 -944.9557       NA
## 6 -973.6945       NA       NA       NA         NA        NA       NA
## 7 -972.4915       NA       NA       NA         NA        NA       NA
## 8 -979.0933       NA       NA       NA         NA        NA       NA
## 9 -982.7571       NA       NA       NA         NA        NA       NA
##        VVV
## 1 -934.522
## 2       NA
## 3       NA
## 4       NA
## 5       NA
## 6       NA
## 7       NA
```

```
## 8       NA
## 9       NA
##
## Top 3 models based on the BIC criterion:
##     EEE,3     EEE,2     EEE,4
## -920.3919 -925.7113 -931.3249
```

```
paste("The optimal BIC value is", round(fit$bic, 5))
```

```
## [1] "The optimal BIC value is -920.39192"
```

```
paste("and the optimal number of mixture components (clusters) are", fit$G)
```

```
## [1] "and the optimal number of mixture components (clusters) are 3"
```

```
paste("and the model where the optimal BIC occurs (best covariance structure) is", fit$modelName)
```

```
## [1] "and the model where the optimal BIC occurs (best covariance structure) is EEE"
```

```
paste(", which is ellipsoidal, equal volume, shape, and orientation")
```

```
## [1] ", which is ellipsoidal, equal volume, shape, and orientation"
```

Comment:

Cluster is for unsupervised categorical data.

It is always better to standardize the data. However, since the instruction did not ask me to do it, I will just keep the way it is.

Please see the Best model reported using BIC.

Part b) Plot X2 (Top10) versus X5 (Expenses) with different clusters and university names marked out.

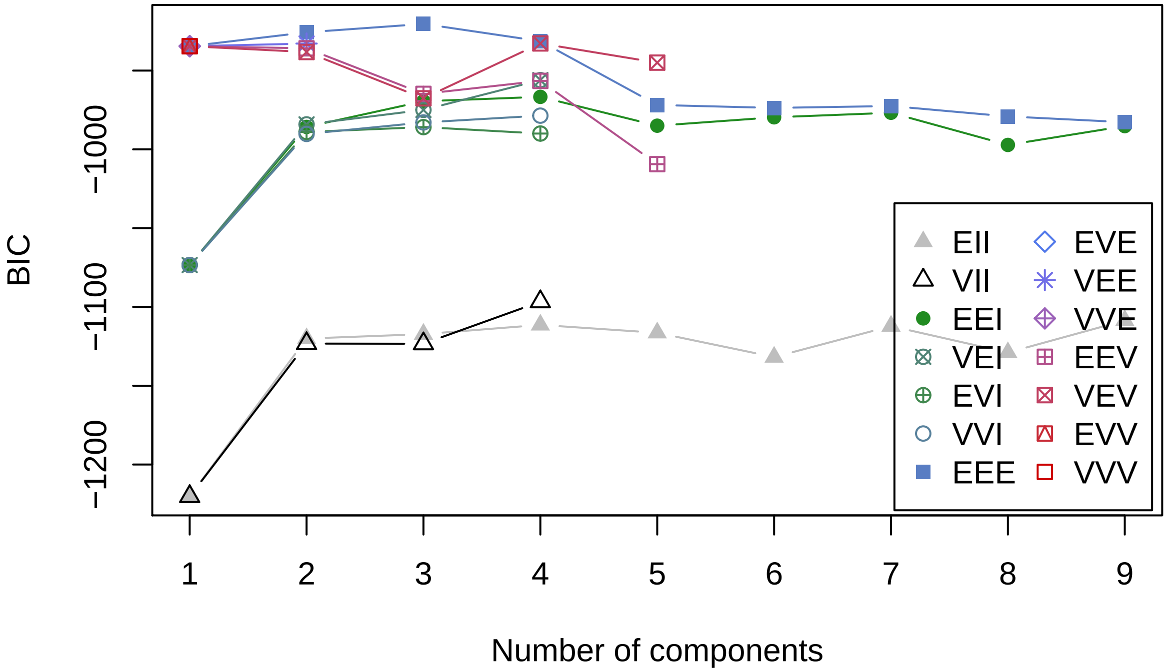
```
summary(fit, parameters = T, classification = T)
```

```
## ----------------------------------------------------
## Gaussian finite mixture model fitted by EM algorithm
## ----------------------------------------------------
##
## Mclust EEE (ellipsoidal, equal volume, shape and orientation) model with 3 components:
##
##  log.likelihood  n df       BIC       ICL
##        -394.209 25 41 -920.3919 -920.3921
##
## Clustering table:
##  1  2  3
##  2 18  5
##
## Mixing probabilities:
##          1          2          3
## 0.08000002 0.71999552 0.20000446
```

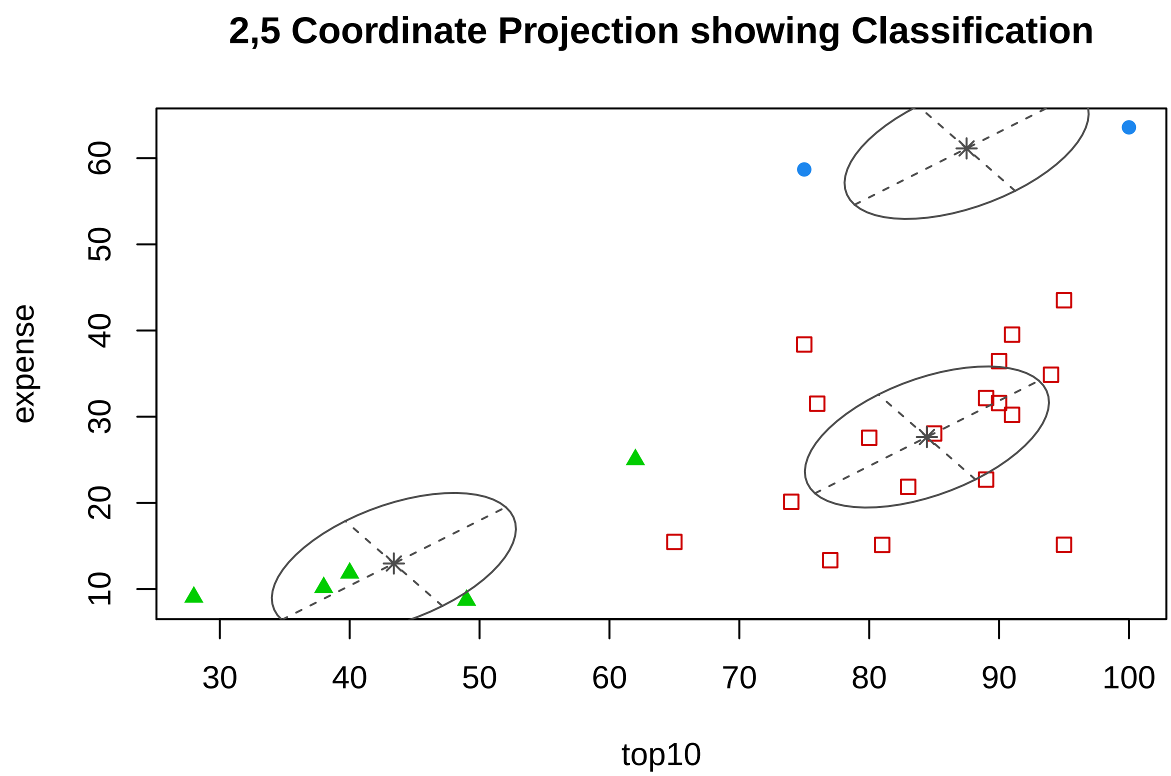
```
##
## Means:
##                  [,1]     [,2]     [,3]
## SAT        13.600000 13.01944 11.01205
## top10      87.499999 84.44450 43.40073
## acceptrate 34.500001 31.77783 67.79902
## ratio       6.500001 12.16667 17.19988
## expense    61.132992 27.64420 12.96801
## graduate   84.000001 91.16669 71.80036
##
## Variances:
## [,,1]
##                   SAT      top10 acceptrate      ratio    expense
## SAT         0.4203624   4.857124  -6.478414  -1.350462   4.332228
## top10       4.8571243  88.330197 -80.299547 -10.530095  37.935602
## acceptrate -6.4784138 -80.299547 168.584725  17.735677 -56.016789
## ratio      -1.3504618 -10.530095  17.735677   8.552121 -14.222641
## expense     4.3322285  37.935602 -56.016789 -14.222641  67.020795
## graduate    1.1814985   4.244901 -32.504605  -6.052377  12.862107
##              graduate
## SAT          1.181498
## top10        4.244901
## acceptrate -32.504605
## ratio       -6.052377
## expense     12.862107
## graduate    19.413126
## [,,2]
##                   SAT      top10 acceptrate      ratio    expense
## SAT         0.4203624   4.857124  -6.478414  -1.350462   4.332228
## top10       4.8571243  88.330197 -80.299547 -10.530095  37.935602
## acceptrate -6.4784138 -80.299547 168.584725  17.735677 -56.016789
## ratio      -1.3504618 -10.530095  17.735677   8.552121 -14.222641
## expense     4.3322285  37.935602 -56.016789 -14.222641  67.020795
## graduate    1.1814985   4.244901 -32.504605  -6.052377  12.862107
##              graduate
## SAT          1.181498
## top10        4.244901
## acceptrate -32.504605
## ratio       -6.052377
## expense     12.862107
## graduate    19.413126
## [,,3]
##                   SAT      top10 acceptrate      ratio    expense
## SAT         0.4203624   4.857124  -6.478414  -1.350462   4.332228
## top10       4.8571243  88.330197 -80.299547 -10.530095  37.935602
## acceptrate -6.4784138 -80.299547 168.584725  17.735677 -56.016789
## ratio      -1.3504618 -10.530095  17.735677   8.552121 -14.222641
## expense     4.3322285  37.935602 -56.016789 -14.222641  67.020795
## graduate    1.1814985   4.244901 -32.504605  -6.052377  12.862107
##              graduate
## SAT          1.181498
## top10        4.244901
## acceptrate -32.504605
## ratio       -6.052377
```

```
## expense      12.862107
## graduate     19.413126
##
## Classification:
##  [1] 2 2 2 2 2 2 1 2 2 1 2 2 2 2 2 2 2 2 3 2 2 3 3 3 3
```

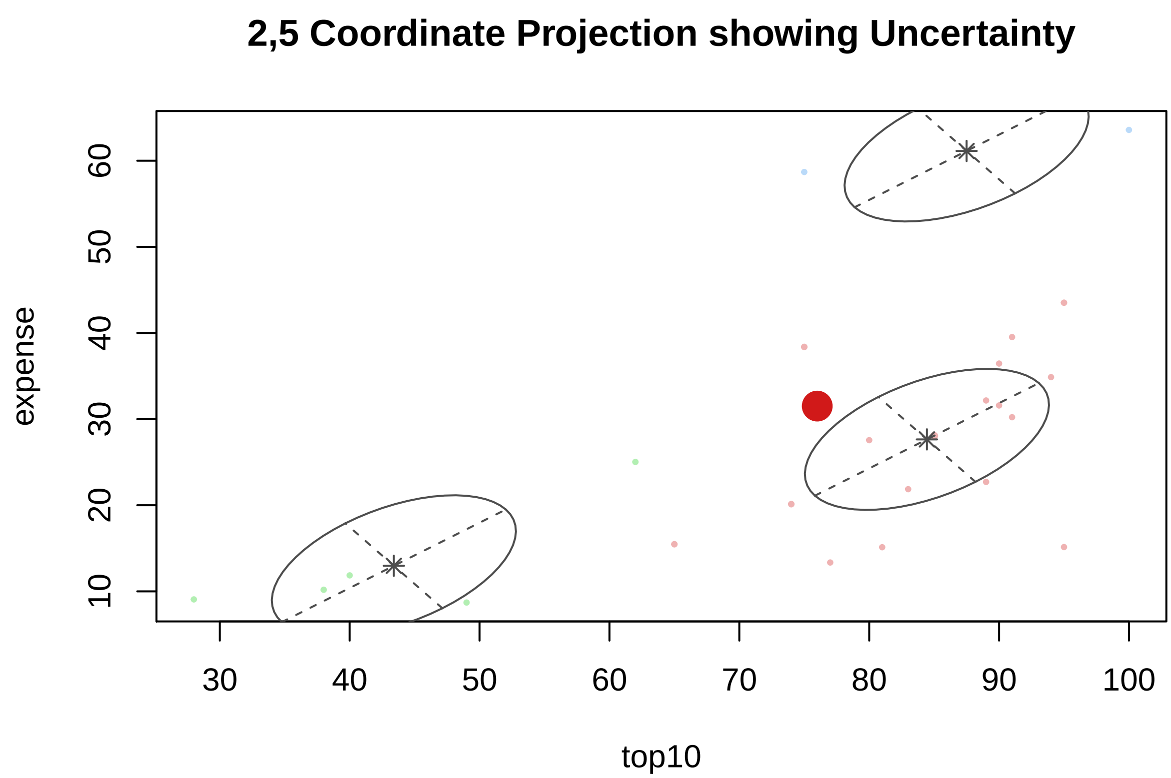
```r
plot(fit, what = "BIC", dimens = c(2, 5))
```



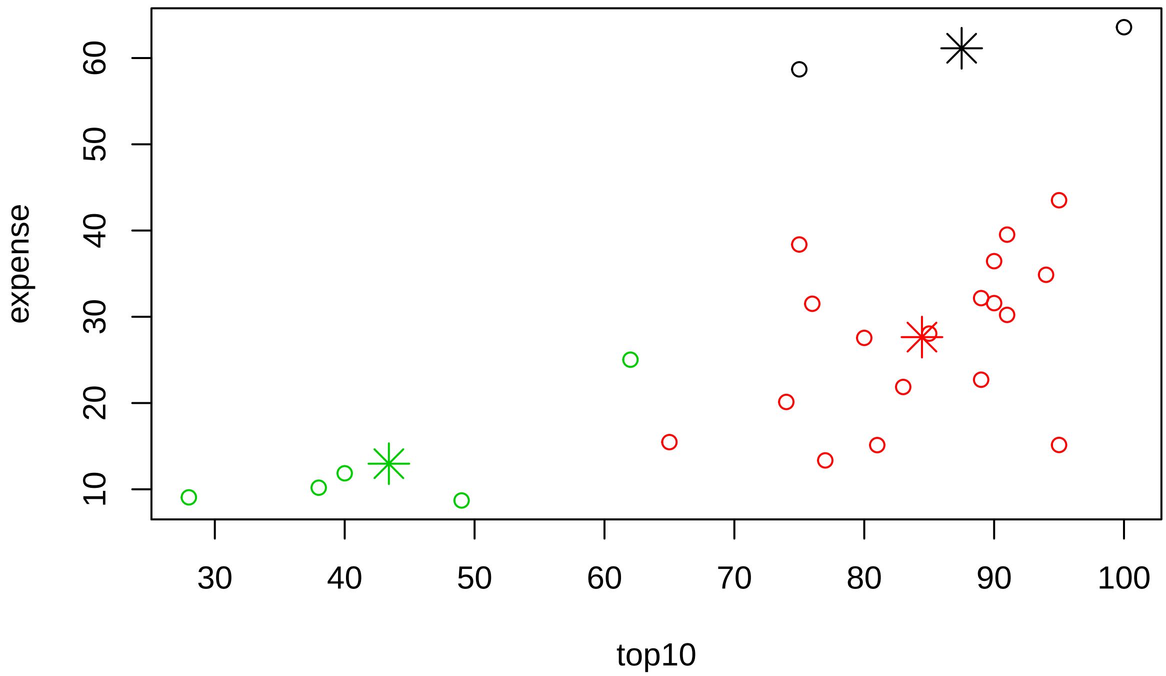

```r
plot(fit, what = "classification", dimens = c(2, 5))
```



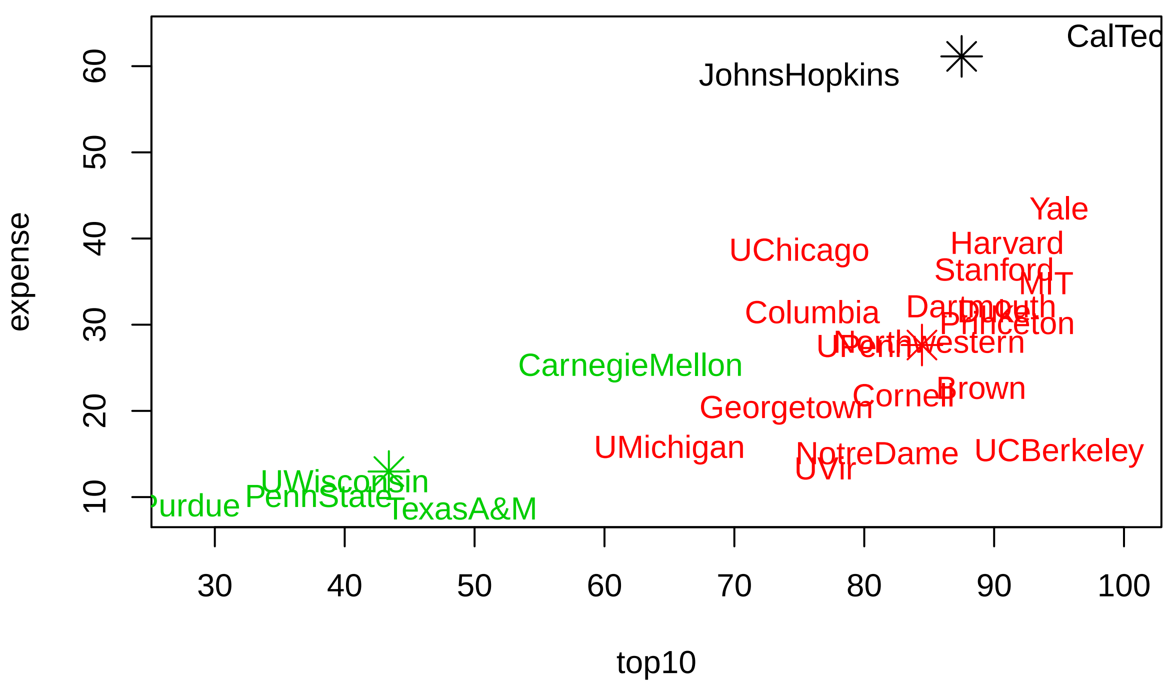

```r
plot(fit, what = "uncertainty", dimens = c(2, 5))
```




```r
plot(univ[,c(3,6)], col = fit$classification)
points(t(fit$parameters$mean[c(2,5),]), col = 1:3, pch = 8, cex = 2)
```




```
i <- 3
j <- 6
plot(univ[,i], univ[,j], xlab = colnames(univ)[i], ylab = colnames(univ)[j], type="n")

for(k in 1:nrow(univ)){
  text(univ[k,i], univ[k,j], univ[k,1], col = fit$classification[k])
}
points(t(fit$parameters$mean[c(2,5),]), col = 1:3, pch = 8, cex = 2)
```



Comment:

Basically, I indicated the different clusters with different colors, and I included all the universities names as the instruction says. Furthermore, I included the centroid for each cluster with the star signs. Last but not least, you could find whic dots are not certain from my uncertainty plot and check out my BIC plot as well for each model.

Part c) Apply k-means to this data with the number of clusters equal to the best number found in (a).

```
kmean <- kmeans(univ[,-1], fit$G)

kmean
```

```
## K-means clustering with 3 clusters of sizes 10, 6, 9
##
## Cluster means:
##        SAT    top10 acceptrate     ratio   expense graduate
## 1 12.71000 81.50000   35.40000 12.900000 23.38000 89.40000
## 2 11.14333 47.00000   67.83333 17.000000 13.38467 74.00000
## 3 13.62778 90.55556   24.33333  9.666667 41.17689 92.22222
##
## Clustering vector:
##  [1] 3 3 3 3 3 3 3 3 1 3 1 1 1 1 1 1 1 1 2 2 1 2 2 2 2
##
## Within cluster sum of squares by cluster:
## [1] 2017.334 2385.934 2506.920
##  (between_SS / total_SS =  73.2 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

```
summary(kmean)
```

```
##              Length Class  Mode
## cluster      25     -none- numeric
## centers      18     -none- numeric
## totss         1     -none- numeric
## withinss      3     -none- numeric
## tot.withinss  1     -none- numeric
## betweenss     1     -none- numeric
## size          3     -none- numeric
## iter          1     -none- numeric
## ifault        1     -none- numeric
```

Comment:

I used the kmeans function coming with stats package, with the optimal number of clusters I found from the model based cluster.

1. Randomly select K rows of the dataset and treat them as the initial cluster centroids $g_1, \ldots, g_K$.
2. For each observation x_i,
   a. compute $d(x_i, g_k) = ||x_i - g_k||_2^2$ for each k.
   b. Find $k^* = \arg\min_{k=1,\ldots,K} d(x_i, g_k)$.
   c. Assign x_i to cluster k^*.
3. For each cluster C_k, compute the cluster centroid $g_k = |C_k|^{-1} \sum_{i \in C_k} x_i$.
4. Repeat Step 2 and Step 3 until convergence (that is, the cluster assignment does not change).

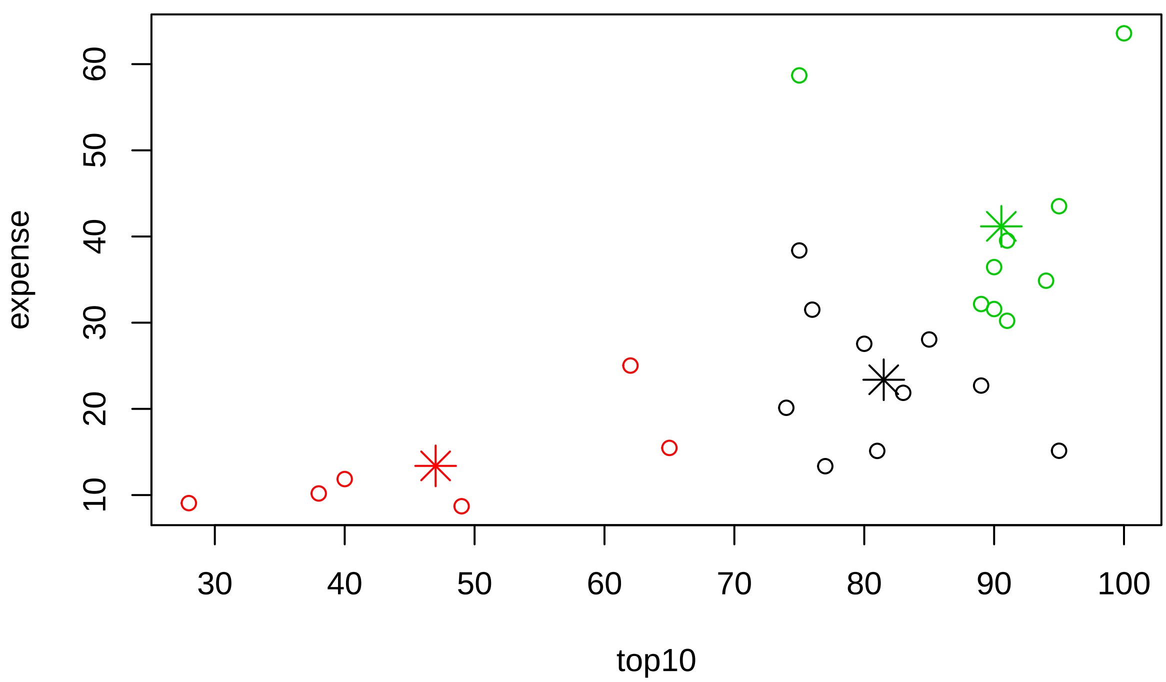
Figure 3: algorithm

Here is a good algorithm I found online, below:

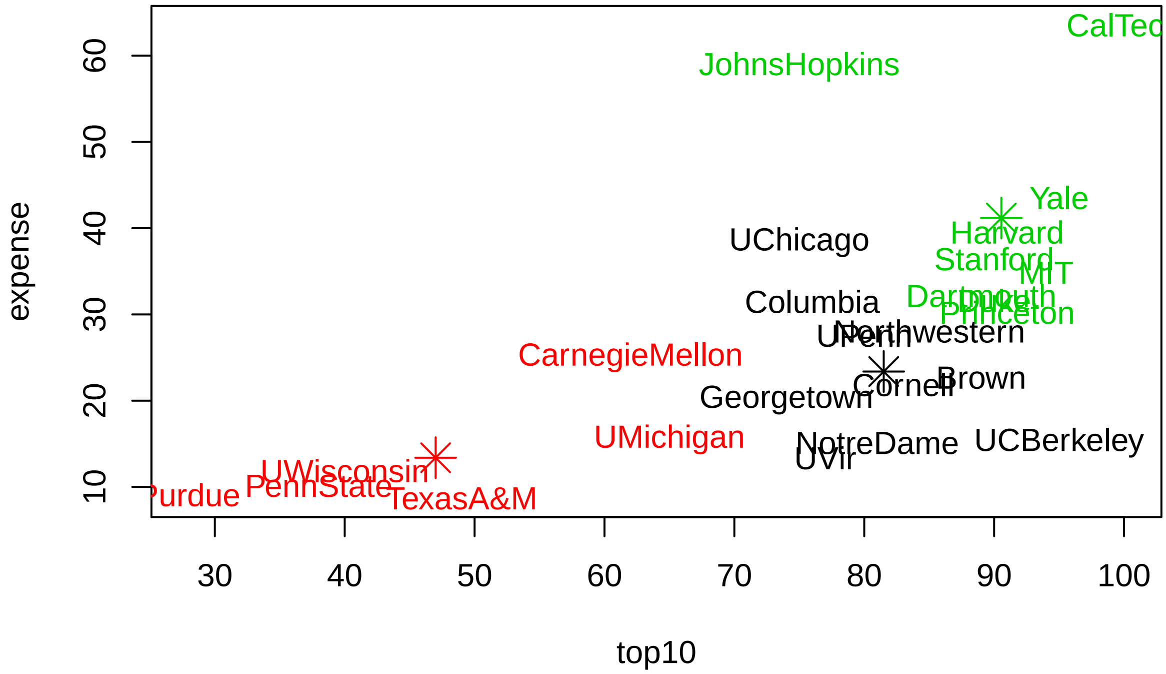
Part d) Repeat (b) with clusters found by kmeans now, and compare it with the results found in (b)

Link: https://stackoverflow.com/questions/28942195/k-means-plot-with-the-original-data-in-r & https://rstudio-pubs-static.s3.amazonaws.com/33876_1d7794d9a86647ca90c4f182df93f0e8.html

```
set.seed(1000)
plot(univ[,c(3,6)], col = kmean$cluster)
points((kmean$centers[,c(2,5)]), col = 1:3, pch = 8, cex = 2)
```



```r
i <- 3
j <- 6
plot(univ[,i], univ[,j], xlab = colnames(univ)[i], ylab = colnames(univ)[j], type="n")

for(k in 1:nrow(univ)){
  text(univ[k,i], univ[k,j], univ[k,1], col = kmean$cluster[k])
}
points((kmean$centers[,c(2,5)]), col = 1:3, pch = 8, cex = 2)
```



Comment:

Now, UMichigan belongs to the other group (the one in the left bottom).

Also, Yale, Harvard, Stanford, MIT, Dartmouth, Duke, and Princeton are belonged to the other group (the one on top right of the plot).

Seems like this plot makes more sense to me.

Part e) Apply hierarchical clustering to this data with average linkage. Report the clustering results using the number of clusters equal to the best number found in (a).

Link: http://www.sthda.com/english/articles/25-cluster-analysis-in-r-practical-guide/111-types-of-clustering-methods-overvie

https://www.youtube.com/watch?v=rg2cjfMsCk4

```r
hierach <- univ[,-1] %>% dist(method = "euclidean") %>% hclust(method = "average")
hierach
```

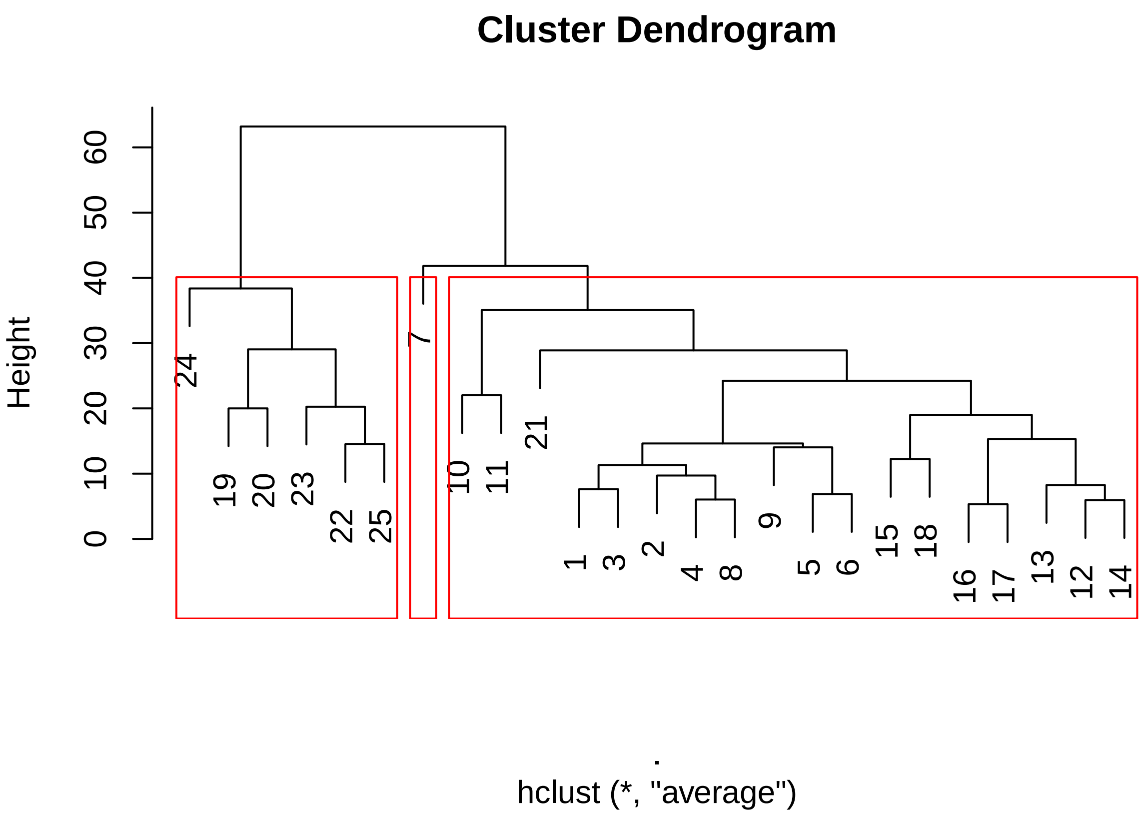
```
##
```

```
## Call:
## hclust(d = ., method = "average")
##
## Cluster method   : average
## Distance         : euclidean
## Number of objects: 25
```

```r
summary(hierach)
```

```
##             Length Class  Mode
## merge       48     -none- numeric
## height      24     -none- numeric
## order       25     -none- numeric
## labels       0     -none- NULL
## method       1     -none- character
## call         3     -none- call
## dist.method  1     -none- character
```

```r
plot(hierach)
rect.hclust(hierach, k = fit$G, border="red")
```



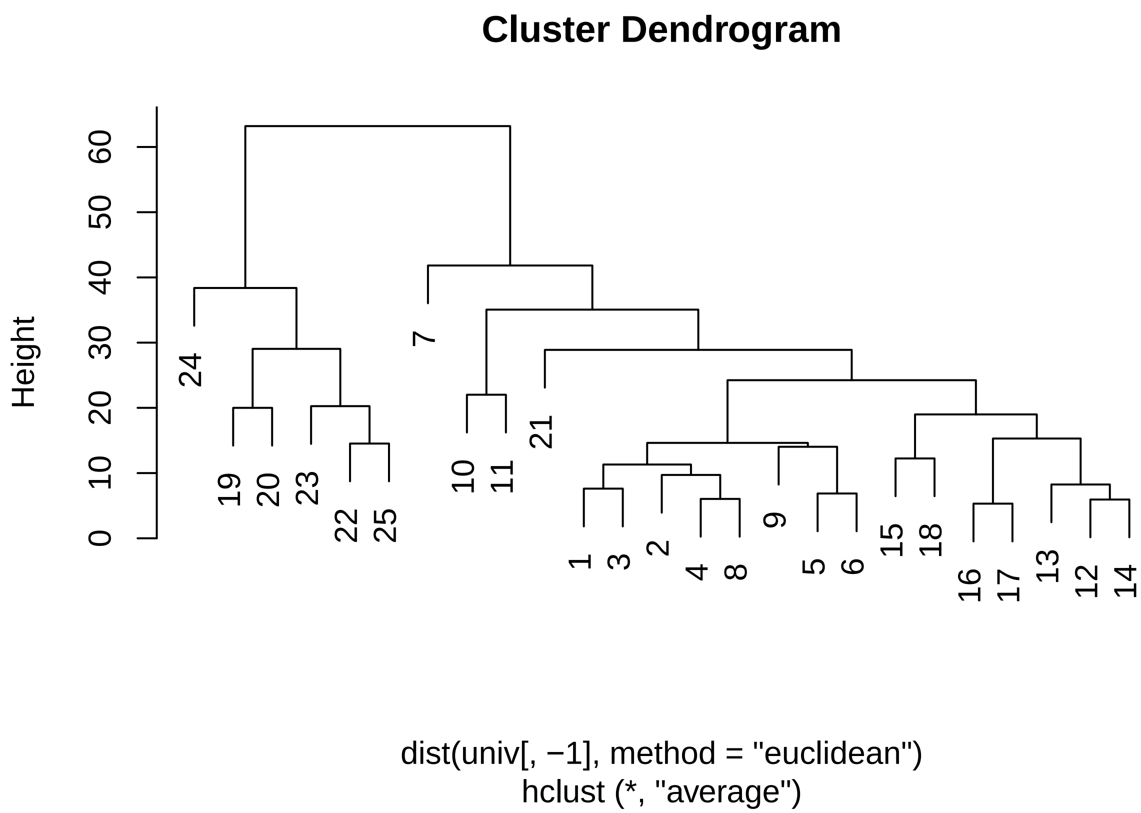

```r
grouping <- cutree(hierach, k = fit$G)
print("Here is a clustering results:")
```

```
## [1] "Here is a clustering results:"
```

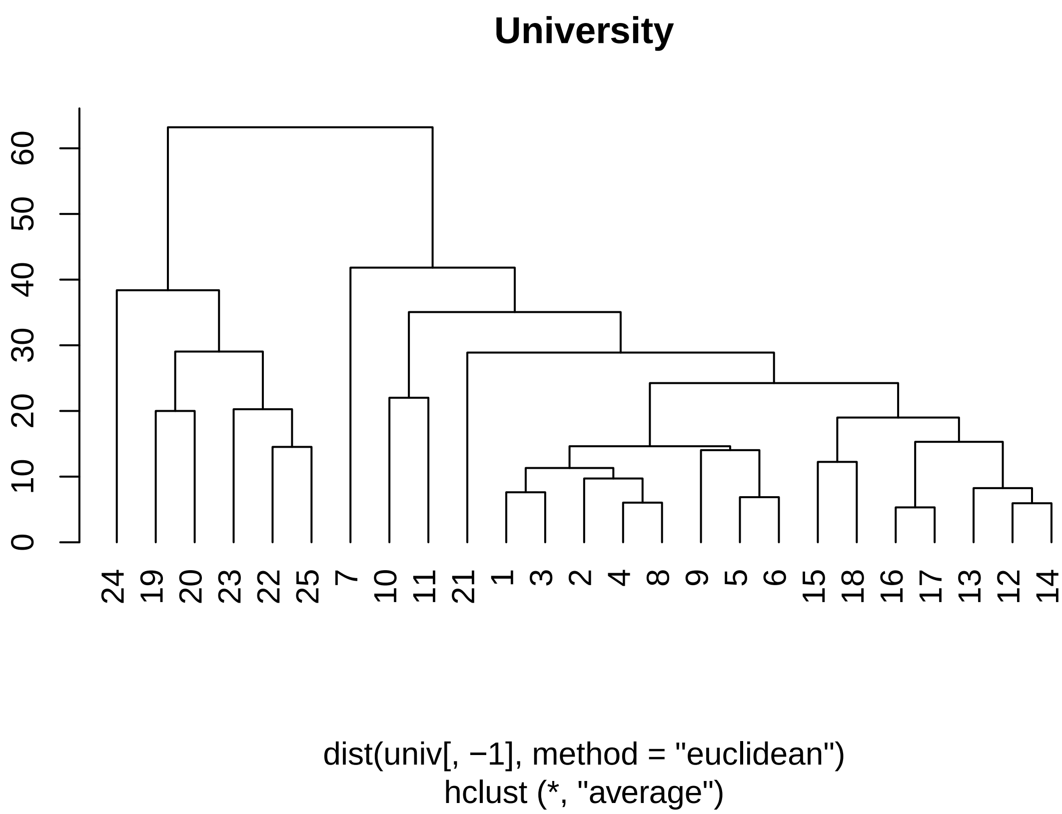
```r
print(grouping)
```

```
##  [1] 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 3 3 1 3 3 3 3
```

```r
# If you scaled...
# fviz_dend(hierach, k = fit$G,
#           cex = 0.5, # label size
#           k_colors = c("#2E9FDF", "#00AFBB", "#E7B800", "#FC4E07"),
#           color_labels_by_k = TRUE, # color labels by groups
#           rect = TRUE # Add rectangle around groups
#           )

hier <- hclust(dist(univ[,-1], method = "euclidean"), method = "average")
plot(hier)
```




```r
plot(hier, hang = -1, main = "University", ylab = NULL)
```



Comment:

Please refer to the diagram above. (I included a red box so the reader can get the clusters easily)

Also, please refer to the groups I printed out using cutree function.

One of the problems of K-means algorithm is that they need to define K before the algorithm runs, but hierarchial clustering does not need to define k beforehand. In hierarchial clustering, we need to define n-clusters (where n is the number of obervations), and then, generally merge from bottom to up, until there is only one cluster left. So, we are basically repeatedly combining the two clusters with the shortest distance each other. And, there are different types of cluster dissimilarity measures (linkage).