stats101c_hw6

Takao

2022-11-29

Homework 6

Takao Oba

Q1) You may have seen the "betterbirths2000" data in Stats 10. It consists of a random sample of 2000 births in North Carolina that are collected in order to track health issues in new born babies. These data are saved in a file better2000births.csv in the Week 9 section of the CCLE.

```
births <- read.csv("/Users/takaooba/Downloads/better2000births (1).csv")

births <- na.omit(births)
births$Gender <- as.factor(births$Gender)
births$Premie <- as.factor(births$Premie)
births$Marital <- as.factor(births$Marital)
births$Racemom <- as.factor(births$Racedad)
births$Racedad <- as.factor(births$Racedad)
births$Hispmom <- as.factor(births$Hispmom)
births$Hispdad <- as.factor(births$Hispdad)
births$MomPriorCond <- as.factor(births$MomPriorCond)
births$BirthDef <- as.factor(births$BirthDef)
births$BirthComp <- as.factor(births$BirthComp)
births$DelivComp <- as.factor(births$DelivComp)
```

a) Split your data into Training and Testing. You should have 1000 observations in your training data after omitting the missing values in your data. Use the set.seed "1128" to do the split. Use a tree (not pruned) to predict whether a baby will be born prematurely or normal. What is the testing misclassification error?

```
dim(births)
```

```
## [1] 1998   21
```

```
set.seed(1128)
test.i <- sample(1:nrow(births), 1000, replace = F)

births.test <- births[-test.i,]
births.train <- births[test.i,]
```

Using a tree

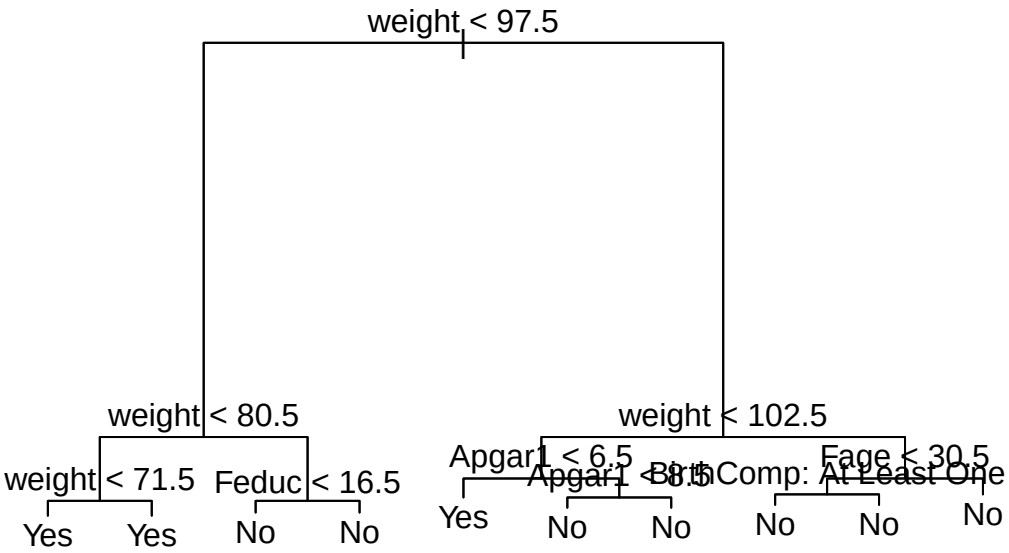
```
# install.packages("tree")
library(tree)
```

```
## Warning: package 'tree' was built under R version 4.1.2
```

```
births.model <- tree(formula = factor(Premie) ~ ., data = births.train)
```

```
## Warning in tree(formula = factor(Premie) ~ ., data = births.train): NAs
## introduced by coercion
```

```
plot(births.model)
text(births.model, pretty =0)
```



```r
births.y <- births.test$Premie

preds <- predict(births.model, newdata = births.test, type = "class")
```

```
## Warning in pred1.tree(object, tree.matrix(newdata)): NAs introduced by coercion
```

```r
conf.matrx <- table(preds, factor(births.test$Premie))

conf.matrx
```

```
##
## preds  No Yes
##   No  895  49
##   Yes  13  41
```

```r
(conf.matrx[2,1] + conf.matrx[1,2])/sum(conf.matrx)
```

```
## [1] 0.06212425
```

The misclassification rate is 0.0621245

b) Use cross-validation to determine if the tree can be improved through pruning. If so, prune the tree to the appropriate size and provide a plot.

```r
cv.train <- cv.tree(births.model, FUN = prune.misclass)
```

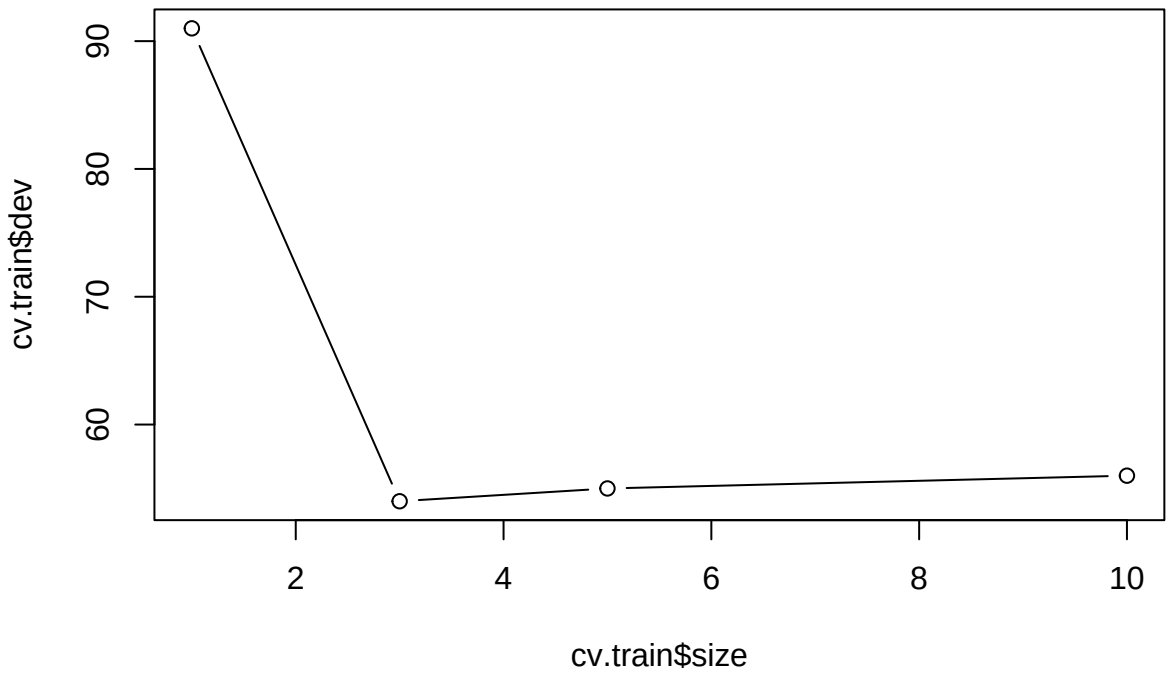
```
## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion
```

```
## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion
```

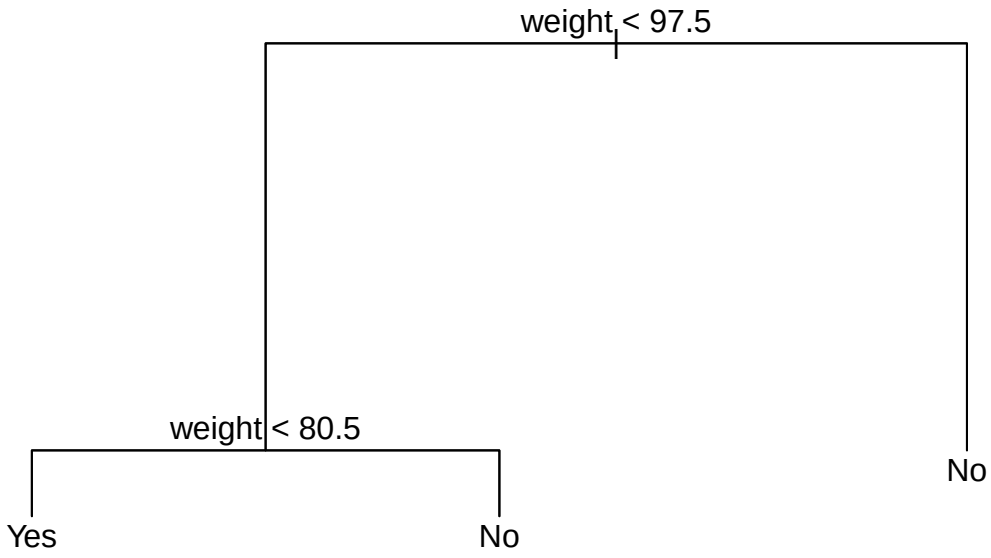
```
names(cv.train)
```

```
## [1] "size"   "dev"    "k"      "method"
```

```
# Find best value using the plot function
plot(cv.train$dev ~ cv.train$size, type = "b")
```




```
pruned.fit <- prune.misclass(births.model, best = 3)
plot(pruned.fit)
text(pruned.fit, pretty = TRUE)
```



Looking at the generated graph above, we can see that the appropriate size is 3. The lowest y value is when size = 3.

```
summary(pruned.fit)
```

```
##
## Classification tree:
## snip.tree(tree = births.model, nodes = c(4L, 5L, 3L))
## Variables actually used in tree construction:
## [1] "weight"
## Number of terminal nodes:  3
## Residual mean deviance:  0.322 = 321.1 / 997
## Misclassification error rate: 0.054 = 54 / 1000
```

```
# births.train
```

We will see if pruning makes the misclassification rate better. Recall that without pruning, we had a misclassification rate of 0.06212425

Now, we see that the misclassification rate is 0.056 which is lower than the misclassification rate without pruning.

c) Interpret your pruned tree (or your tree in (a) if you did not need to prune). In particular, does it tell us whether smoking is a potential cause of premature births? What factors are associated with premature births?

Basing on the pruned tree, we have that the factors that are associated with premature births is Weights. The pruned plot have performed fairly well with a low misclassification rate. Intuitively, smoking should be a potential cause of premature births, but in this decision tree, we were not able to conclude that smoking is directly associated with premature births. However, according to CDC.gov (https://www.cdc.gov/tobacco/campaign/tips/diseases/pregnancy.html#:~:text=Smoking%20slows%20your%20baby's%20growth,babies%20often%20have% we have that smoking slows the baby's growth before birth, or more specifically, the weight of the baby. The factor that is most associated with premature births is Weights.

d) What is the testing misclassification error rate of your pruned tree? Keep in mind that approximately 9% of all births are premature. This means that if a doctor simply predict "not premature" ALWAYS, he or she will have only a 9% misclassification error. Did you do better based on your tree models?

```
preds <- predict(pruned.fit, newdata = births.test, type = "class")
```

```
## Warning in pred1.tree(object, tree.matrix(newdata)): NAs introduced by coercion
```

```
conf.matrx <- table(preds, factor(births.test$Premie))

conf.matrx
```

```
##
## preds  No Yes
##   No  901  50
##   Yes   7  40
```

```
(conf.matrx[2,1] + conf.matrx[1,2])/sum(conf.matrx)
```

```
## [1] 0.05711423
```

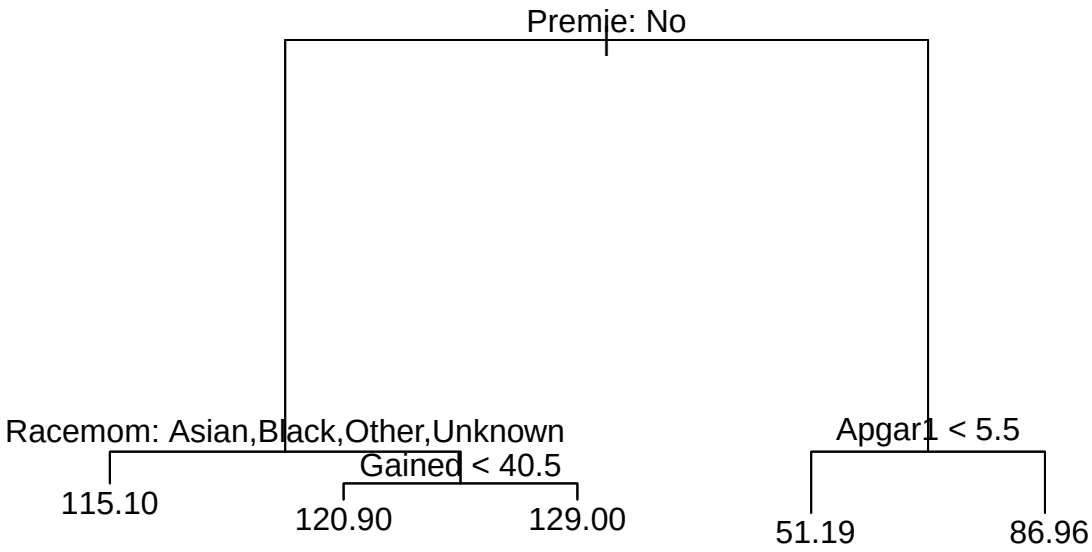
The misclassification rate for testing is 0.05711423. Since the misclassification rate is 5.71%, which is less than the 9% misclassification error, we have that this tree models is better than simply predicting "not premature" for all the babies.

Q2) a) Use the same data (training and testing parts from Q1). Use tree (not pruned) to predict the weight of the baby (a numerical variable) using all the other variables in the data aspredictors. What is your MSE?

```
prd.weight <- tree(weight ~ ., data = births.train)
```

```
## Warning in tree(weight ~ ., data = births.train): NAs introduced by coercion
```

```
plot(prd.weight)
text(prd.weight, pretty = 0)
```




```
yhat.train <- predict(prd.weight, newdata = births.test, type = "vector")
```

```
## Warning in pred1.tree(object, tree.matrix(newdata)): NAs introduced by coercion
```

```
head(yhat.train)
```

```
##        5        6       10       14       16       17
## 115.0982 120.8698 120.8698 115.0982 115.0982 115.0982
```

```
# the MSE is determined through direct calculation

mean((births.test$weight - yhat.train)^2)
```

```
## [1] 276.8005
```

b) Use cross-validation to determine if the tree can be improved through pruning. If so, prune the tree to the appropriate size and provide a plot.

```r
cv.train = cv.tree(prd.weight, FUN = prune.tree)
```

```
## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion
```

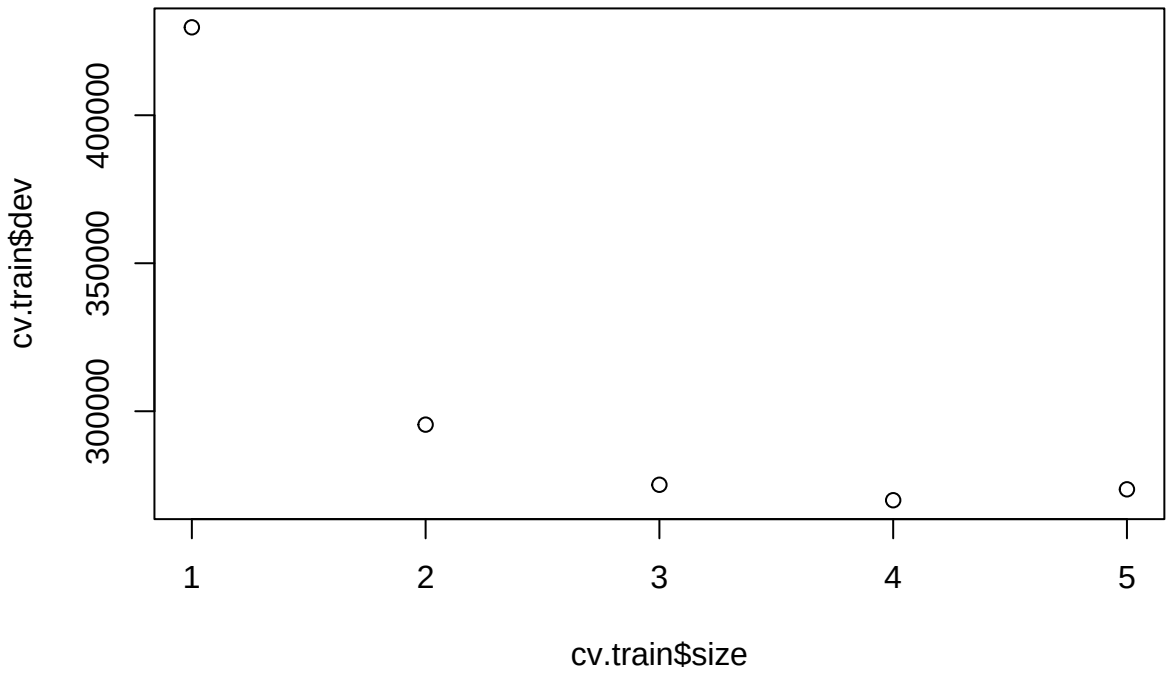
```
names(cv.train)
```

```
## [1] "size"   "dev"    "k"      "method"
```

```
summary(cv.train)
```

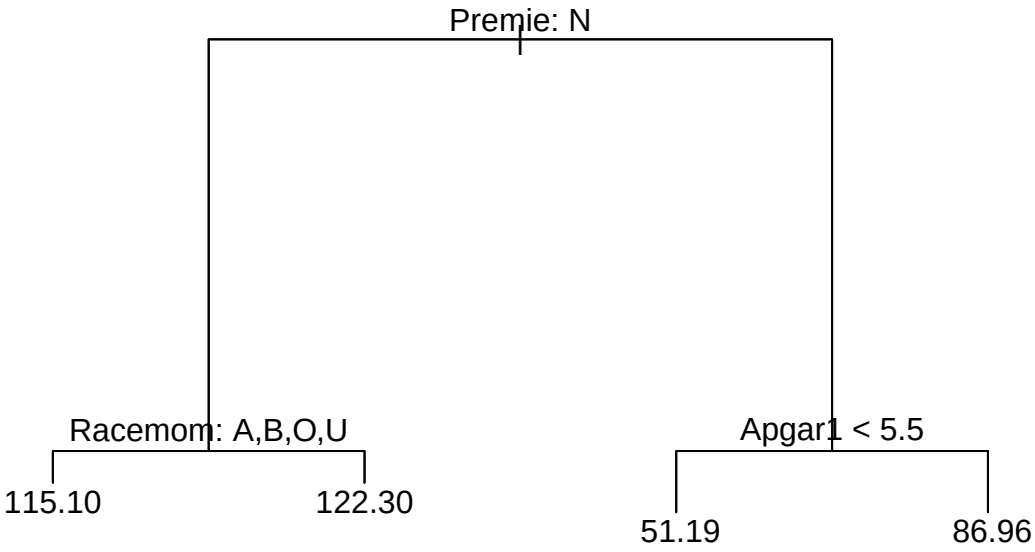
```
##        Length Class  Mode
## size   5      -none- numeric
## dev    5      -none- numeric
## k      5      -none- numeric
## method 1      -none- character
```

```
plot(cv.train$size, cv.train$dev)
```



From the above table, we have that the best size we can choose is 4.

```
# make pruned tree

train.pruned <- prune.tree(prd.weight, best = 4)
plot(train.pruned)
text(train.pruned, pretty = TRUE)
```




```
summary(train.pruned)
```

```
##
## Regression tree:
## snip.tree(tree = prd.weight, nodes = 5L)
## Variables actually used in tree construction:
## [1] "Premie"  "Racemom" "Apgar1"
## Number of terminal nodes:  4
## Residual mean deviance:  260.1 = 259000 / 996
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -58.9600 -10.2600  -0.2556   0.0000   9.7840  69.0400
```

c) Interpret your pruned tree (or your tree in (a) if you did not need to prune). In particular, does it tell us whether the number of visits predictor is an important feature of baby weights at birth? What other predictor you think are import predictors are associated with weight based on your pruned tree model?

Referring back to the constructed tree above, we have that the number of visits predictor is not an important feature of baby weights at birth. This is because the number of visitors are not one of the components in the decision tree. Further, the predictors that are used in the trees are Premie, Apgar1, and Racemom. These three predictors are associated with weight based on my pruned tree model.

d) Report the MSE of the updated pruned tree model.

```
# Directly calculating the MSE

prd.test <- predict(train.pruned, newdata = births.test)
```

```
## Warning in pred1.tree(object, tree.matrix(newdata)): NAs introduced by coercion
```

```
tempweight <- births.test$weight

mse <- mean((tempweight - prd.test)^2)
mse
```

```
## [1] 277.2483
```

Q3) Download the icu data from ccle week 9: Split the data into 70% training and 30% testing using set.seed(1128).

```
icu <- read.csv("/Users/takaooba/Downloads/icu data.csv", stringsAsFactors = T)

dim(icu)
```

```
## [1] 200  20
```

```
head(icu)
```

```
##       STA race.n   age.c  sys.c   hra.c    SEX                SER    CAN
## 1 survive  white -30.545   9.72 -10.925 Female Medical, Surgical1 cancer
## 2 survive  white   1.455 -20.28 -18.925   Male Medical, Surgical1 cancer
## 3 survive  white  19.455 -32.28 -28.925   Male Medical, Surgical2 cancer
## 4 survive  white  -3.545   9.72   4.075   Male Medical, Surgical1 cancer
## 5 survive  white  29.455 -22.28  55.075 Female Medical, Surgical2 cancer
## 6 survive  white  11.455 -22.28  33.075   Male Medical, Surgical1 cancer
##             CRN          INF CPR         PRE       TYP         FRA  PO2   PH
## 1 renal failure    infection  no no previous emergency no fracture < 60 7.25
## 2 renal failure no infection  no    previous emergency no fracture < 60 7.25
## 3 renal failure no infection  no no previous  elective no fracture < 60 7.25
## 4 renal failure    infection  no no previous emergency    fracture < 60 7.25
## 5 renal failure    infection  no    previous emergency no fracture < 60 7.25
## 6 renal failure    infection  no no previous emergency no fracture > 60 7.25
##   PCO BIC CRE     LOC
## 1  45  18   2 no coma
## 2  45  18   2 no coma
## 3  45  18   2 no coma
## 4  45  18   2 no coma
## 5  45  18   2 no coma
## 6  45 <18   2 no coma
```

```r
# Splitting the data into testing and training
set.seed(1128)
sam <- sample(1:200, 140, replace = F)
icu.train <- icu[sam,]
icu.test <- icu[-sam,]
```

a) Apply Bagging classification technique to predict the surviving status of an ICU patient. Note we have 19 predictors, so we use randomForest library to do so. Report the model summary (confusion matrices of both the training and the testing data, report the corresponding misclassification rates)

```r
set.seed(1128)
dim(icu.train)
```

```
## [1] 140  20
```

```r
# The amount of predictors is 19
# install.packages("randomForest")
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.1.2
```

```
## randomForest 4.7-1.1
```

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

```r
bag.model <- randomForest(STA ~ ., data = icu.train, mtry = 19, importance = T)
```

```r
# For training data
icu.train.pred <- predict(bag.model, data = icu.train)

# Confusion matrix
table(icu.train.pred, icu.train$STA)
```

```
##
## icu.train.pred die survive
##        die      12      11
##        survive  20      97
```

```r
# misclassification rate
mean(icu.train.pred != icu.train$STA)
```

```
## [1] 0.2214286
```

```r
# For testing data
icu.test.pred <- predict(bag.model, newdata = icu.test)

# Confusion matrix
table(icu.test.pred, icu.test$STA)
```

```
##
## icu.test.pred die survive
##       die       5       5
##       survive   3      47
```

```
# misclassification rate
mean(icu.test.pred != icu.test$STA)
```

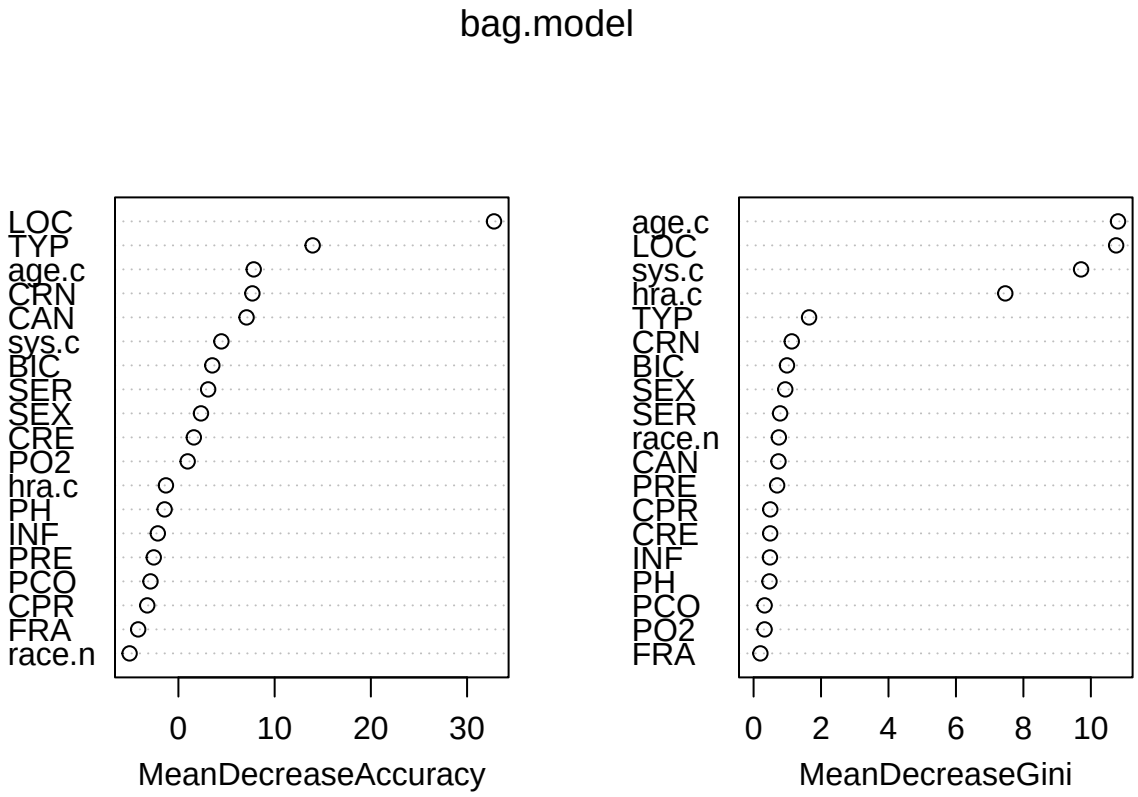
```
## [1] 0.1333333
```

b) Use the importance function on your Bagging model to identify the 6 most important predictors.

```
# Assess the important predictors using the varImpPlot
importance(bag.model)
```

```
##               die    survive MeanDecreaseAccuracy MeanDecreaseGini
## race.n -2.9881889 -5.1846928            -5.076056        0.7492179
## age.c   0.1919041  8.5595952             7.825084       10.8049564
## sys.c  -5.6997050  8.2800338             4.465767        9.7099454
## hra.c  -3.3970470  0.6656981            -1.298847        7.4625797
## SEX    -1.4651675  3.4325460             2.344846        0.9393143
## SER    -1.2736853  3.6916568             3.089471        0.7863559
## CAN    -4.2180142  9.0505726             7.083177        0.7375722
## CRN    -1.9561341  8.7950377             7.686593        1.1323909
## INF    -3.0430026 -0.3015418            -2.142706        0.4860537
## CPR    -4.4014742 -1.8338503            -3.238567        0.4941065
## PRE     0.8078667 -3.1018846            -2.563976        0.6984054
## TYP     3.0380907 13.5559856            13.950072        1.6456202
## FRA    -1.6609119 -4.0026075            -4.183330        0.2021530
## PO2    -1.5581668  1.7524329             0.959790        0.3258500
## PH     -1.9523887 -0.8423953            -1.436669        0.4714184
## PCO    -1.9672587 -2.1603295            -2.902313        0.3265955
## BIC     0.6824196  3.4407292             3.520671        0.9899627
## CRE     0.8426723  1.4906788             1.602019        0.4920947
## LOC    24.9242724 32.8295993            32.799333       10.7496643
```

```
varImpPlot(bag.model)
```



c) Apply Random Forest classification technique to predict the surviving status of an ICU patient using the predictors listed in part b). Report the model summary (confusion matrices of both the training and the testing data, report the corresponding misclassification rates)

```
set.seed(1128)
head(icu.train)
```

```
##         STA race.n   age.c  sys.c   hra.c    SEX                SER       CAN
## 105 survive  black -33.545   7.72 -12.925 Female Medical, Surgical1    cancer
## 35  survive  white  15.455   1.72 -38.925 Female Medical, Surgical2 no cancer
## 195     die  white  14.455 -42.28  61.075 Female Medical, Surgical1    cancer
## 114 survive  white   0.455  21.72  -0.925   Male Medical, Surgical1    cancer
## 103 survive  white  19.455   3.72  39.075 Female Medical, Surgical1    cancer
## 72  survive  white -39.545 -20.28 -22.925   Male Medical, Surgical2    cancer
##               CRN          INF CPR         PRE       TYP         FRA  PO2
## 105 renal failure no infection  no no previous emergency no fracture < 60
## 35  renal failure no infection  no no previous  elective no fracture < 60
## 195 renal failure no infection  no no previous emergency no fracture < 60
## 114 renal failure no infection  no no previous emergency no fracture < 60
## 103 renal failure    infection  no no previous  elective no fracture > 60
## 72  renal failure no infection  no no previous emergency    fracture < 60
##         PH PCO BIC CRE     LOC
```

```
## 105   7.25  45  18   2 no coma
## 35    7.25 >45  18   2 no coma
## 195   7.25  45  18   2 no coma
## 114   7.25  45  18   2 no coma
## 103 > 7.25 >45  18   2 no coma
## 72    7.25  45  18   2 no coma
```

```
icu.forest <- randomForest(STA~age.c + sys.c + LOC + TYP + CRN + CAN, data = icu.train, mtry = 6, impor

summary(icu.forest)
```

```
##                 Length Class  Mode
## call               5   -none- call
## type               1   -none- character
## predicted        140   factor numeric
## err.rate        1500   -none- numeric
## confusion          6   -none- numeric
## votes            280   matrix numeric
## oob.times        140   -none- numeric
## classes            2   -none- character
## importance        24   -none- numeric
## importanceSD      18   -none- numeric
## localImportance    0   -none- NULL
## proximity          0   -none- NULL
## ntree              1   -none- numeric
## mtry               1   -none- numeric
## forest            14   -none- list
## y                140   factor numeric
## test               0   -none- NULL
## inbag              0   -none- NULL
## terms              3   terms  call
```

```
# Using the above model, we will predict the y value
icu.forest.train <- predict(icu.forest, data = icu.train)


# The confusion matrix is below
table(icu.forest.train, icu.train$STA)
```

```
##
## icu.forest.train die survive
##          die      12      11
##          survive  20      97
```

```
# From the confusion matrix, we have the misclassification rate is
mean(icu.forest.train != icu.train$STA)
```

```
## [1] 0.2214286
```

```
# Moving on to the testing points, we have the following
icu.forest <- randomForest(STA~age.c + sys.c + LOC + TYP + CRN + CAN, data = icu.test, mtry = 6, importa
icu.forest.test <- predict(icu.forest, data = icu.test)
table(icu.forest.test, icu.test$STA)
```

```
##
## icu.forest.test die survive
##         die       2       2
##         survive   6      50
```

```
mean(icu.forest.test != icu.test$STA)
```

```
## [1] 0.1333333
```

We have that the misclassification rate for the training data is 0.1785714 and for the testing data is 0.1833333

d) What is Enough number of trees in both models (Bagging and Random Forest)?

```
set.seed(1128)

num <- c(seq(0.1, 0.9, 0.5), 1:200)
icu.mse.bag <- c()
icu.mse.forest <- c()
k <- 1

for (i in num){
  icu.bagging <- randomForest(STA ~ ., data = icu.train, mtry = 19, importance = T, ntree = 10*i)
  icu.forest <- randomForest(STA~age.c + sys.c + LOC + TYP + CRN + CAN, data = icu.train, mtry = 6, imp

  pred.bag <- predict(icu.bagging, newdata = icu.test)
  pred.forest <- predict(icu.forest, newdata = icu.test)
  error.bag <- mean(icu.test$STA != pred.bag)
  error.forest <- mean(icu.test$STA != pred.forest)

  icu.mse.bag[k] <- mean(error.bag)
  icu.mse.forest[k] <- mean(error.forest)
  k <- k + 1
}

which.min(icu.mse.bag)*10
```

```
## [1] 100
```

```
which.min(icu.mse.forest)*10
```

```
## [1] 10
```

```
library(ggplot2)
```

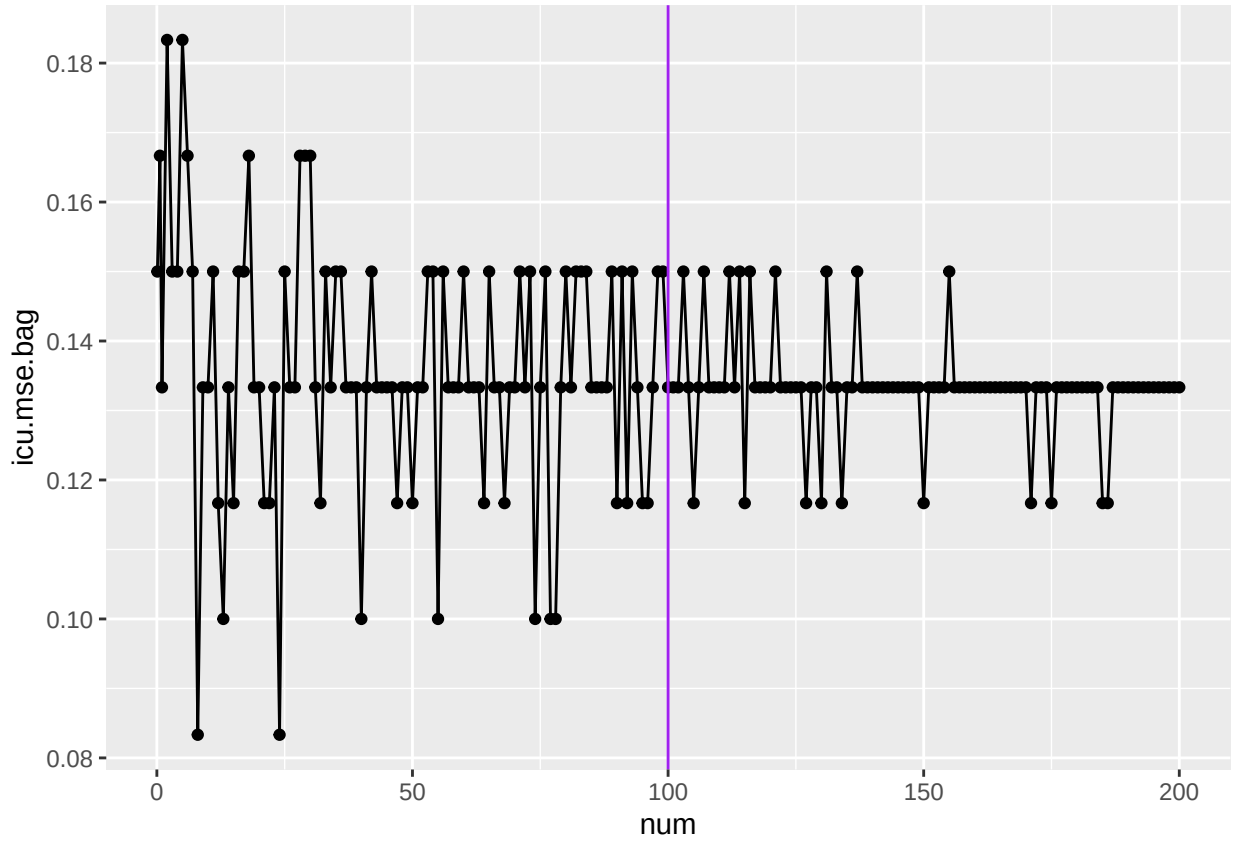
```
## Warning: package 'ggplot2' was built under R version 4.1.2
```

```
##
## Attaching package: 'ggplot2'
```

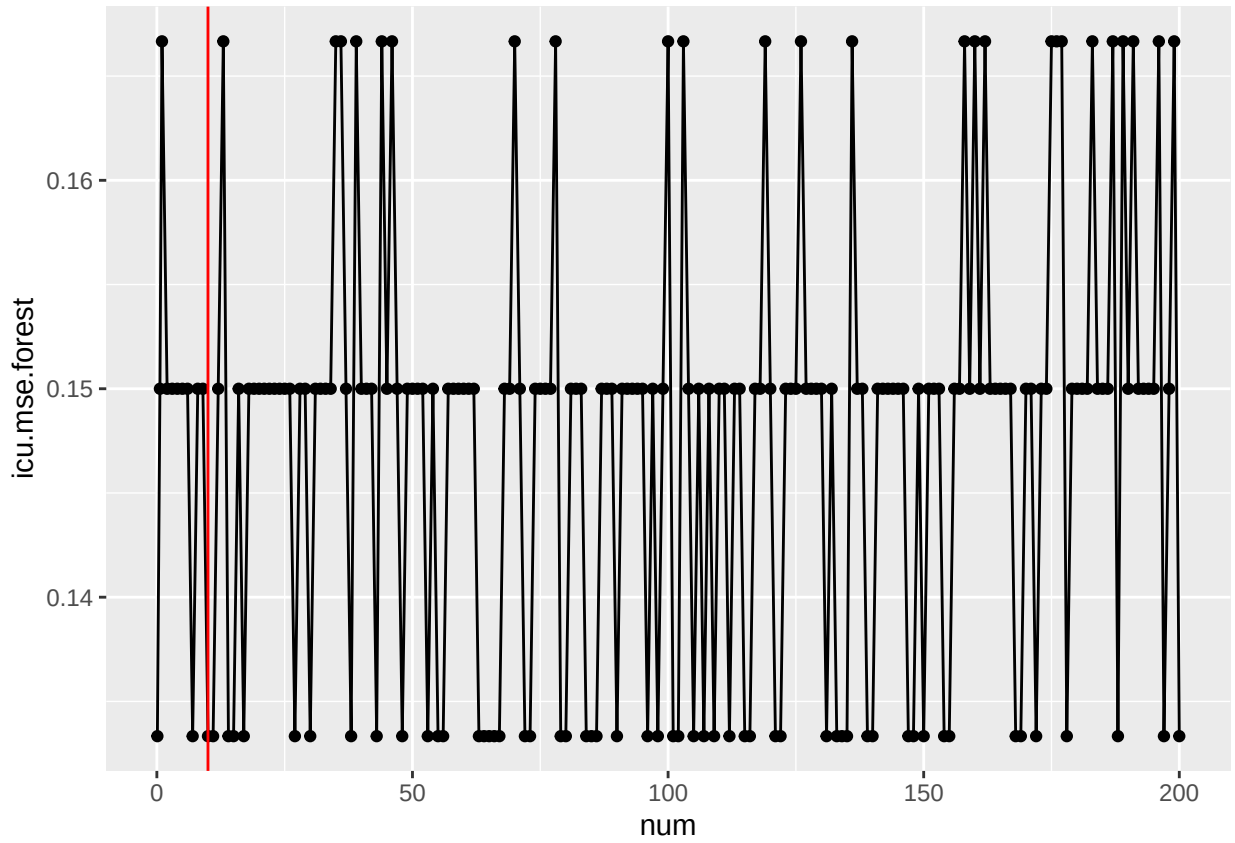
```
## The following object is masked from 'package:randomForest':
##
##     margin
```

```
qplot(num, icu.mse.bag) + geom_line() + geom_vline(xintercept = which.min(icu.mse.bag)*10, col = "purpl
```

```
## Warning: `qplot()` was deprecated in ggplot2 3.4.0.
```




```
qplot(num, icu.mse.forest) + geom_line() + geom_vline(xintercept = which.min(icu.mse.forest)*10, col =
```



We can see that the purple and red verticle line in the two graphs above is distinct. We have that for the bagging, we have that the enough number of trees is 10 and for forest, we have that the enough number of trees is 100.

Q4) Consider the USArrest data (Built-in data Posted on week 9). We will now perform hierarchical clustering on the state.

```
ar <- USArrests
dim(ar)
```

```
## [1] 50  4
```

```
head(ar)
```

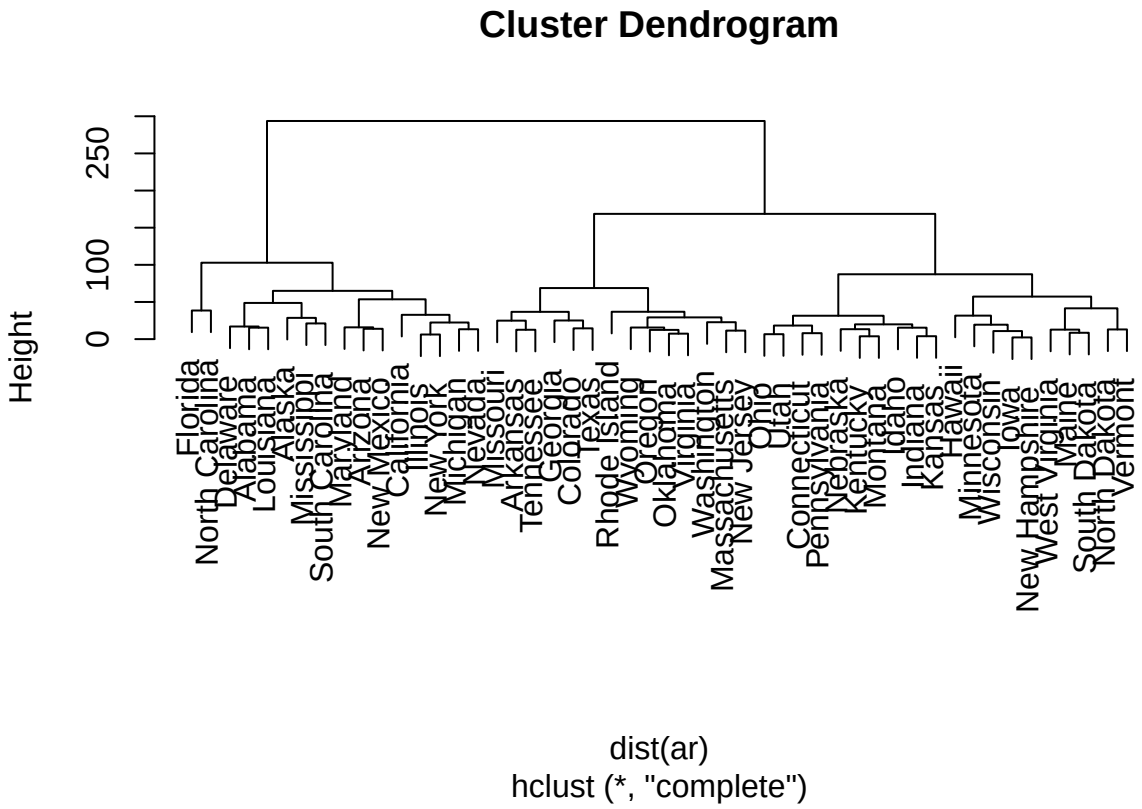
```
##            Murder Assault UrbanPop Rape
## Alabama      13.2     236       58 21.2
## Alaska       10.0     263       48 44.5
## Arizona       8.1     294       80 31.0
## Arkansas      8.8     190       50 19.5
## California    9.0     276       91 40.6
## Colorado      7.9     204       78 38.7
```

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

```
hclust.ar <- hclust(dist(ar), method = "complete")
hclust.ar
```

```
##
## Call:
## hclust(d = dist(ar), method = "complete")
##
## Cluster method   : complete
## Distance         : euclidean
## Number of objects: 50
```

```
plot(hclust.ar)
```



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

```
hclust.ar.3 <- cutree(hclust.ar, k = 3)
```

```r
# In the output below, we can see that all of the states are provided the values either 1, 2, or 3
# We can see which states belong to which clusters in below
hclust.ar.3
```

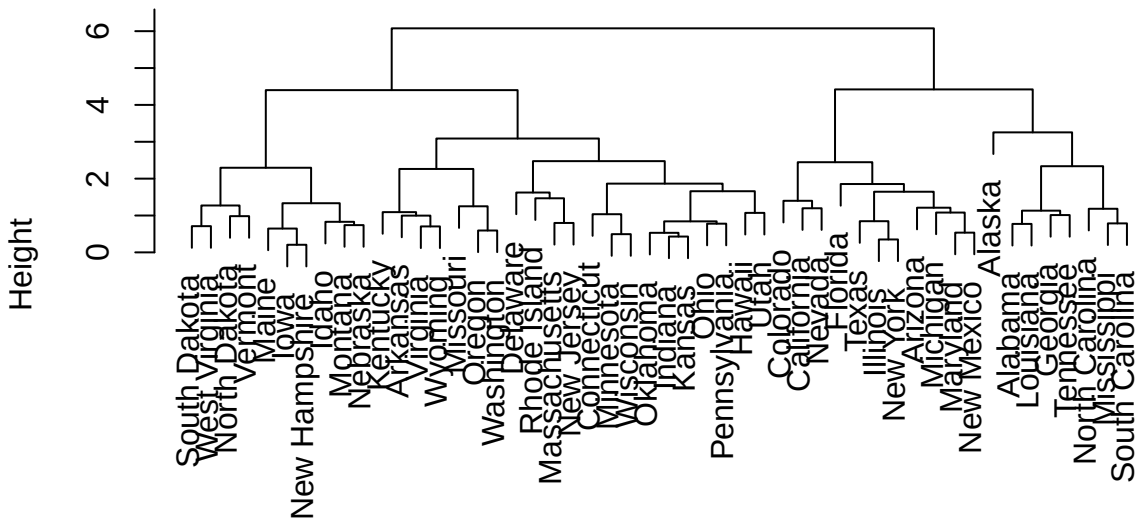
```
##        Alabama         Alaska        Arizona       Arkansas     California
##              1              1              1              2              1
##       Colorado    Connecticut       Delaware        Florida        Georgia
##              2              3              1              1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

```r
# The frequencies for the amount of states in each clusters
table(hclust.ar.3)
```

```
## hclust.ar.3
##  1  2  3
## 16 14 20
```

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

```r
# First we aim to scale the variables
ar.scale <- scale(ar)
hclust.ar.scale <- hclust(dist(ar.scale), method = "complete")
plot(hclust.ar.scale)
```




```
# In the output below, we can see that all of the states are provided the values either 1, 2, or 3
# We can see which states belong to which clusters in below
hclust.ar.scale.3 <- cutree(hclust.ar.scale, k = 3)
hclust.ar.scale.3
```

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

```r
# The frequencies for the amount of states in each clusters
table(hclust.ar.scale.3)
```

```
## hclust.ar.scale.3
##  1  2  3
##  8 11 31
```

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

There is evident effect on the hierarchical clustering obtained when scaling the variables. This can be seen through looking at the frequency of each state in the three clusters. Prior to scaling, we have that for cluster 1,2,3 the frequencies are 16,14,20 respectively. After scaling, we see that for cluster 1,2,3 the frequencies are 8,11,31 respectively. Further, scaling the variables impacts the branch lengths, height of the trees, and the clusters that are obtained. In my opinion, the variables should be scaled before the inter-observation dissimilarities are computed. In this way, the variables can be more comparable and can draw a better result.

Q5) Consider the Olives data posted on ccle week 9.

> names(olive)

[1] "region" "area" "palmitic" "palmitoleic"

[5] "stearic" "oleic" "linoleic" "linolenic"

[9] "arachidic" "eicosenoic"

```r
temp <- read.csv("/Users/takaooba/Downloads/Olives.csv")
temp <- temp[,-1]
dim(temp)
```

```
## [1] 572  10
```

```r
head(temp)
```

```
##   region          area palmitic palmitoleic stearic oleic linoleic linolenic
## 1      1 North-Apulia      1075          75     226  7823      672        36
## 2      1 North-Apulia      1088          73     224  7709      781        31
## 3      1 North-Apulia       911          54     246  8113      549        31
## 4      1 North-Apulia       966          57     240  7952      619        50
## 5      1 North-Apulia      1051          67     259  7771      672        50
## 6      1 North-Apulia       911          49     268  7924      678        51
##   arachidic eicosenoic
## 1        60         29
## 2        61         29
## 3        63         29
## 4        78         35
## 5        80         46
## 6        70         44
```

a) Ignore the variable "area" from your data set. Convert the variable "region" to a factor variable.

```
olive_1 <- temp[,-2]
olive_1$region <- as.factor(olive_1$region)

# We can see that the region has a class of fctr
head(olive_1)
```

```
##   region palmitic palmitoleic stearic oleic linoleic linolenic arachidic
## 1      1     1075          75     226  7823      672        36        60
## 2      1     1088          73     224  7709      781        31        61
## 3      1      911          54     246  8113      549        31        63
## 4      1      966          57     240  7952      619        50        78
## 5      1     1051          67     259  7771      672        50        80
## 6      1      911          49     268  7924      678        51        70
##   eicosenoic
## 1         29
## 2         29
## 3         29
## 4         35
## 5         46
## 6         44
```

b) Use all the numerical variables (all but area and region) to create 3 clusters. First use k-means to create your 3 clusters, the use hierarchal clustering to create your three clusters (Use Average Linkage).

```
head(olive_1)
```

```
##   region palmitic palmitoleic stearic oleic linoleic linolenic arachidic
## 1      1     1075          75     226  7823      672        36        60
## 2      1     1088          73     224  7709      781        31        61
## 3      1      911          54     246  8113      549        31        63
## 4      1      966          57     240  7952      619        50        78
## 5      1     1051          67     259  7771      672        50        80
## 6      1      911          49     268  7924      678        51        70
##   eicosenoic
## 1         29
## 2         29
## 3         29
## 4         35
## 5         46
## 6         44
```

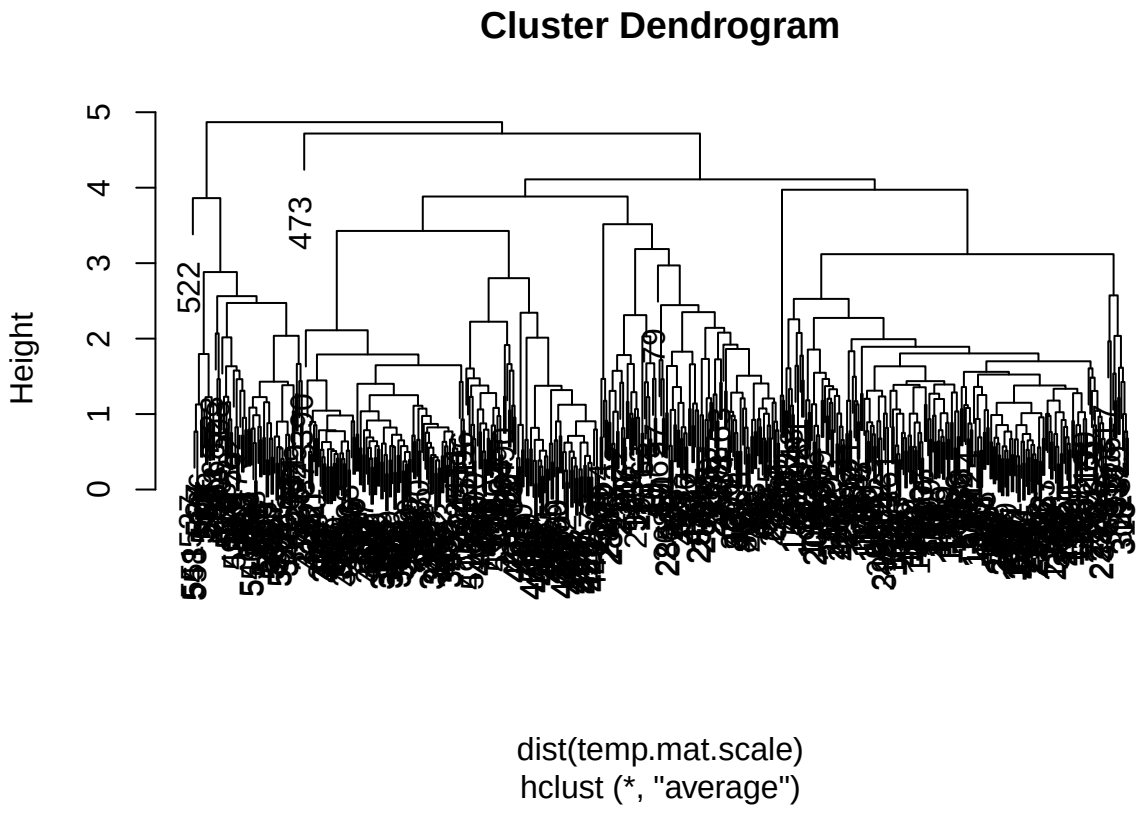
```
# Looking at the predictors which will be column 2 to 9
temp.mat <- olive_1[, 2:9]
prop.table(table(olive_1$region))
```

```
##
##         1         2         3
## 0.5646853 0.1713287 0.2639860
```

```
set.seed(1128)
out <- kmeans(temp.mat, 3, nstart = 50)
table(out$cluster, olive_1$region)
```

```
##
##       1   2   3
##   1  42   0 134
##   2 190  22   0
##   3  91  76  17
```

```
temp.mat.scale <- scale(temp.mat)


# Performing a hierarchal clustering
h.avg <- hclust(dist(temp.mat.scale), method = "average")
plot(h.avg)
```

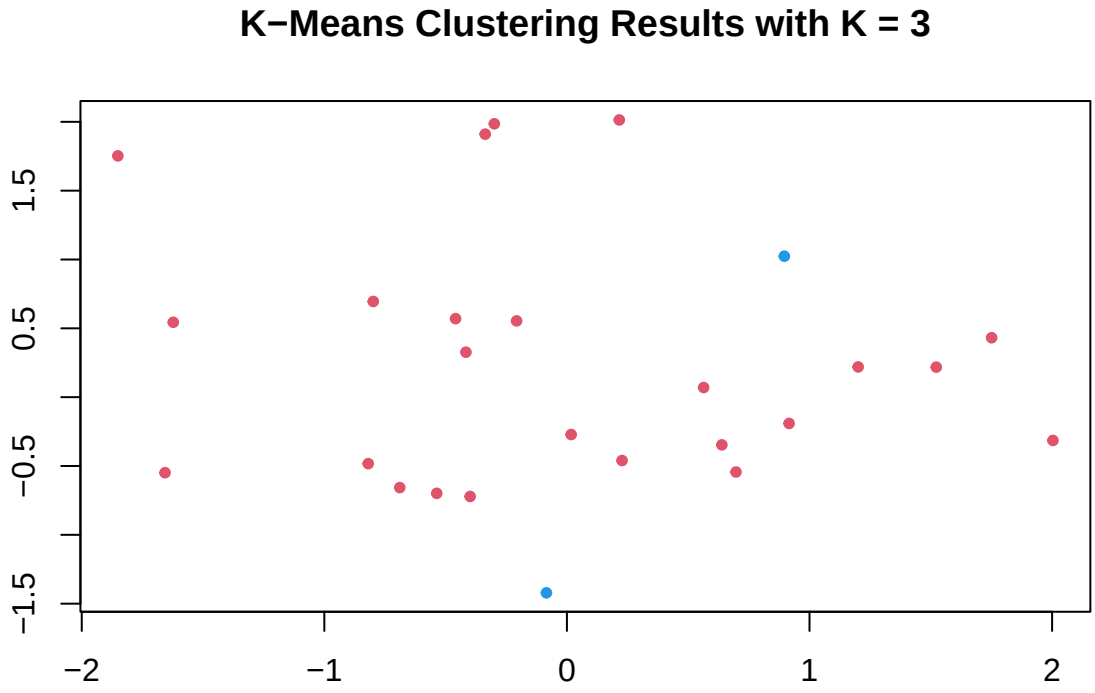



```
# cutting the tree to 3 clusters
h_comp_olive <- cutree(h.avg, 3)

table(h_comp_olive)
```

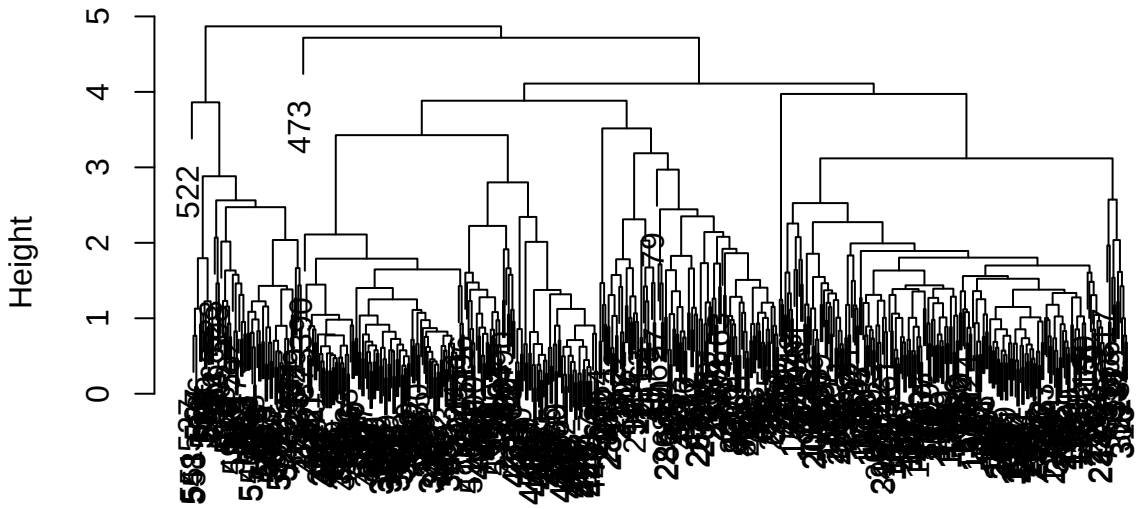
```
## h_comp_olive
##   1   2   3
## 503   1  68
```

c) Provide plots and statistical summaries of your clustering methods.

```
set.seed(1128)
x <- matrix(rnorm(52), ncol = 2)


plot(x, col = (out$cluster + 1), main = "K-Means Clustering Results with K = 3", xlab = "", ylab = "", p
```




```
plot(h.avg)
```




```
# two summary functions
summary(out)
```

```
##              Length Class  Mode
## cluster      572    -none- numeric
## centers       24    -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
```

```
summary(h.avg)
```

```
##             Length Class  Mode
## merge       1142   -none- numeric
## height       571   -none- numeric
## order        572   -none- numeric
## labels         0   -none- NULL
## method         1   -none- character
## call           3   -none- call
## dist.method    1   -none- character
```

d) Evaluate the success of the clustering by comparing to the three regions in the original data. What did you notice?

```
# Original Data
table(temp$region)
```

```
##
##   1   2   3
## 323  98 151
```

Above is for the original data set. We have that for region 1,2,3 the frequency is 323, 98, and 151 respectively.

Compare this to below, which is from clustering.

```
# Success of clustering
table(out$cluster)
```

```
##
##   1   2   3
## 176 212 184
```

Notice that the frequencies has drastically changed. For above, we have that the frequency 176, 212, 184 respectively. We have that the frequencies became more even or there are similar amount of points in each clusters compared to the original data. In the original data, we have that the frequency for 1,2,3 are fluctuating.

e) Perform PCA on the numerical variables. Take the first two principal components. Report the amount of variation explained by those two components.

```
pca.olive <- prcomp(temp.mat, scale = T)
summary(pca.olive)
```

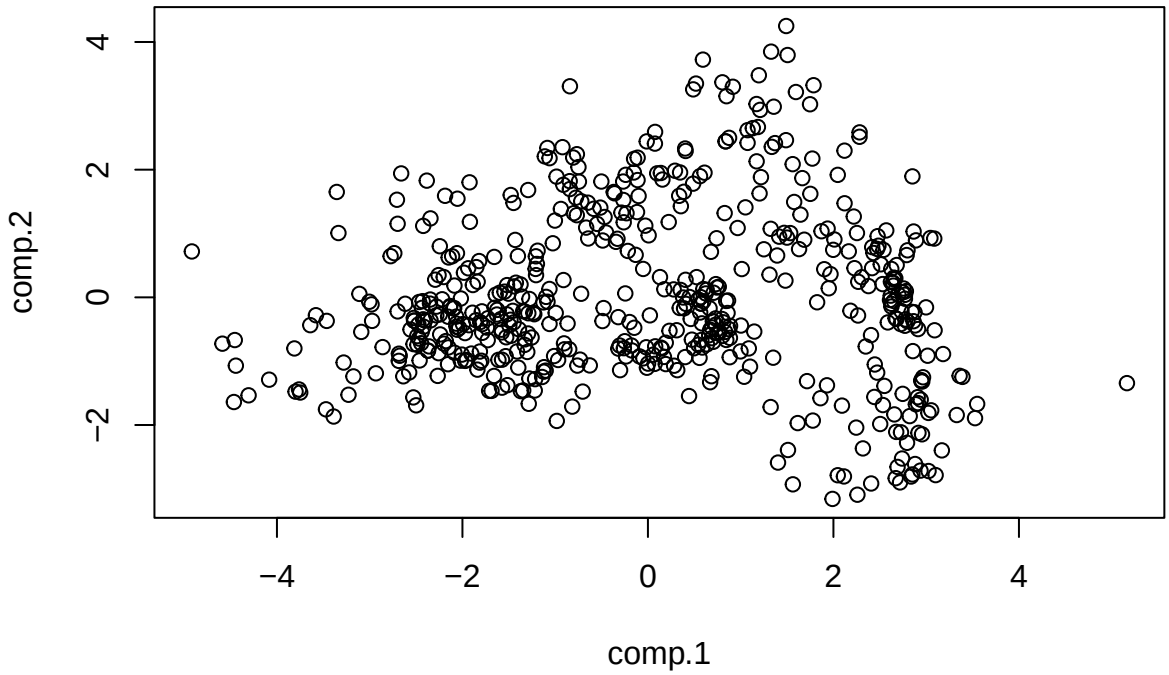
```
## Importance of components:
##                           PC1    PC2    PC3     PC4     PC5    PC6     PC7
## Standard deviation     1.9291 1.3288 1.0081 0.89045 0.57777 0.4988 0.34470
## Proportion of Variance 0.4652 0.2207 0.1270 0.09911 0.04173 0.0311 0.01485
## Cumulative Proportion  0.4652 0.6859 0.8129 0.91206 0.95378 0.9849 0.99974
##                           PC8
## Standard deviation     0.04563
## Proportion of Variance 0.00026
## Cumulative Proportion  1.00000
```

From the output above, we look at the row "Proportion of Variance" and "Cumulative Proportion"

Specifically examining the first two principal components, we have that PC1 has a proportion of variance 0.4652 and PC2 is 0.2207. Overall, the first two principal components explains 68.59% of variance.

f) Plot PCA1 vs PCA2 and color them based on region. Perform k-means clustering with k=3 on PCA1 and PCA2. Plot PCA1 vs PCA2 and color them based on three clusters. What did you notice?

```
princomp.olive <- princomp(temp.mat, cor = T)
comp <- princomp.olive$scores
comp.1 <- -1*comp[,1]
comp.2 <- -1*comp[,2]

# We will be clustering below
plot(comp.1, comp.2)
```



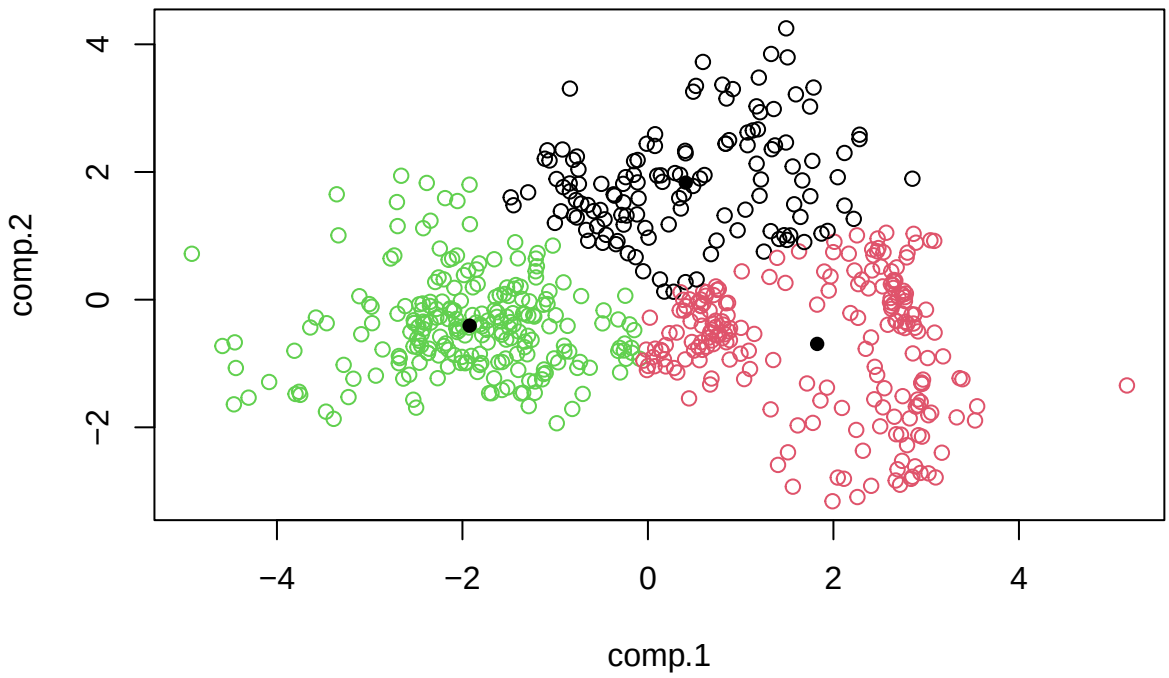

```
summary(princomp.olive)
```

```
## Importance of components:
##                           Comp.1    Comp.2    Comp.3     Comp.4     Comp.5
## Standard deviation     1.9290956 1.3288331 1.0081446 0.89044867 0.57776956
## Proportion of Variance 0.4651763 0.2207247 0.1270444 0.09911235 0.04172721
## Cumulative Proportion  0.4651763 0.6859009 0.8129454 0.91205772 0.95378493
##                           Comp.6     Comp.7       Comp.8
## Standard deviation     0.49881727 0.34470293 0.0456263329
## Proportion of Variance 0.03110233 0.01485251 0.0002602203
## Cumulative Proportion  0.98488727 0.99973978 1.0000000000
```

```r
temp <- cbind(comp.1, comp.2)
cl <- kmeans(temp, 3)
cl$cluster
```

```
##   [1] 1 1 1 1 1 1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1
##  [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 3 1 1 1 1 1 1 1 1 3 1 1 1 1
##  [75] 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 3 3 3 3 3 3
## [112] 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
## [149] 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
## [186] 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
## [223] 3 3 3 3 3 3 3 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 3 3 1 3
## [260] 3 3 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 3 3 3 3 1 3 1 1 1 1 1 1 1 1 1 1 3 3 3
## [297] 3 2 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 2 2 2 2 2 2
## [334] 2 2 3 2 2 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2
## [371] 3 2 3 3 2 3 2 2 3 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 1 2 2 1 2 2 2 2 2 2
## [408] 2 2 2 3 2 2 3 2 2 2 2 2 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [445] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 1 2 1 2 2 2 2 1 1 2 1 1 2 2 2 1
## [482] 1 1 1 1 1 2 1 1 1 1 2 2 2 2 2 1 1 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 1 2 2
## [519] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [556] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

```r
# The clustered points with 3 groups is as follows
# Notice that there are three colors: green, red, and black
plot(comp.1, comp.2, col = cl$cluster)

points(cl$centers, pch= 16)
```




```
table(cl$cluster)
```

```
##
##   1   2   3
## 131 212 229
```

We were able to cluster a group of points into three clusters utilizing k-means. Comparing to the prior scatterplot, we are able to see that the computer is able to cluster the points fairly well. However, notice that there are minor overlaps with certain points in different region which indicates that the clustering is not perfect.