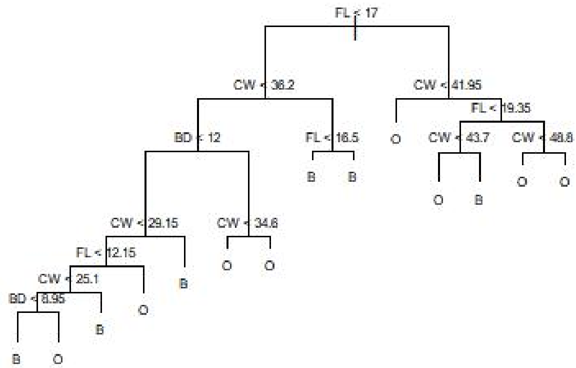
Stat415homework9

(a) Set the random seed to 45678 and randomly select 80% of the data as your training data.

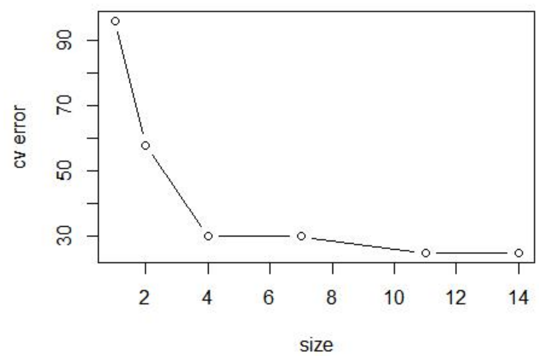
```
set.seed(45678)
library(MASS)
data(crabs)
# clarify the combination index
crabsbf = which(crabs$sp=="B" & crabs$sex=="F" )
crabsbm = which(crabs$sp=="B" & crabs$sex=="M" )
crabsof = which(crabs$sp=="O" & crabs$sex=="F" )
crabsom = which(crabs$sp=="O" & crabs$sex=="M" )
train = c(sample(crabsbf, size =trunc(0.80 *length(crabsbf))),sample(cr
absbm, size =trunc(0.80 *length(crabsbm))),sample(crabsof, size =trunc
(0.80 *length(crabsof))),sample(crabsom, size =trunc(0.80 *length(crabs
om))))
train_data = crabs[train,]
test_data = crabs[-train,]
summary(train_data)
```

(b) Train a classification tree to predict Species from the five numerical measurements and sex, selecting the optimal size by cross- validation but using no more than 8 splits.

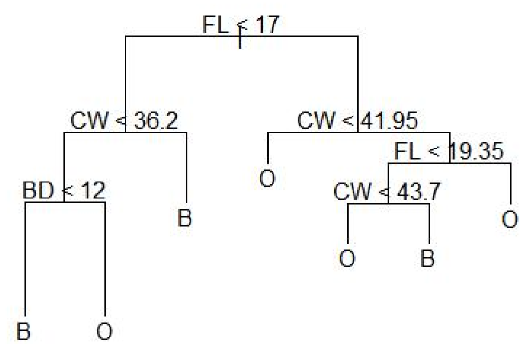
```
library(tree)

tree.crabs = tree(sp~.-index,train_data)
tree.pred = predict(tree.crabs,test_data,type="class")
tree.train = predict(tree.crabs,train_data,type="class")
summary(tree.crabs)
## Classification tree:
## tree(formula = sp ~ . - index, data = train_data)
## Variables actually used in tree construction:
## [1] "FL" "CW" "BD"
## Number of terminal nodes:  14
## Residual mean deviance:  0.2333 = 34.06 / 146
## Misclassification error rate: 0.04375 = 7 / 160

# calculate the test error
calc_class_err = function(actual, predicted){
  mean(actual != predicted)
}
calc_class_err(predicted = tree.pred, actual = test_data$sp)

## [1] 0.15

# plot the tree
plot(tree.crabs)
text(tree.crabs,pretty=0,cex=0.5)
```




```r
# improve the tree by CV
set.seed(45678)
cv.crabs = cv.tree(tree.crabs,FUN = prune.misclass)
cv.crabs

## $size
## [1] 14 11  7  4  2  1
##
## $dev
## [1] 25 25 30 30 58 96
##
## $k
## [1]      -Inf  0.000000  2.500000  2.666667 11.000000 33.000000
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"

plot(cv.crabs$size,cv.crabs$dev,ylab="cv error", xlab="size",type="b")
```




```
prune.crabs=prune.misclass(tree.crabs,best=7)
plot(prune.crabs)
text(prune.crabs,pretty=0)
```

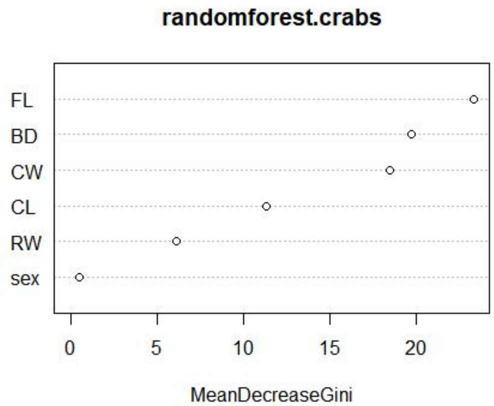



```
tree.pred=predict(prune.crabs,test_data,type="class")
table(tree.pred,test_data$sp)
## tree.pred  B  O
##         B 14  0
##         O  6 20

6/40

## [1] 0.15
```

Comment: according to the output, the training error rate is 0.04375 and test error is 0.15. After using CV method, we choose the size to be 7 and corresponding dev is 30, which is the smallest with no more than 8 splits. Simplified tree has been plotted and the corresponding test error is 0.15.Variables used are “FL”,“CW” and “BD”.

(c) Now train random forests on the data, using three randomly selected predictors at each split, and 1000 trees total.

```r
library(randomForest)

set.seed(45678)
randomforest.crabs = randomForest(sp~.-index,data=crabs,subset=train,mt
ry=3,ntree=1000)
randomforest.crabs
## Call:
##  randomForest(formula = sp ~ . - index, data = crabs, mtry = 3,
 ntree = 1000, subset = train)
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 13.75%
## Confusion matrix:
##    B  O class.error
## B 67 13      0.1625
## O  9 71      0.1125

# Training error
(9+13)/160

## [1] 0.1375

# Test error
randomforest.pred = predict(randomforest.crabs,newdata=test_data)
calc_class_err(predicted = randomforest.pred, actual = test_data$sp)

## [1] 0.05

# variable importance plot
importance(randomforest.crabs)

##     MeanDecreaseGini
## sex        0.4794565
## FL        23.3221954
## RW         6.1198232
## CL        11.3307614
## CW        18.5043081
## BD        19.7512054

varImpPlot(randomforest.crabs)
```

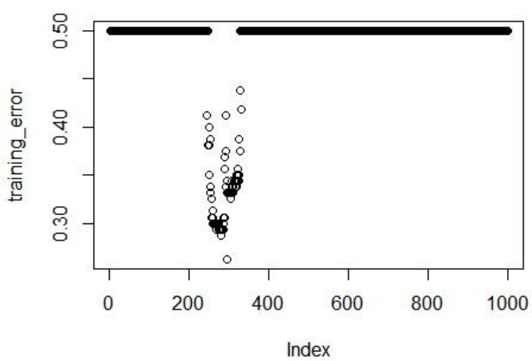


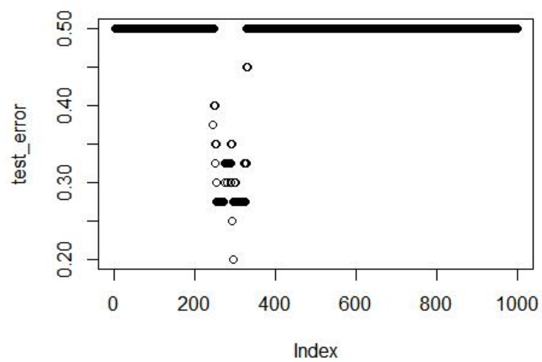
Comment: according to the output, the training error is 0.1375 and the test error is 0.05. The test error indicates that Random Forest method performs better than a single tree. And FL & BD are by far the two most important variables.

(d) Finally, train AdaBoost on the data. Plot the training and test errors as a function of the number of trees constructed by boosting up to 1000. Compute training and test errors.

```
library(gbm)

set.seed(45678)
levels(train_data$sp) = c(0,1)
levels(test_data$sp) = c(0,1)
levels(crabs$sp) = c(0,1)
# calculate the error
test_error = rep(0,1000)
training_error = rep(0,1000)
for (i in 1:1000){
  boost.crabs = gbm(sp~.-index,data=train_data,distribution="adaboost",
n.trees=i)
  # training
  adatrain.pred = predict.gbm(boost.crabs,newdata=train_data,n.trees=i)
  train_class = rep(0,160)
  for(j in 1:160){
    train_class[j] = ifelse(adatrain.pred[j] >= 0.5, 1, 0)
  }
  # train error
  training_error[i] = calc_class_err(predicted = train_class, actual =
train_data$sp)
  # test
  adatest.pred = predict.gbm(boost.crabs,newdata=test_data,n.trees=i)
  adatest.pred_class = rep(0,length(adatest.pred))
  for(j in 1:length(adatest.pred_class)){
    adatest.pred_class[j] = ifelse(adatest.pred[j] >= 0.5, 1, 0)
  }
  # test error
  test_error[i] = calc_class_err(predicted = adatest.pred_class, actual
 = test_data$sp)
```

```
}
plot(test_error)

plot(training_error)
```






```
training_error[which(training_error == min(training_error))]

## [1] 0.2625

test_error[which(test_error == min(test_error))]

## [1] 0.2

which(training_error == min(training_error))

## [1] 294

which(test_error == min(test_error))

## [1] 294
```

Comment: According to the output, the test error is 0.2 and the smallest training error is 0.2625. The output shows that both training error and test error decrease first then turn to be high again. Since Boost method may cause over-fitting, M can not be too large. In this data set, when n.trees=294, both training error and test error are the smallest.

(e) Comment on which method appears to perform best for this data set and how consistent the results are across methods.

From the above output, for a single tree, the training error is 0.04375 and the test error is 0.15. For the Random Forest, the training error is 0.1375 and the test error is 0.05. For boosting method, the minimum training and test errors are 0.2625 and 0.2.Thus according to the test error, Random Forest method has best performance.