Stats 101C Homework 6

Christy Hui

Due 12/03/2021

Problem 1

Part A

```
births = read.csv("better2000births.csv")
sum(is.na(births))
```

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

```
dim(births)
```

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

Instructions are incorrect. There are no NAs and there are 1998 observations (not 2000).

```
set.seed(1128)
# factor all non-numeric for trees to work
births$Gender = as.factor(births$Gender)
births$Premie = as.factor(births$Premie)
births$Marital = as.factor(births$Marital)
births$Racemom = as.factor(births$Racemom)
births$Racedad = as.factor(births$Racedad)
births$Hispmom = as.factor(births$Hispmom)
births$Hispdad = as.factor(births$Hispdad)
births$Habit = as.factor(births$Habit)
births$MomPriorCond = as.factor(births$MomPriorCond)
births$BirthDef = as.factor(births$BirthDef)
births$DelivComp = as.factor(births$DelivComp)
births$BirthComp = as.factor(births$BirthComp)
split = sample(dim(births)[1], 1000, replace = FALSE)
# split data into training and testing
births.train = births[split,]
births.test = births[-split,]
dim(births.train)
```

```
## [1] 1000   21
```

```
dim(births.test)
```

```
## [1] 998  21
```

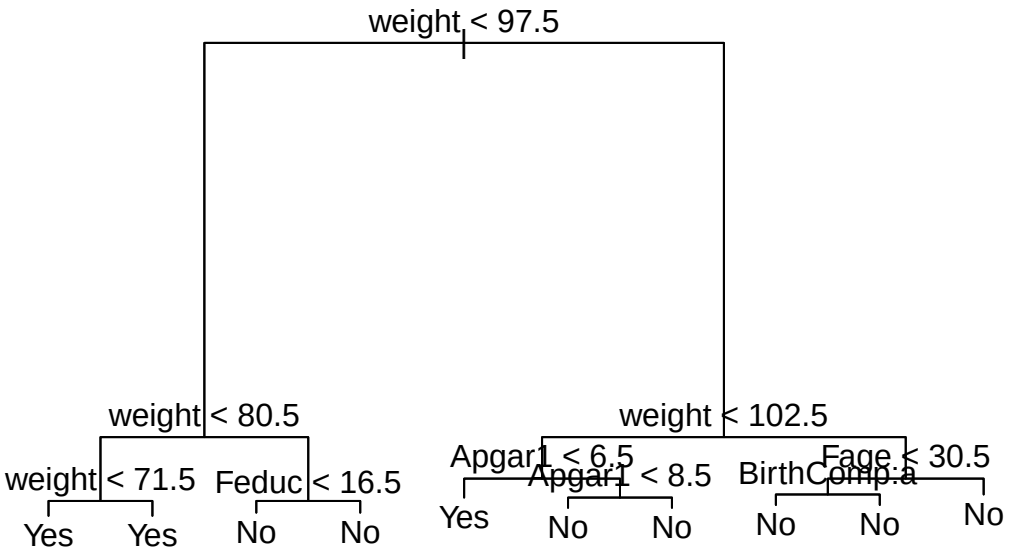
With a prepped data set, we can now work on the models.

```
library(tree)
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
births.tree = tree(Premie~., data = births.train)
plot(births.tree)
text(births.tree)
```




```
summary(births.tree)
```

```
##
## Classification tree:
## tree(formula = Premie ~ ., data = births.train)
## Variables actually used in tree construction:
## [1] "weight"    "Feduc"     "Apgar1"    "Fage"      "BirthComp"
## Number of terminal nodes:  10
## Residual mean deviance:  0.2457 = 243.2 / 990
## Misclassification error rate: 0.051 = 51 / 1000
```

```
births.tree.pred = predict(births.tree, newdata = births.test, type = "class")
confusionMatrix(as.factor(births.tree.pred), as.factor(births.test$Premie))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  895  49
##        Yes  13  41
##
##                Accuracy : 0.9379
##                  95% CI : (0.9211, 0.952)
##     No Information Rate : 0.9098
##     P-Value [Acc > NIR] : 0.0007252
##
##                   Kappa : 0.5382
##
##  Mcnemar's Test P-Value : 8.789e-06
##
##             Sensitivity : 0.9857
##             Specificity : 0.4556
##          Pos Pred Value : 0.9481
##          Neg Pred Value : 0.7593
##              Prevalence : 0.9098
##          Detection Rate : 0.8968
##    Detection Prevalence : 0.9459
##       Balanced Accuracy : 0.7206
##
##        'Positive' Class : No
##
```

```
mean(births.tree.pred != births.test$Premie)
```

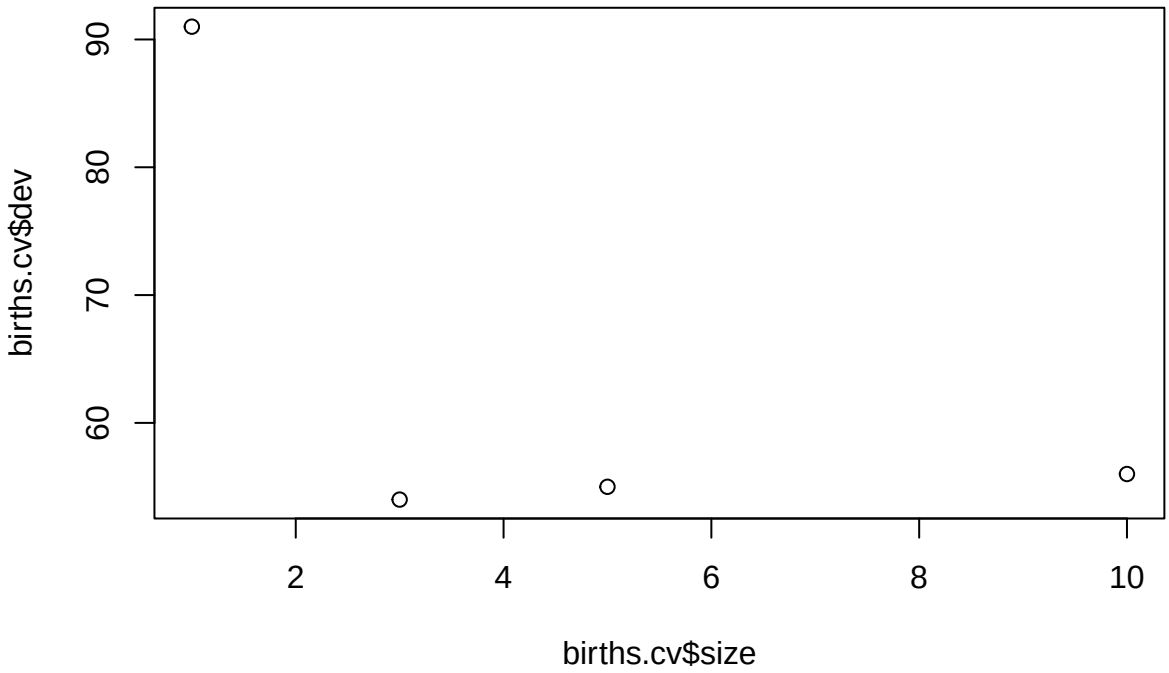
```
## [1] 0.06212425
```

The training misclassification error rate is 51/1000 (or 5.1%).

The testing misclassification error rate is 62/998 (or 6.21242%).

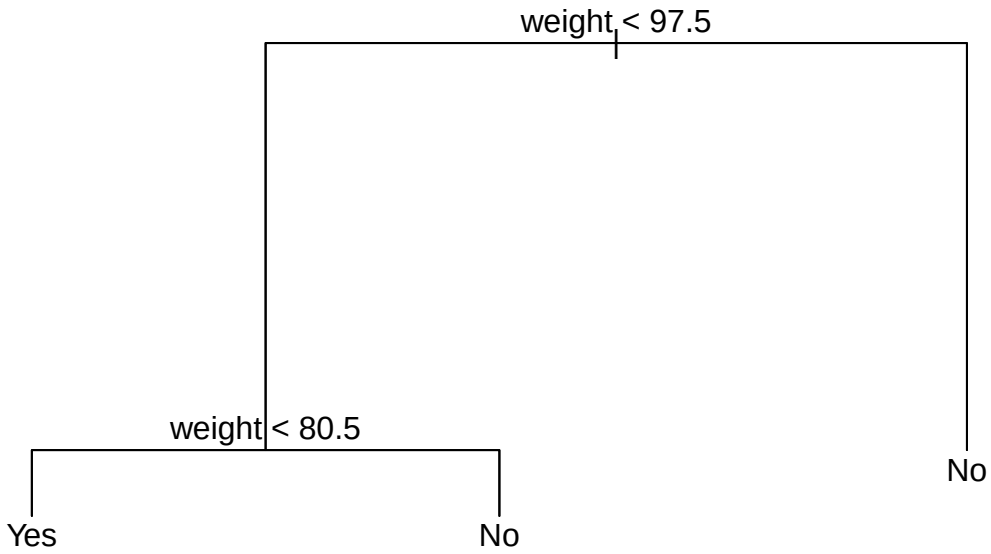
Part B

```
births.cv = cv.tree(births.tree, FUN = prune.misclass)
plot(births.cv$dev~births.cv$size)
```



By the above plot, we see (by CV) that the best amount of nodes needed is 3. Thus, we prune the tree to 3 nodes.

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



Above is the plot for the pruned tree.

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

```
##
## Classification tree:
## snip.tree(tree = births.tree, 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.pruned.fit.pred = predict(births.pruned.fit, newdata = births.test, type = "class")
confusionMatrix(as.factor(births.pruned.fit.pred),
                as.factor(births.test$Premie))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  901  50
##        Yes   7  40
##
##                Accuracy : 0.9429
##                  95% CI : (0.9266, 0.9565)
##     No Information Rate : 0.9098
##     P-Value [Acc > NIR] : 7.004e-05
##
##                   Kappa : 0.5565
##
##  Mcnemar's Test P-Value : 2.651e-08
##
##             Sensitivity : 0.9923
##             Specificity : 0.4444
##          Pos Pred Value : 0.9474
##          Neg Pred Value : 0.8511
##              Prevalence : 0.9098
##          Detection Rate : 0.9028
##    Detection Prevalence : 0.9529
##       Balanced Accuracy : 0.7184
##
##        'Positive' Class : No
##
```

```
mean(births.pruned.fit.pred != births.test$Premie)
```

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

The misclassification rate for the training data (with 3 nodes) is 54/1000 (or 5.4%).

The misclassification rate for the testing data (with 3 nodes) is 57/998 (or 5.71%).

This is a bit better than the regular tree. Thus, we conclude that this pruned tree performs better.

Part C

According to the pruned tree, we see the "weight" variable being the only predictor needed in order to tell whether or not a baby is premature. Smoking is NOT a potential cause of premature births according to this tree.

Part D

```
mean((births.pruned.fit.pred) != (births.test$Premie))
```

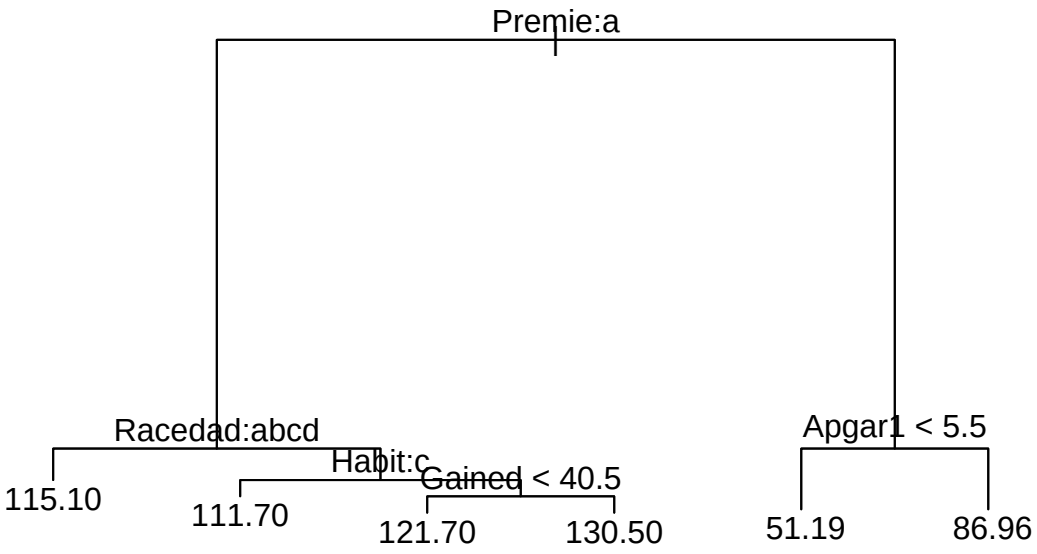
```
## [1] 0.05711423
```

The misclassification rate for the testing data (with 3 nodes) is 0.05711423. If a doctor always only has a 9% misclassification error, we can conclude that he or she does worse than the tree models. In other words, our tree models perform better than an average doctor.

Problem 2

Part A

```
birthsweight.tree = tree(weight~., data = births.train)
plot(birthsweight.tree)
text(birthsweight.tree)
```




```
summary(birthsweight.tree)
```

```
##
## Regression tree:
## tree(formula = weight ~ ., data = births.train)
## Variables actually used in tree construction:
## [1] "Premie"  "Racedad" "Habit"   "Gained"  "Apgar1"
## Number of terminal nodes:  6
## Residual mean deviance:  249.4 = 247900 / 994
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -58.9600  -9.7400  -0.4828   0.0000   9.2600  69.0400
```

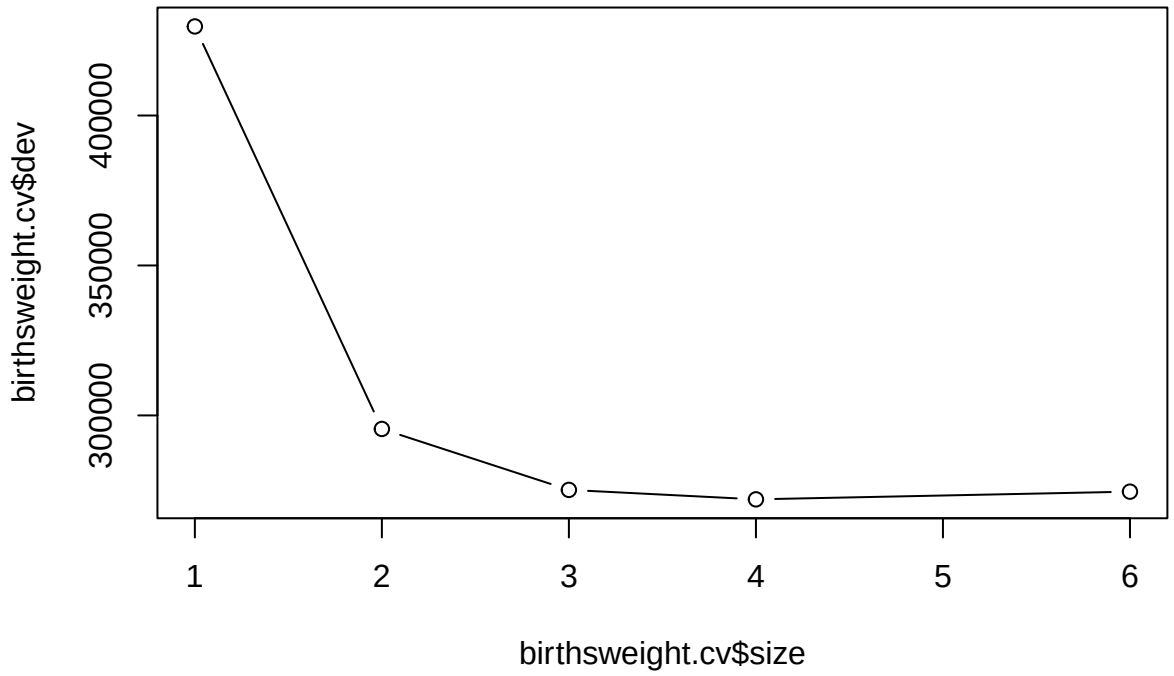
```
birthsweight.tree.pred = predict(birthsweight.tree, newdata = births.test)
birthsweight.tree.mse = mean((birthsweight.tree.pred - births.test$weight)^2)
birthsweight.tree.mse
```

```
## [1] 274.6267
```

The testing MSE is 274.6267.

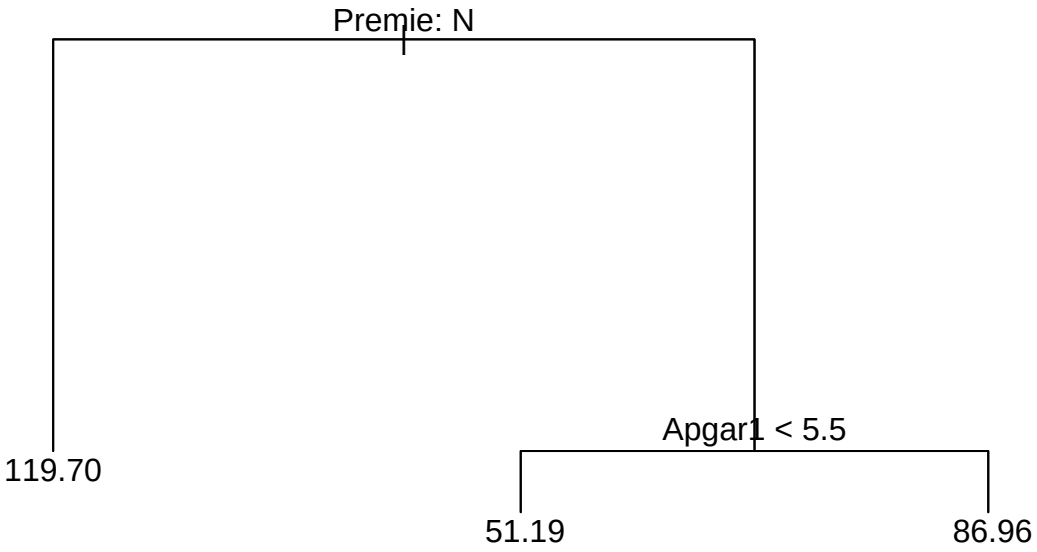
Part B

```
birthsweight.cv = cv.tree(birthsweight.tree, FUN = prune.tree)
plot(birthsweight.cv$dev ~ birthsweight.cv$size, type = "b" )
```



By the above plot, we see that 4 nodes performs the best. However, to perhaps prevent over fitting, let us use 3.

```
birthsweight.pruned.fit = prune.tree(birthsweight.tree, best = 3)
plot(birthsweight.pruned.fit)
text(birthsweight.pruned.fit, pretty = TRUE)
```



Above is the plot for a pruned tree with 3 nodes.

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

```
##
## Regression tree:
## snip.tree(tree = birthsweight.tree, nodes = 2L)
## Variables actually used in tree construction:
## [1] "Premie" "Apgar1"
## Number of terminal nodes:  3
## Residual mean deviance:  270.5 = 269700 / 997
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -58.9600 -10.6900  -0.6887   0.0000  10.3100  69.0400
```

```
birthsweight.pruned.fit.pred = predict(births.pruned.fit, newdata = births.test)
```

Part C

To predict a baby's weight, the best predictor used is "Premie." Another important predictor in predicting a baby's weight is Apgar1. The number of visits predictor is NOT an important feature in predicting baby weight.

Part D

```
birthsweight.pruned.fit.mse = mean((birthsweight.pruned.fit.pred - births.test$weight)^2)
birthsweight.pruned.fit.mse
```

```
## [1] 13780.34
```

Above is the testing MSE with the pruned tree.

Problem 3

```
icu = read.csv("icu_data.csv")
sum(is.na(icu))
```

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

```
# factor all non-numeric for trees to work
icu$STA = as.factor(icu$STA)
icu$race.n = as.factor(icu$race.n)
icu$SER = as.factor(icu$SER)
icu$CAN = as.factor(icu$CAN)
icu$CRN = as.factor(icu$CRN)
icu$INF = as.factor(icu$INF)
icu$CPR = as.factor(icu$CPR)
icu$PRE = as.factor(icu$PRE)
icu$TYP = as.factor(icu$TYP)
icu$FRA = as.factor(icu$FRA)
icu$LOC = as.factor(icu$LOC)
# split data
split = sample(dim(icu)[1], dim(icu)[1]*0.70, replace = FALSE)
icu.train = icu[split,]
icu.test = icu[-split,]
dim(icu.train)
```

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

```
dim(icu.test)
```

```
## [1] 60 20
```

Part A

```
library(randomForest)
```

```
## randomForest 4.6-14
```

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

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
icu.bag = randomForest(STA ~., data = icu.train, mtry = 19, importance = TRUE)
summary(icu.bag)
```

```
##                 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        76   -none- numeric
## importanceSD      57   -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
```

```
icu.bag
```

```
##
## Call:
##  randomForest(formula = STA ~ ., data = icu.train, mtry = 19,      importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 19
##
##         OOB estimate of  error rate: 17.86%
## Confusion matrix:
##         die survive class.error
## die      13      17  0.56666667
## survive   8     102  0.07272727
```

```
# train confusion matrix
icu.bag.pred = predict(icu.bag, newdata = icu.train)
confusionMatrix(as.factor(icu.bag.pred),
                as.factor(icu.train$STA))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction die survive
##    die      30       0
##    survive   0     110
##
##                Accuracy : 1
```

```
##                  95% CI : (0.974, 1)
##     No Information Rate : 0.7857
##     P-Value [Acc > NIR] : 2.173e-15
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.0000
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 1.0000
##              Prevalence : 0.2143
##          Detection Rate : 0.2143
##    Detection Prevalence : 0.2143
##       Balanced Accuracy : 1.0000
##
##        'Positive' Class : die
##
```

```
# test confusion matrix
icu.bag.pred = predict(icu.bag, newdata = icu.test)
confusionMatrix(as.factor(icu.bag.pred),
                as.factor(icu.test$STA))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction die survive
##    die       4       4
##    survive   6      46
##
##                Accuracy : 0.8333
##                  95% CI : (0.7148, 0.9171)
##     No Information Rate : 0.8333
##     P-Value [Acc > NIR] : 0.5834
##
##                   Kappa : 0.3478
##
##  Mcnemar's Test P-Value : 0.7518
##
##             Sensitivity : 0.40000
##             Specificity : 0.92000
##          Pos Pred Value : 0.50000
##          Neg Pred Value : 0.88462
##              Prevalence : 0.16667
##          Detection Rate : 0.06667
##    Detection Prevalence : 0.13333
##       Balanced Accuracy : 0.66000
##
##        'Positive' Class : die
##
```

The misclassification rate of the training data is 0%

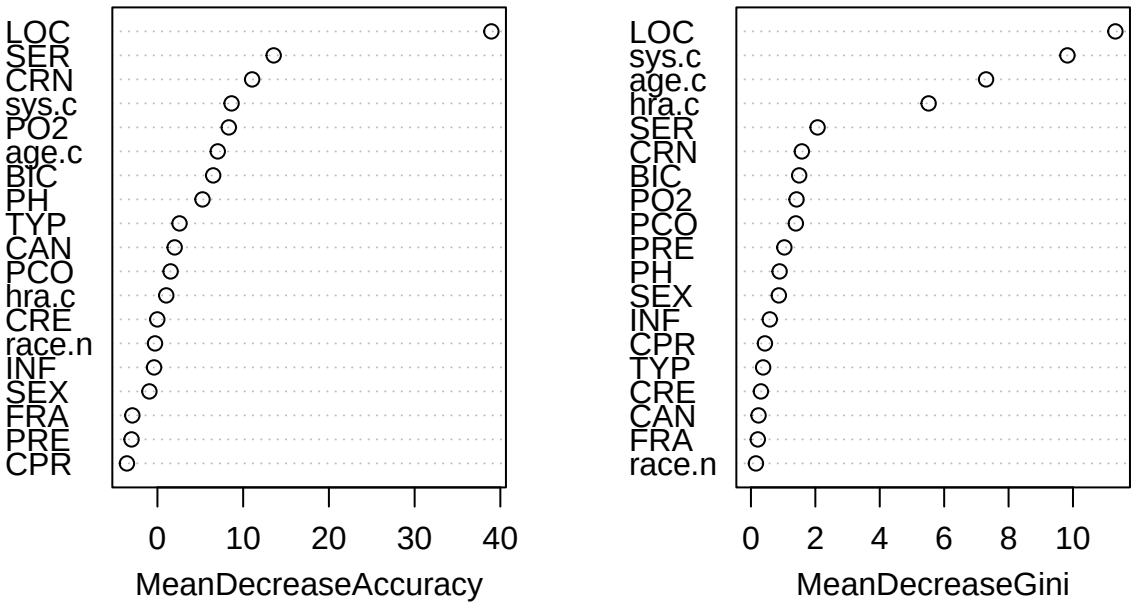
The misclassification rate of the testing data is 30%

Part B

```
importance(icu.bag)
```

```
##               die    survive MeanDecreaseAccuracy MeanDecreaseGini
## race.n  1.0010015 -0.4370116          -0.28382578        0.1552920
## age.c   3.1140848  7.0651771           7.03474815        7.3008217
## sys.c  -0.3192069 10.6786189           8.64815190        9.8276211
## hra.c  -4.0484097  3.1916974           1.02874064        5.5178822
## SEX    -2.1861929  0.2510228          -0.93879420        0.8662525
## SER     4.9437570 13.4398947          13.55936521        2.0694116
## CAN    -3.0356737  2.7389640           2.00884031        0.2359769
## CRN     4.1744168 10.4605754          11.04870571        1.5809528
## INF     3.4744806 -1.9545907          -0.39371796        0.5858956
## CPR    -3.7384482 -2.4595720          -3.56066013        0.4338141
## PRE     1.8316560 -4.2334886          -3.01998364        1.0366804
## TYP     4.2741597  1.6729945           2.57469027        0.3789318
## FRA    -1.7212608 -2.7132497          -2.92970233        0.2129824
## PO2    -1.1594394  9.5035101           8.32419476        1.4164422
## PH     -3.2034369  6.8141380           5.25694471        0.8902185
## PCO    -2.9834704  4.5200856           1.53438578        1.3945108
## BIC     0.7110797  6.5448557           6.50452515        1.4985695
## CRE    -2.1792245  1.1415042          -0.01660115        0.3100408
## LOC    28.8011481 37.8324865          38.95168512       11.3167032
```

```
varImpPlot(icu.bag)
```



The 6 most important predictors according to accuracy is LOC, SER, CRN, sys.c, PO2, age.c, and BIC.

Part C

```
icu.rf = randomForest(STA ~., data = icu.train, mtry = 6, importance = TRUE)
summary(icu.rf)
```

```
##                 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        76   -none- numeric
## importanceSD      57   -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
```

```
icu.rf
```

```
##
## Call:
##  randomForest(formula = STA ~ ., data = icu.train, mtry = 6, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 6
##
##         OOB estimate of  error rate: 17.86%
## Confusion matrix:
##         die survive class.error
## die      11      19  0.63333333
## survive   6     104  0.05454545
```

```
# train confusion matrix
icu.rf.pred = predict(icu.rf, newdata = icu.train)
confusionMatrix(as.factor(icu.rf.pred),
                as.factor(icu.train$STA))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction die survive
##    die      30       0
##    survive   0     110
##
##                Accuracy : 1
##                  95% CI : (0.974, 1)
##     No Information Rate : 0.7857
##     P-Value [Acc > NIR] : 2.173e-15
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.0000
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 1.0000
##              Prevalence : 0.2143
##          Detection Rate : 0.2143
##    Detection Prevalence : 0.2143
##       Balanced Accuracy : 1.0000
##
##        'Positive' Class : die
##
```

```
# test confusion matrix
icu.rf.pred = predict(icu.rf, newdata = icu.test)
confusionMatrix(as.factor(icu.rf.pred),
                as.factor(icu.test$STA))
```

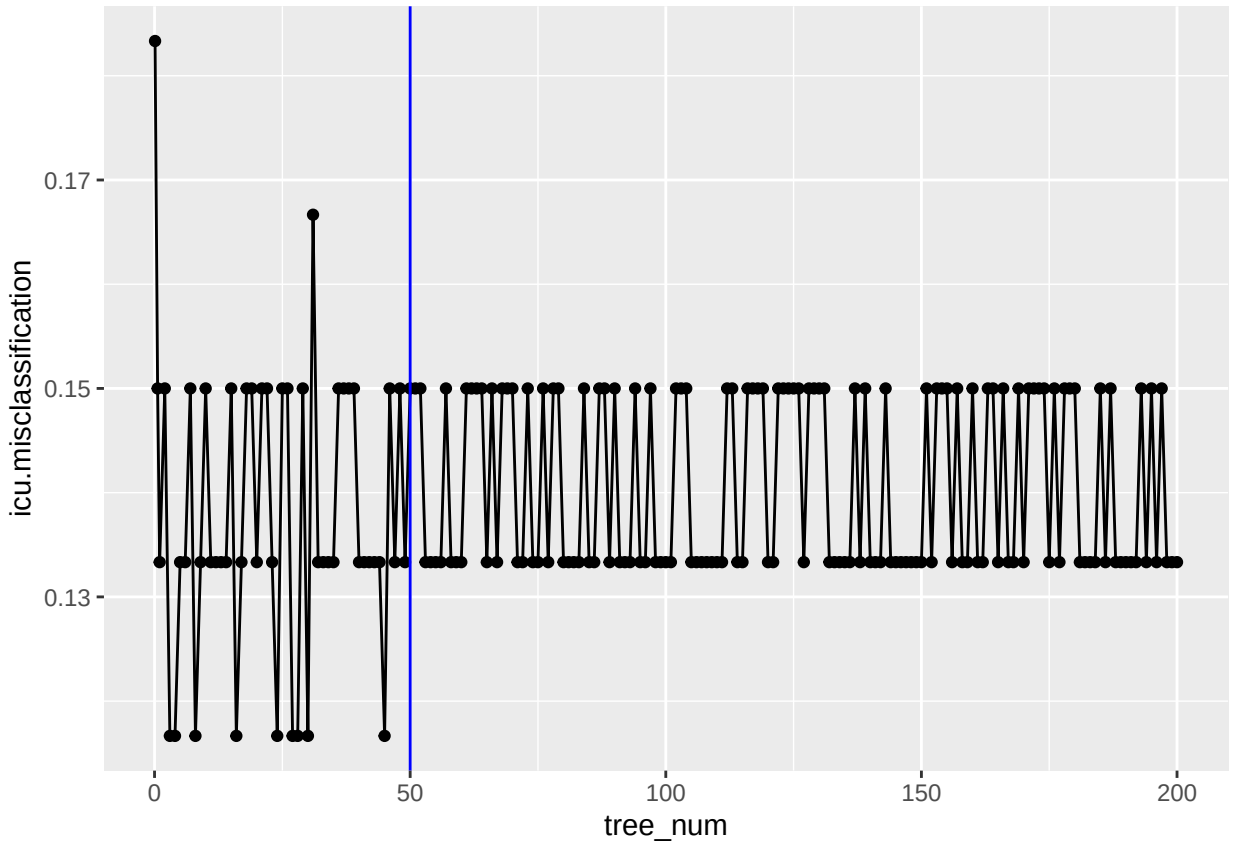
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction die survive
##    die       4       2
##    survive   6      48
##
##                Accuracy : 0.8667
##                  95% CI : (0.7541, 0.9406)
##     No Information Rate : 0.8333
##     P-Value [Acc > NIR] : 0.3120
##
##                   Kappa : 0.4286
##
##  Mcnemar's Test P-Value : 0.2888
##
##             Sensitivity : 0.40000
##             Specificity : 0.96000
##          Pos Pred Value : 0.66667
##          Neg Pred Value : 0.88889
##              Prevalence : 0.16667
##          Detection Rate : 0.06667
##    Detection Prevalence : 0.10000
##       Balanced Accuracy : 0.68000
##
##        'Positive' Class : die
##
```

Part D

```
k <- 1
icu.misclassification <- c()
tree_num <- c(seq(0.1, 0.9, 0.5), 1:200)
for(i in tree_num) {
  icu_train1 <- randomForest(STA~.,
                             data = icu.train,
                             mtry = 6,
                             importance = TRUE,
                             ntree = i * 10)
  pred <- predict(icu_train1, newdata = icu.test)
  icu.misclassification[k] <- mean((pred) != (icu.test$STA))
  k <- k + 1
}
which.min(icu.misclassification)*10
```

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

```
qplot(tree_num,icu.misclassification) + geom_line() + geom_vline(xintercept = which.min(icu.misclassifi
```



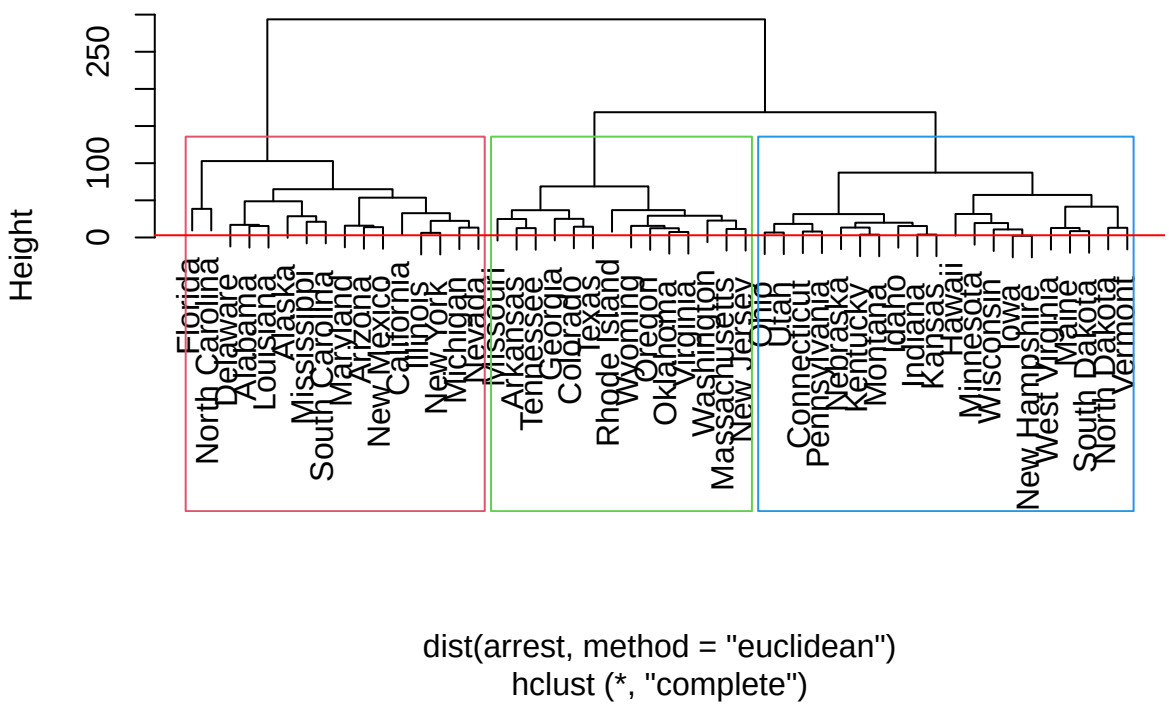
Problem 4

Part A

```
arrest = read.csv("USArrest.csv")
sum(is.na(arrest))
```

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

```
rownames(arrest) = arrest[, 1]
arrest = arrest[, -1]
arrest.hclust = hclust(dist(arrest, method = "euclidean"),
                       method = "complete")
plot(arrest.hclust)
rect.hclust(arrest.hclust , k = 3, border = 2:6)
abline(h = 3, col = 'red')
```



Part B

```
arrest.hclust3 = cutree(arrest.hclust, k = 3)
table(arrest.hclust3)
```

```
## arrest.hclust3
##  1  2  3
## 16 14 20
```

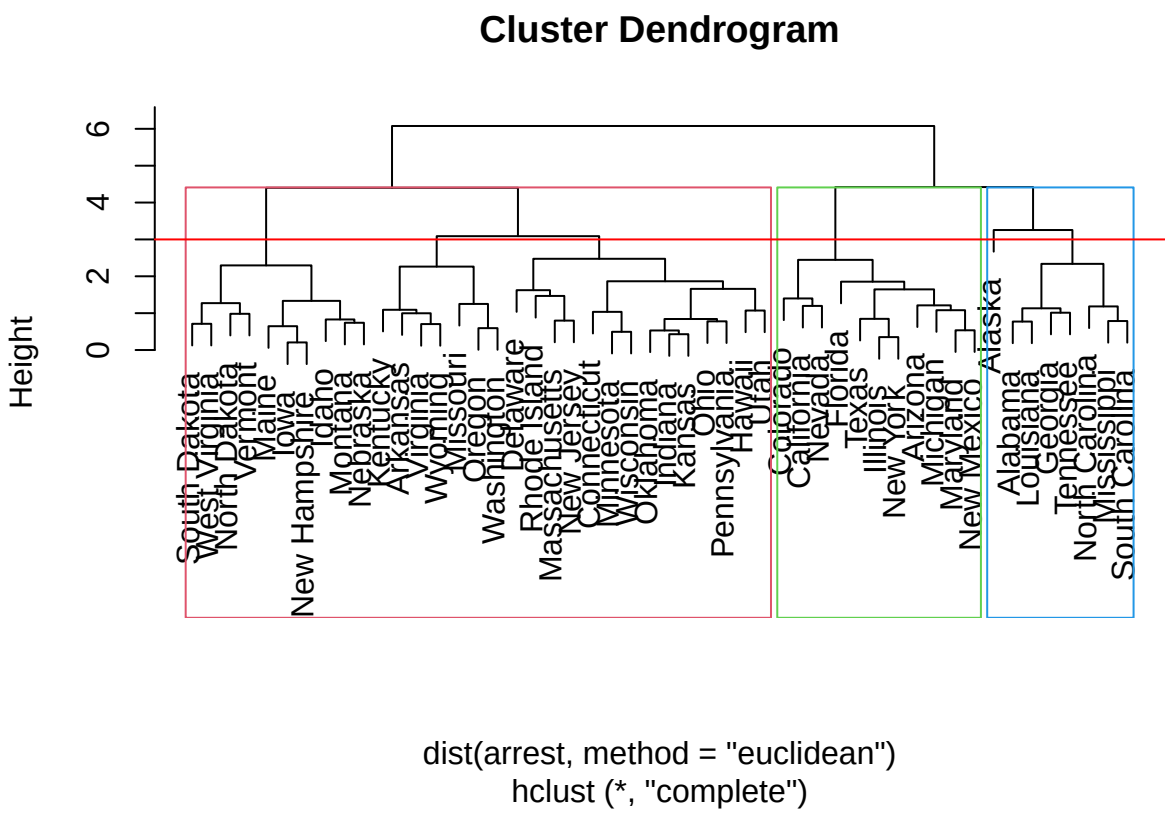
```
arrest.hclust3
```

```
##        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
```

The above shows which state belongs to which cluster. For example, Alabama belongs to cluster 1, Arkansas belongs to cluster 2, and Conneticut belongs to cluster 3. In total, there are 16 states in cluster 1, 14 in cluster 2, and 20 in cluster 3. ### Part C

```
arrest = scale(arrest)
arrest.hclust.scaled = hclust(dist(arrest, method = "euclidean"),
                              method = "complete")
plot(arrest.hclust.scaled)
rect.hclust(arrest.hclust.scaled , k = 3, border = 2:6)
abline(h = 3, col = 'red')
```




```
arrest.hclust3.scaled = cutree(arrest.hclust.scaled, k = 3)
table(arrest.hclust3.scaled)
```

```
## arrest.hclust3.scaled
##  1  2  3
##  8 11 31
```

```
arrest.hclust3.scaled
```

```
##        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
```

After scaling the variables, we see that there are 8 states in cluster 1, 11 states in cluster 2, and 31 states in cluster 3.

Part D

Scaling has greatly affected the clusters. In a way, it has made them more accurate. We should always look to scale when doing hierarchical clustering, since it is dependent on distance (in our case, we decided on euclidean distance). With a scaled data set, we now have a more accurate representation of clusters.

Furthermore, by scaling the predictors, we now have a shorter tree. This is important to note, since it allows us to say that the tree is a simpler model.

Problem 5

Part A

```
olives = read.csv("Olives.csv")
olives = olives[, -c(3, 1)]
olives$region = as.factor(olives$region)
olives[, 2:9] = scale(olives[, 2:9])
```

Part B

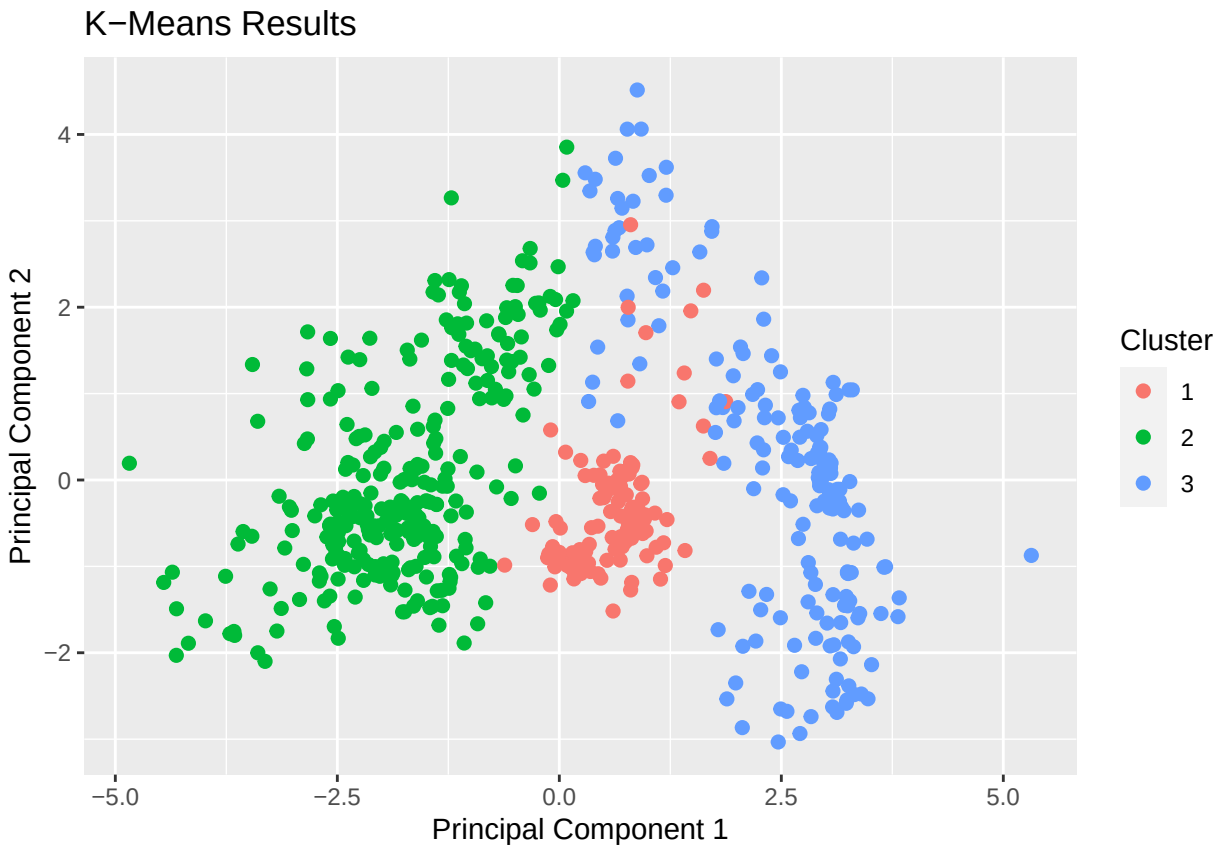
```
olives.kmeans = kmeans(olives, 3)
olives.hclust = hclust(dist(olives, method = "euclidean"),
                       method = "average")
olives.hclust3 = cutree(olives.hclust, k = 3)
```

Part C

```
summary(olives.kmeans)
```

```
##              Length Class  Mode
## cluster      572    -none- numeric
## centers       27    -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
```

```
library(useful)
plot(olives.kmeans, data = olives)
```




```
table(olives.kmeans$cluster)
```

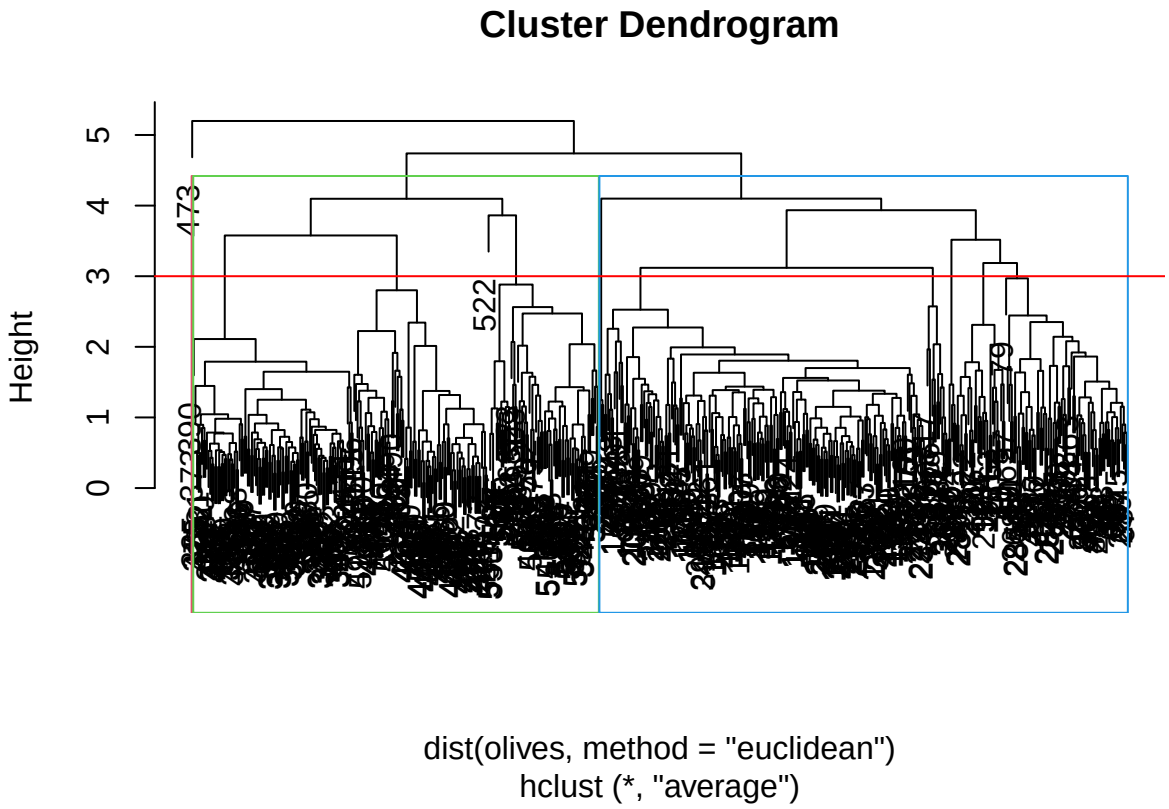
```
##
##   1   2   3
## 109 284 179
```

Above is the kmeans() plot as well as the table that tells us how many data points belong to each of the 3 clusters. Notice how it is not extremely easy determining the difference between the three clusters (since they are so close to each other in the center).

```
summary(olives.hclust3)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.000   1.000   1.437   2.000   3.000
```

```
plot(olives.hclust)
rect.hclust(olives.hclust , k = 3, border = 2:6)
abline(h = 3, col = 'red')
```




```
table(olives.hclust3)
```

```
## olives.hclust3
##   1   2   3
## 323 248   1
```

Above is the tree plot for the olives data set as well as the table that tell us how many data points belong to each of the 3 clusters.

Part D

Notice that the two methods have wildly different data points in each cluster.

```
table(Cluster = as.factor(olives.kmeans$cluster),
      Region = as.factor(olives$region))
```

```
##         Region
## Cluster   1   2   3
##       1   0  98  11
##       2 284   0   0
##       3  39   0 140
```

Looking at the table, we can infer that cluster 1 correctly classified 98 points as region 2. Cluster 2 correctly classified 248 points as region 1. Cluster 3 correctly classified 140 points as region 3.

Cluster 1 had an accuracy rate of 0.873065.

Cluster 2 had an accuracy rate of 1.

Cluster 3 had an accuracy rate of 0.9271523.

```
table(Cluster = as.factor(olives.hclust3),
      Region = as.factor(olives$region))
```

```
##         Region
## Cluster   1   2   3
##       1 323   0   0
##       2   0  98 150
##       3   0   0   1
```

Looking at this table, we can infer that cluster 1 correctly classified 323 points as region 1. Cluster 2 correctly classified 150 points as region 3. Cluster 3 correctly classified 0 points as region 2.

Cluster 1 had an accuracy rate of 1.

Cluster 2 had an accuracy rate of 0.9933775.

Cluster 3 had an accuracy rate of 0.

It seems on average, hierarchical clustering did better than kmeans.

Part E

```
library(pls)
```

```
##
## Attaching package: 'pls'

## The following object is masked from 'package:caret':
##
##     R2

## The following object is masked from 'package:stats':
##
##     loadings
```

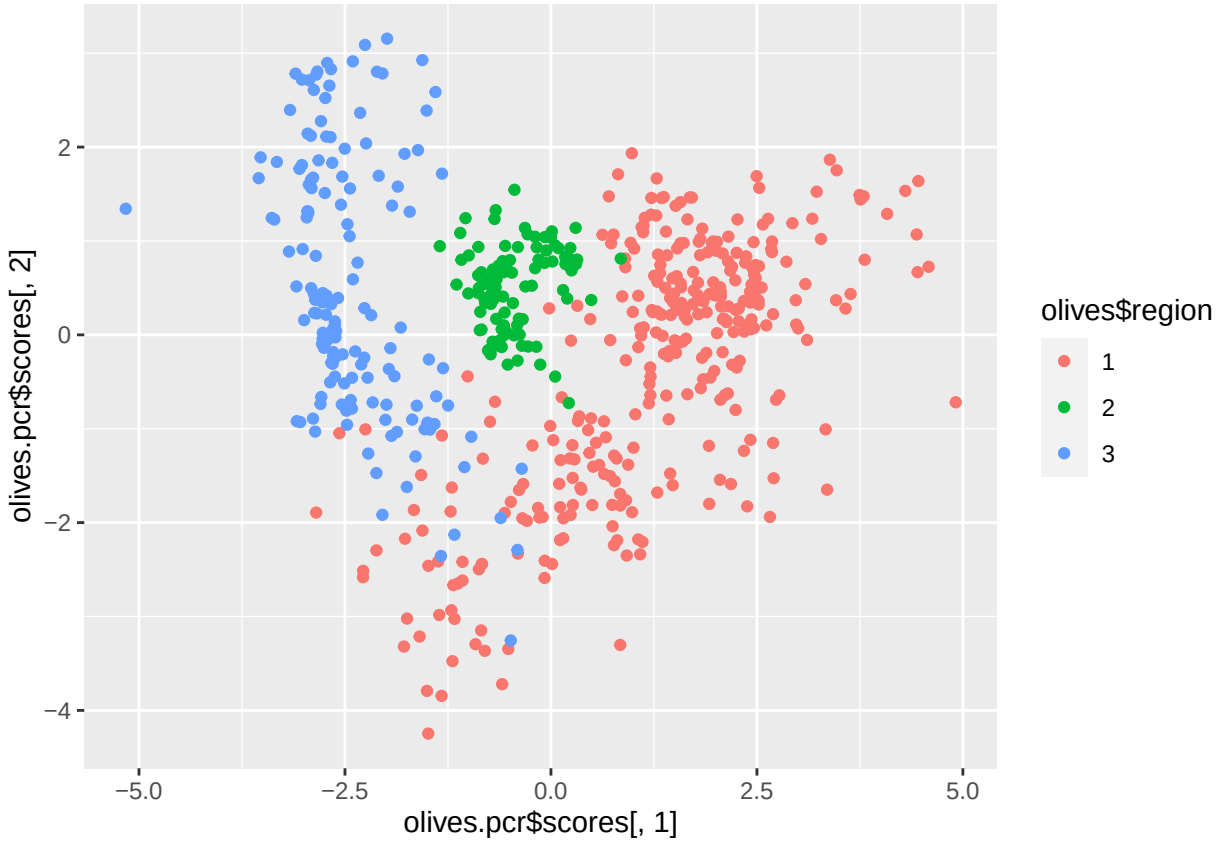
```
olives.pcr = pcr(as.numeric(region) ~ ., data = olives)
summary(olives.pcr)
```

```
## Data:    X dimension: 572 8
##  Y dimension: 572 1
## Fit method: svdpc
## Number of components considered: 8
## TRAINING: % variance explained
##                      1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## X                      46.52    68.59    81.29    91.21    95.38    98.49
## as.numeric(region)     66.23    75.74    77.15    77.56    86.70    86.70
##                      7 comps  8 comps
## X                      99.97   100.00
## as.numeric(region)     86.73    86.75
```

The amount of variation explained from the first component is 46.52%. The amount of variation explained from the second component is 68.59%.

Part F

```
qplot(olives.pcr$scores[,1], olives.pcr$scores[,2], color = olives$region)
```

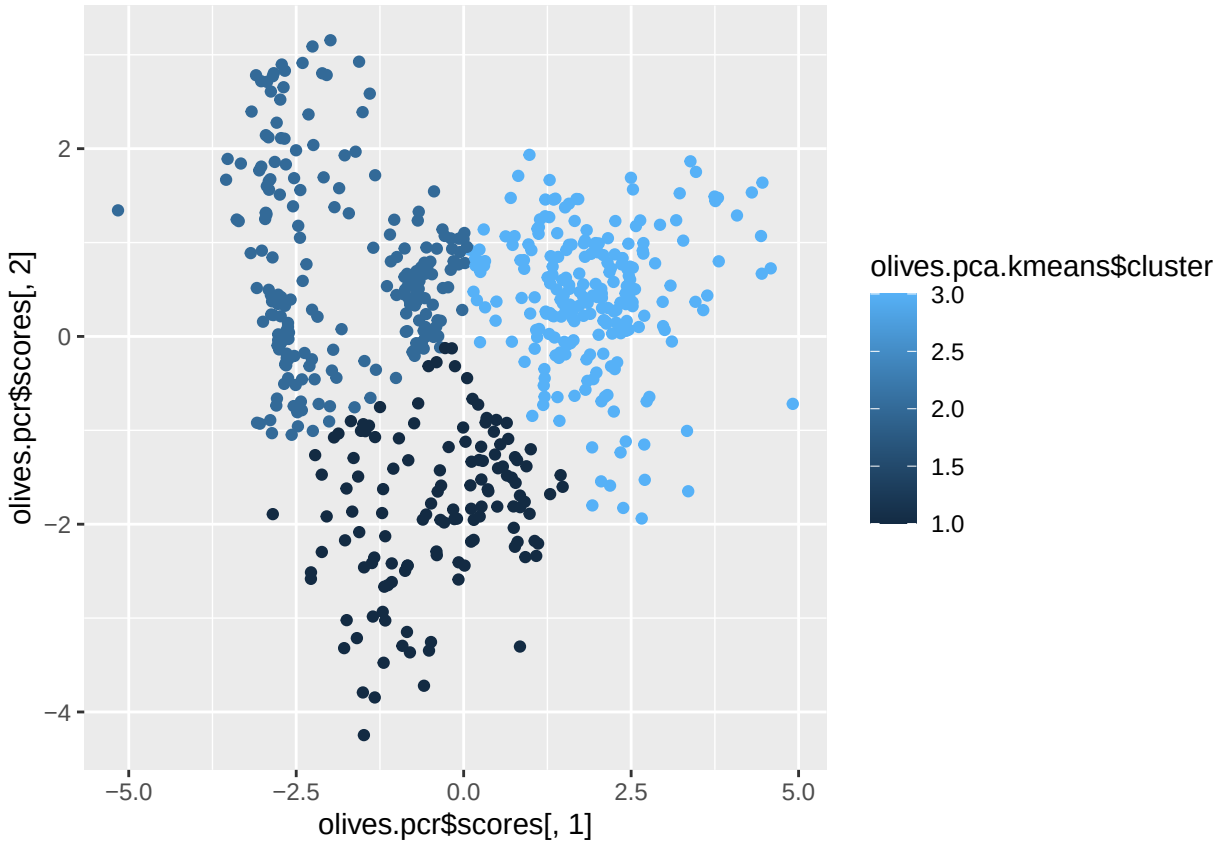


Above is the plot where the PCA components are plotted against each other. Notice how difficult it is to plot region 2 from 3.

```
olives.pca.kmeans = kmeans(cbind(olives.pcr$scores[,1], olives.pcr$scores[,2]), 3)
table(Cluster = as.factor(olives.pca.kmeans$cluster),
      Region = olives$region)
```

```
##        Region
## Cluster   1   2   3
##       1 103   7  21
##       2   4  78 130
##       3 216  13   0
```

```
qplot(olives.pcr$scores[,1], olives.pcr$scores[,2], color = olives.pca.kmeans$cluster)
```



Notice how it seems to be a lot easier, now that we have used kmeans, to cluster the points, since points seem to be LESS cluttered. However, both seem to do pretty poorly. This is probably because the regions are already quite cluttered already, and it is difficult for the unsupervised learning process to clutter them correctly.