Homework 4

Kejia Hu

1. Prepare the data

We will use the airline delay data again. Before we start, we will take a random sample of the observations. Take a sample of n = 100;000 from each year of the data. This will give us 22 years of data with 100;000 observations in each for 2:2 million observations.
The code to download, decompress and random draw observations are attached here:

```
for (x in 1987:2008){
system(sprintf("curl -s http://eeyore.ucdavis.edu/stat242/data/%s.csv.bz2 -o - |
bunzip2 |shuf -n 100000 | cut -d, -f 1-19 > %s.csv",    x,x))}
```

To merge all random files together in a big file:

```
for (x in 1987:2008){
system(sprintf("cat %s.csv >> airline.csv",    x))}
```

We try the sapply in Parallel Version with package snow

```
library(snow)
#Create a cluster of 10 R slave processes:
cl <- makeCluster(10)
doso=function(x){ system(sprintf("curl -s
http://eeyore.ucdavis.edu/stat242/data/%s.csv.bz2 -o - | bunzip2 |shuf -n
100000 | cut -d, -f 1-19 > %s.csv",    x,x))
}
parSapply(cl, 1987:2008, doso)
```

2.Random Forest

The randomForest package grow separate trees. In order to make it faster to fit a given number of trees, we will use the randomForest() function in parallel to fit all of the desired trees and combine the results.
The generate tree and combine them together using the following code:

```
rf1 <- randomForest(Delay ~ ., airline, ntree=50)
rf2 <- randomForest(Delay ~ ., airline, ntree=50)
rf3 <- randomForest(Delay ~ ., airline, ntree=50)
rf.all <- combine(rf1, rf2, rf3)
```

In R, there are several packages available for parallel computing, here we use three approaches to illustrate the usage of some of them.
Approach 1:

```
library('foreach')
library('doSMP')
library('randomForest')
NbrOfCores <- 8
workers <- startWorkers(NbrOfCores) # number of cores
registerDoSMP(workers)
getDoParName() # check name of parallel backend
getDoParVersion() # check version of parallel backend
getDoParWorkers() # check number of workers
set.seed(1)
ntree=1000
ntree2 <- ntree/NbrOfCores
#running serialized version of random forests
rf1 <- randomForest(Delay ~ ., airline, ntree = ntree)
#running parallel version of random forests
rf2 <- foreach(ntree = rep(ntree2, 8), .combine = combine, .packages =
"randomForest") %dopar% randomForest(Delay ~ ., airline, ntree = ntree)
```

Approach 2:

```
library(doMC)
registerDoMC()
#running parallel version of random forests
rf <- foreach(ntree = rep(ntree2, 8), .combine = combine, .packages =
"randomForest") %dopar% randomForest(Delay ~ ., airline, ntree = ntree)
#running serialized version of random forests
rf2 <- foreach(ntree = rep(ntree2, 8), .combine=combine) %do%
randomForest(Delay ~ ., airline, ntree = ntree)
```

Approach 3:

```
library("foreach")
library("doSNOW")
registerDoSNOW(makeCluster(4, type="SOCK"))
#running parallel version of random forests
rf <- foreach(ntree = rep(ntree2, 8), .combine = combine, .packages =
"randomForest") %dopar% randomForest(Delay ~ ., airline, ntree = ntree)
```

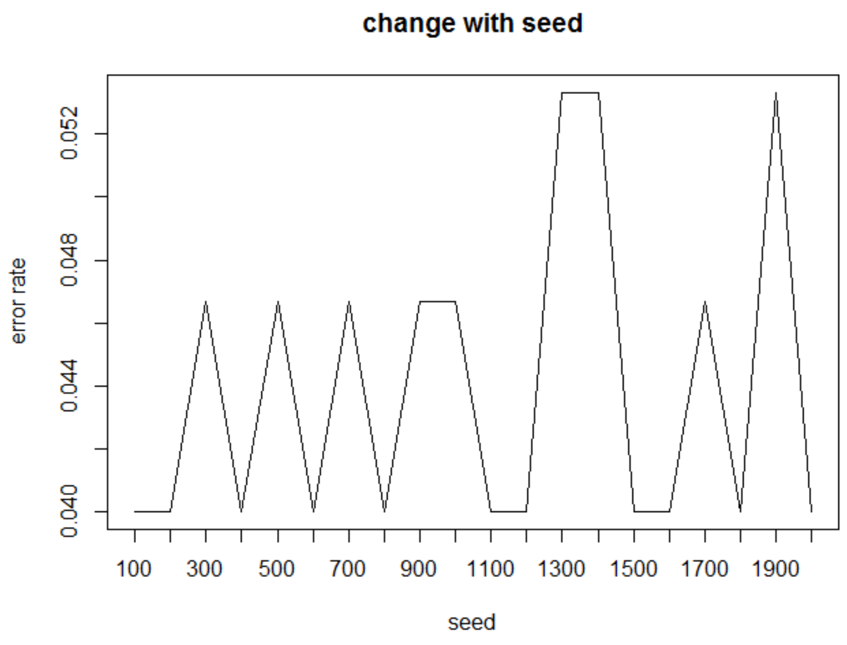
In parallel execution, the tree generation is pretty quick, but the rest of the time is consumed in combining the results (combine option). So, the worth to run parallel execution for the number of trees is really high.

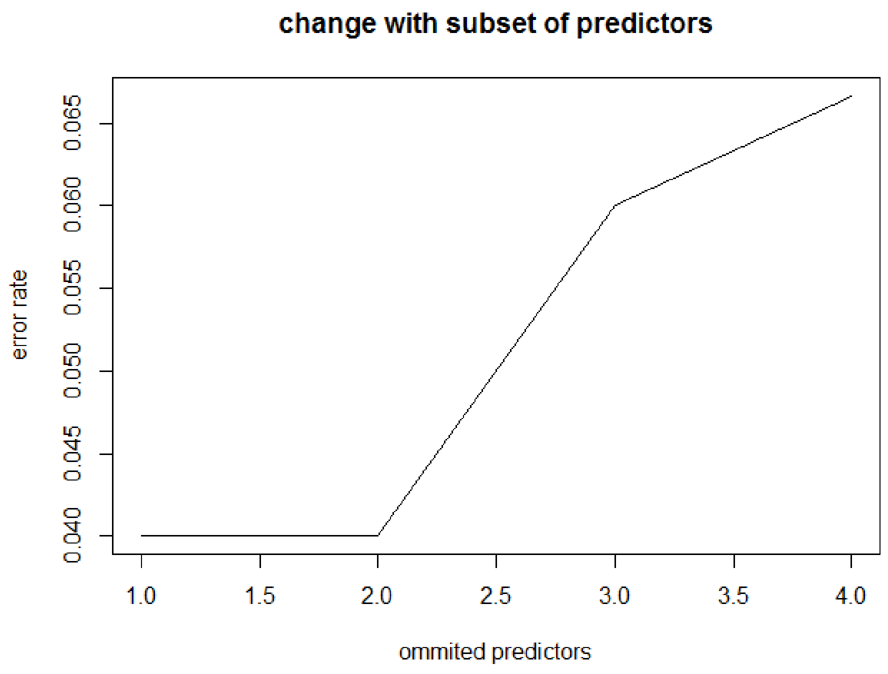
A prediction for random forests object and generate the votes is

```
Rf.all$votes
```

Bagging just aggregates the fits across different bootstrap samples. Random forests selects random subsets of the predictor variables at each node in each tree. However, what will happen if we change the seed of randomization of the variables and what will happen if we adopt different subsets of predictors for random forest. The performance is evaluated based on the total classification error rate.

```
length(which(rf$predicted!=data[,index]))/dim(data)[1]
```


change with seed
error rate
seed


change with subset of predictors
error rate
ommited predictors