Lab 7 Bagging and Random Forest

1) survived is a numeric value. We need to first transform it to a categorical value and saved it as a new variable survived01. Use `titanic3$survived01 = as.factor(titanic3$survived)` to do so and check that this variable has been included in the dataset.

```
library(dplyr)

titanic3 <- read.csv("C:\\Users\\tingting\\Dropbox\\modules\\BigData\\sessions\\labs\\Lab7 RF\\titanic3
titanic3 <- select(titanic3,-name, -ticket, -boat, -body, -home.dest, -cabin) %>%
  mutate(embarked = factor(embarked),
         sex = factor(sex),
         pclass = factor(pclass),
         survived01 = as.factor(titanic3$survived))
#embed survived01=as.factor(titanic3$survived) in the mutate() function
#to simplify the code. Use help to see more about mutate().
#Here, it is same to use survived01 = factor(titanic3$survived)

summary(titanic3)
```

```
##   pclass       survived         sex           age
##  1   :323   Min.   :0.000         :  1   Min.   : 0.1667
##  2   :277   1st Qu.:0.000   female:466   1st Qu.:21.0000
##  3   :709   Median :0.000   male  :843   Median :28.0000
##  NA's:  1   Mean   :0.382                Mean   :29.8811
##             3rd Qu.:1.000                3rd Qu.:39.0000
##             Max.   :1.000                Max.   :80.0000
##             NA's   :1                    NA's   :264
##      sibsp            parch            fare          embarked survived01
##  Min.   :0.0000   Min.   :0.000   Min.   :  0.000    :  3    0   :809
##  1st Qu.:0.0000   1st Qu.:0.000   1st Qu.:  7.896   C:270    1   :500
##  Median :0.0000   Median :0.000   Median : 14.454   Q:123    NA's:  1
##  Mean   :0.4989   Mean   :0.385   Mean   : 33.295   S:914
##  3rd Qu.:1.0000   3rd Qu.:0.000   3rd Qu.: 31.275
##  Max.   :8.0000   Max.   :9.000   Max.   :512.329
##  NA's   :1        NA's   :1       NA's   :2
```

2) Install the package of randomForest and include this package into your code. In order to call the randomForest() function, all the missing value rows need to be dealt with. The simplest way is to remove those rows. Use titanic3 <- na.omit(titanic3) to do that.

```
library(randomForest)
nrow(titanic3)
```

```
## [1] 1310
```

```
titanic3 <- na.omit(titanic3)
nrow(titanic3)
```

```
## [1] 1045
```

3) Use a seed to set half of the dataset to be training dataset and the other half to be test dataset.

```r
set.seed(8)
train <- sample(1:nrow(titanic3), nrow(titanic3)/2)
test <- titanic3[-train,]  #the test set
x_test <- test[,-c(2,9)]  #the predictors in the test set
#-c(2,9) is to remove survived and survived01
# names(titanic3)
#[1] "pclass"     "survived"   "sex"        "age"        "sibsp"
#[6] "parch"      "fare"       "embarked"   "survived01"

survived.test <- titanic3$survived[-train]
survived01.test <- titanic3$survived01[-train]
```

4) Use the training dataset to build a bagged model for

- y: survived
- x: all the features other than survived and survived01.

Compute the mean error rate on the test dataset.

Remark: You might get a warning message, saying that

In randomForest.default(m, y, ...) : The response has five or fewer unique values. Are you sure you want to do regression?

Ignore the message for now. It's doable and you will get a bagged model anyway.

```r
#set.seed(6)
bag.titanic <- randomForest(survived ~ .-survived01,
                            data = titanic3,
                            subset = train,
                            mtry=7,
                            importance = TRUE)
print(bag.titanic)
```

```
##
## Call:
##  randomForest(formula = survived ~ . - survived01, data = titanic3,      mtry = 7, importance = TRUE
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##           Mean of squared residuals: 0.157021
##                     % Var explained: 34.99
```

```r
bag.pred <- predict(bag.titanic, newdata = test)
bag.pred.class <- ifelse(bag.pred <= 0.5, "0", "1")
print(mean(bag.pred.class!=survived.test))
```

```
## [1] 0.2141491
```

5) Using the same training and test dataset, build a bagged model for

- y: survived01
- x: all the features other than survived and survived01

a) Find out on how many trees your model is built and the OOB error
b) Compute the mean error rate on the test dataset.

```
bag.titanic01 <- randomForest(survived01 ~ .-survived,
                              data = titanic3,
                              subset = train,
                              mtry = 7,
                              importance = TRUE)
print(bag.titanic01)
```

```
##
## Call:
##  randomForest(formula = survived01 ~ . - survived, data = titanic3,      mtry = 7, importance = TRUE
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 22.99%
## Confusion matrix:
##     0   1 class.error
## 0 252  57   0.1844660
## 1  63 150   0.2957746
```

```
bag.pred01 <- predict(bag.titanic01, newdata = test, type="class")
print(mean(bag.pred01!=survived.test))
```

```
## [1] 0.2198853
```

```
bag.titanic01
```

```
##
## Call:
##  randomForest(formula = survived01 ~ . - survived, data = titanic3,      mtry = 7, importance = TRUE
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 22.99%
## Confusion matrix:
##     0   1 class.error
## 0 252  57   0.1844660
## 1  63 150   0.2957746
```

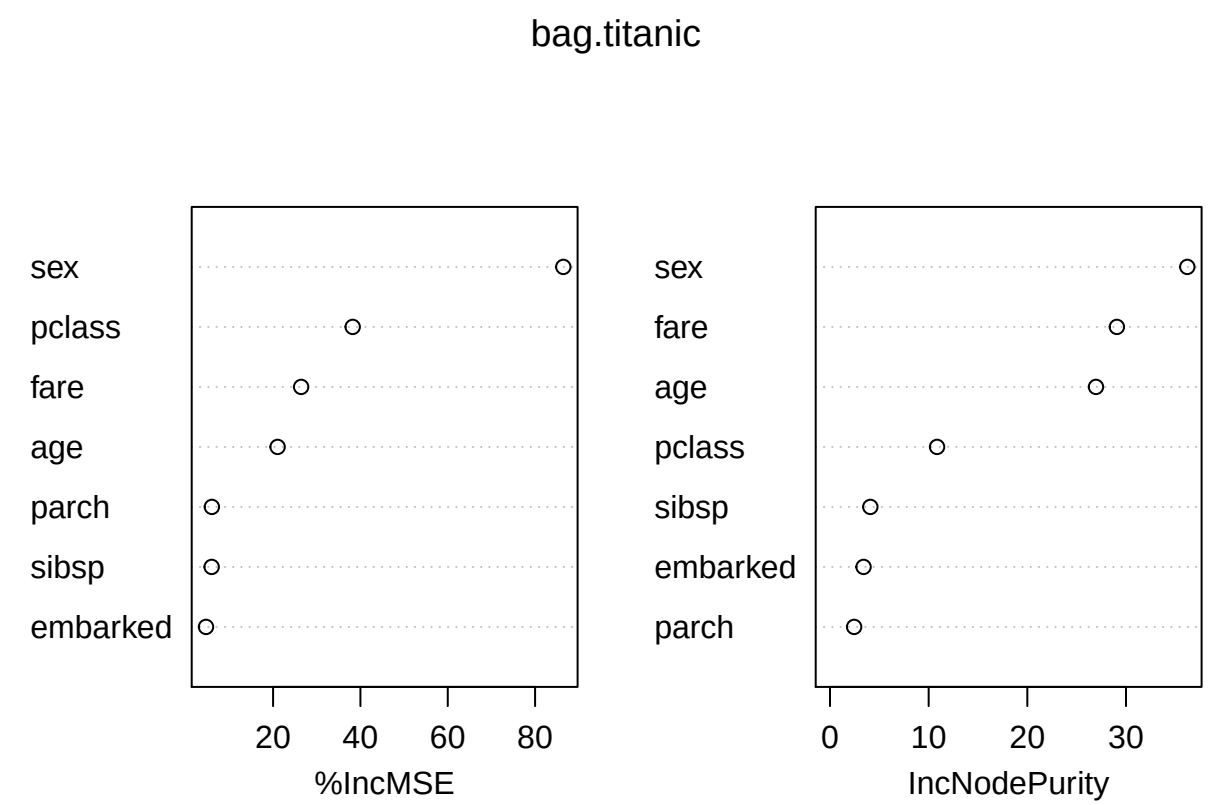
The OOB error is 23.18% and the number of trees grown is 500. Note that OOB error rate is only available for classification trees.

6) Plot the variable importance plot for the two bagged models you built in 4) and 5) and comment whether the importance coincides.

```
importance(bag.titanic)
```

```
##            %IncMSE IncNodePurity
## pclass   38.229329     10.849789
## sex      86.458487     36.213718
## age      21.035222     26.957278
## sibsp     5.935803      4.086833
## parch     5.980129      2.445480
## fare     26.427893     29.075916
## embarked  4.639885      3.399932
```

```
varImpPlot(bag.titanic)
```

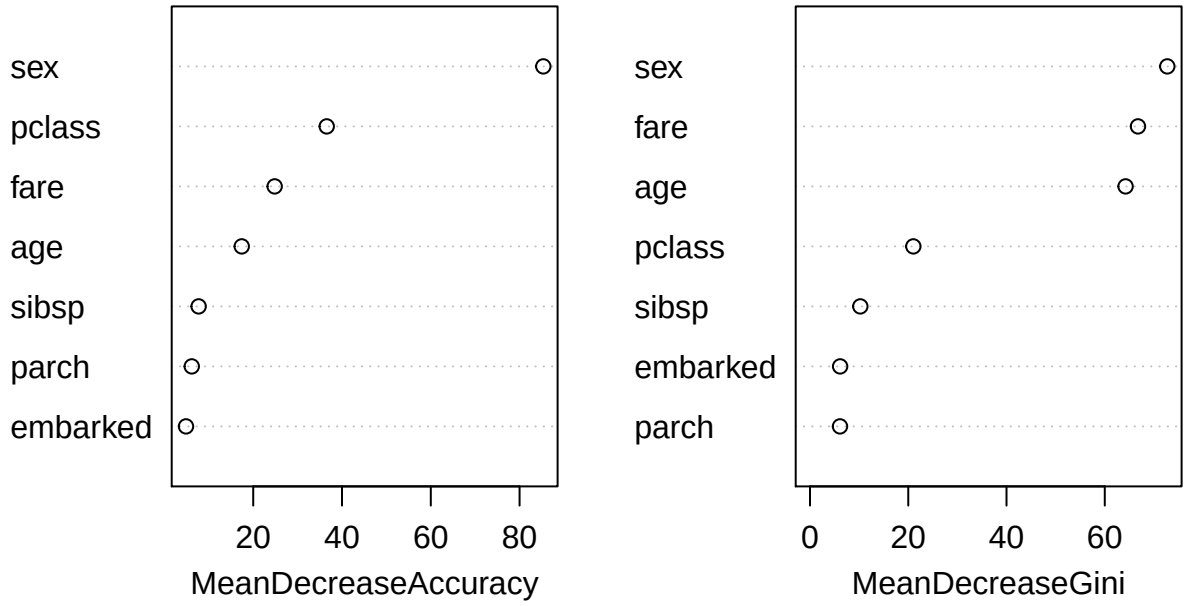



```
importance(bag.titanic01)
```

```
##                 0         1 MeanDecreaseAccuracy MeanDecreaseGini
## pclass   11.399214 37.737466            36.553497        21.030430
## sex      53.103426 76.214604            85.343731        72.729657
## age       8.835855 17.323586            17.412895        64.234899
## sibsp     8.869927 -1.126593             7.706300        10.239392
## parch     4.270089  4.041387             6.158348         6.105879
## fare      1.914595 34.483210            24.830863        66.760682
## embarked  7.354944 -1.949526             4.858996         6.133310
```

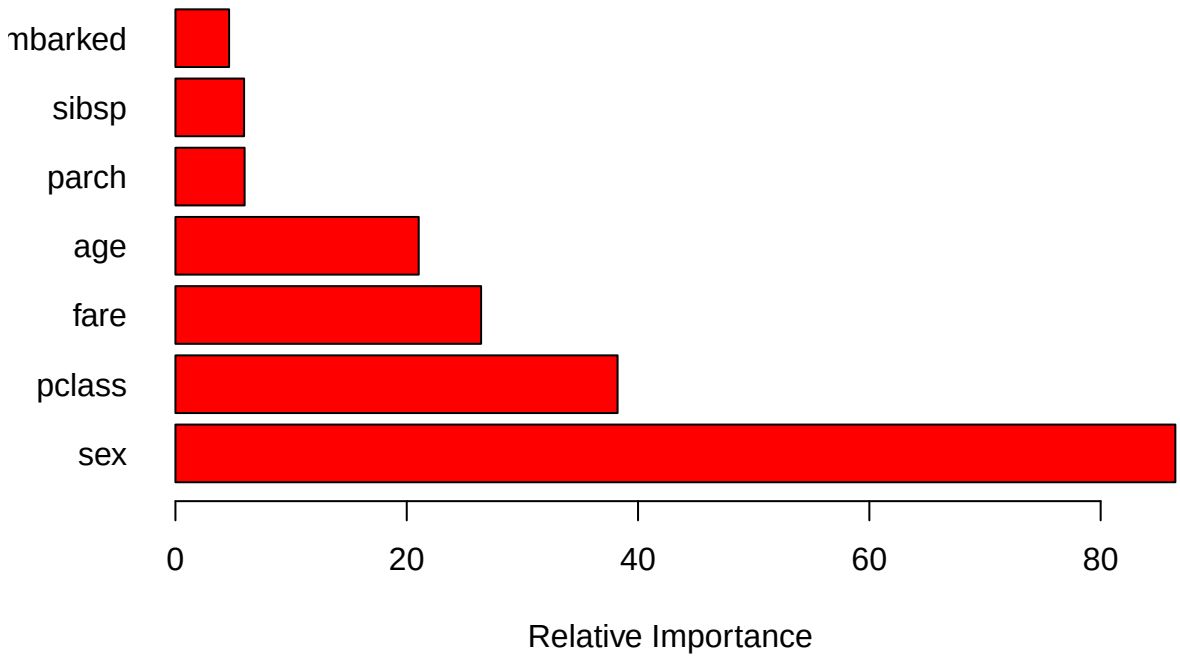
```
varImpPlot(bag.titanic01)
```

bag.titanic01




```r
#The barplot can be:

barplot(sort(importance(bag.titanic)[,1], decreasing = TRUE),
        xlab = "Relative Importance",
        horiz = TRUE,
        col = "red",
        las=1 #The las argument will allow rotation of 90 degrees for labels
        )
```



Yes, the importance of both models coincide if we look at the %IncMSE.

7) Plot a graph that shows the test error rate of a single tree (red dashed line), the mean test error rates for majority vote (black curve) and the test error rates for averaging the probabilities (blue curve), both in relation to the number of trees. Add a legend if you can.

%or ->

```
# calculate the black line:
#Here we insert the test set (xtest and ytest) when building the model.
#Please read the last three slides of Session 7 for more information.
bag.titanic01=randomForest(survived01 ~ .-survived,
                           data=titanic3,
                           subset = train,
                           importance = TRUE,
                           xtest=x_test,
                           ytest=survived01.test,
                           mtry=7,
                           ntree=200)


#plot the black line
plot(1:200, bag.titanic01$test$err.rate[,1],
     type="l",
     xlab="Number of Bootstrap Data Sets",
     ylab="Test Error Rate",
```

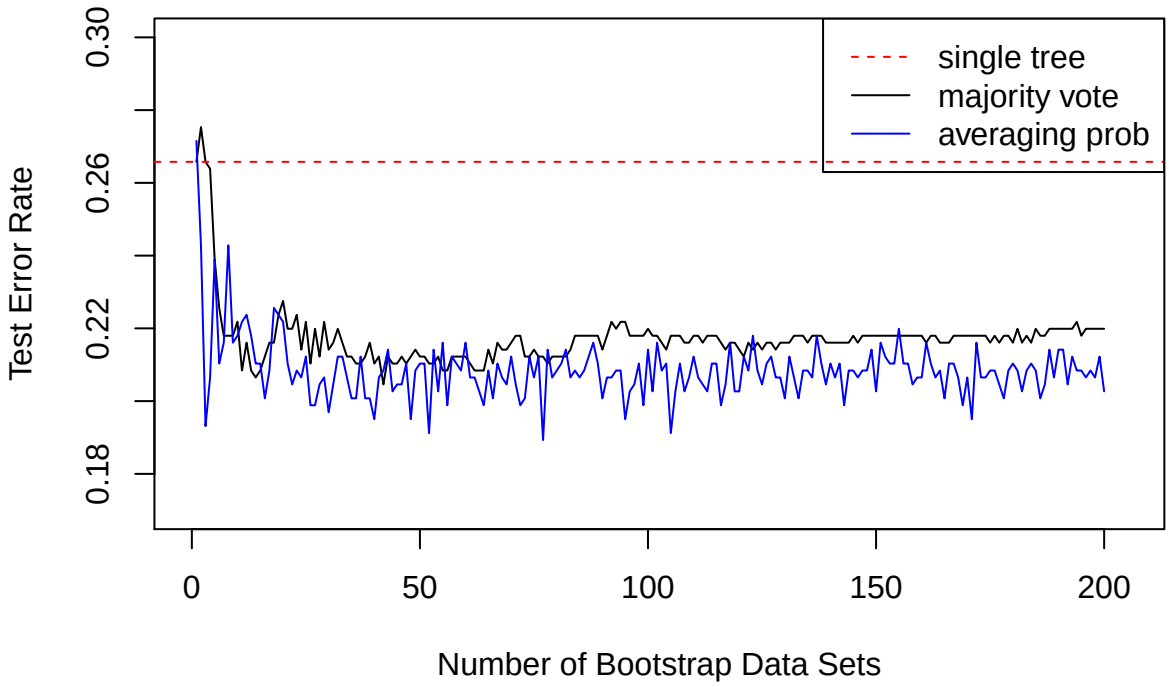
```
     ylim=c(0.17,0.30), xlim=c(0,205))

#plot the red dashed line
abline(h = bag.titanic01$test$err.rate[1,1],
       lty=2,col="red")


yhat.ter.ave <- rep(0,200)     # a vector for Test Error Rate using averaging
for(j in 1:200){
  #set.seed(6)
  bag.titanic <- randomForest(survived ~ .-survived01,
                              data=titanic3,
                              subset = train,
                              mtry=7,
                              importance = TRUE,
                              ntree=j)
  bag.pred <- predict(bag.titanic, newdata = test)
  bag.pred.class <- ifelse(bag.pred<=0.5, "0", "1")
  yhat.ter.ave[j] <- mean(bag.pred.class!=survived.test)
}

lines(yhat.ter.ave,col="blue")

legend("topright",
       c("single tree","majority vote","averaging prob"),
       lty=c(2,1,1),
       col=c("red","black","blue"))
```

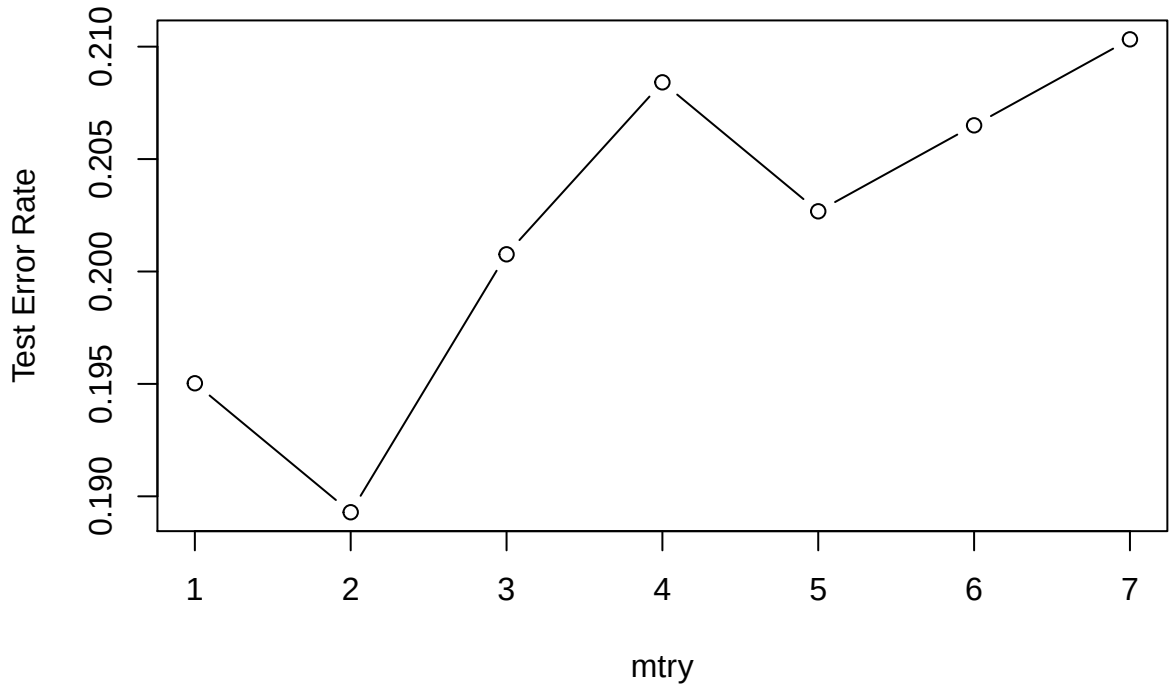


8) Plot a graph that shows the best value of mtry for the random forest model

- y: survived01
- x: all the features other than survived and survived01
- mtry: range from 1 to 7

%–>

```
testErrorRate <- rep(0,7)
for(i in 1:7){
  set.seed(6)
  bag.titanic01 <- randomForest(survived01 ~ .-survived,
                                data=titanic3,
                                subset=train,
                                mtry=i,
                                importance=TRUE,
                                xtest=x_test,
                                ytest=survived01.test,
                                ntree=500)
  testErrorRate[i] <- bag.titanic01$test$err.rate[500,1]
}
plot(testErrorRate,type="b",xlab="mtry",ylab="Test Error Rate")
```

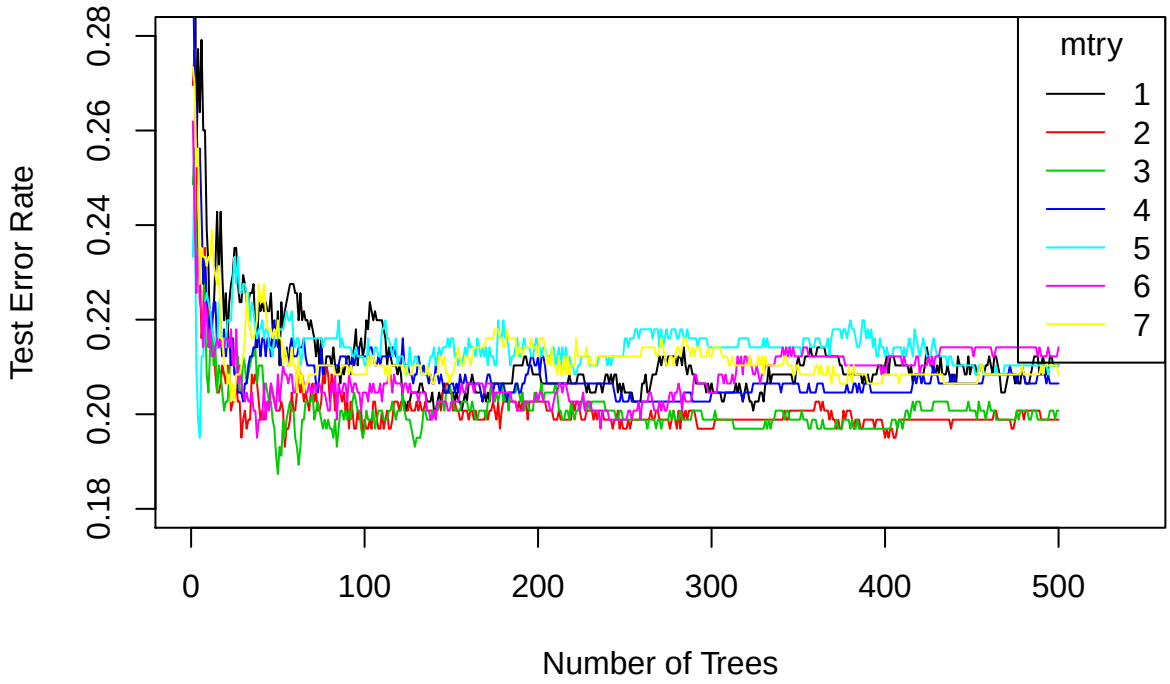


9) Play with mtry and ntree, plot a graph that shows test error rate vs ntree for different mtry, and find the best/reasonably good combination of mtry and ntree from the plot. Add a legend if you can.

```
plot(0,
     xlab="Number of Trees",ylab="Test Error Rate",
     xlim=c(1,540), ylim =c(0.18,0.28))


for(i in 1:7){
  # It's also possible to call randomForest using x and y as the training set.
  # bag.titanic01=randomForest(x=x_train, y=y_train01,
  #                            importance = TRUE,
  #                            xtest=x_test, ytest=survived01.test,
  #                            mtry=7,ntree=500)
  bag.titanic01=randomForest(survived01 ~ .-survived,
                             data=titanic3,
                             subset=train,
                             importance = TRUE,
                             xtest=x_test,
                             ytest=survived01.test,
                             mtry=i,
                             ntree=500)
  lines(bag.titanic01$test$err.rate[,1],col=i,type="l")
}
```

```
legend(title = "mtry",
       "topright",
       c("1","2","3","4","5","6","7"),
       lty=rep(1,7),col=1:7)
```



From the plot, we can see that mtry $= 2$ and 3 are better than the others in the long run. This result coincides with the empirical result: pick `mtry = sqrt(p)` when it is a classification tree. Here `p = 7` and `sqrt(p) = sqrt(7) = 2.65` $\in [2, 3]$.