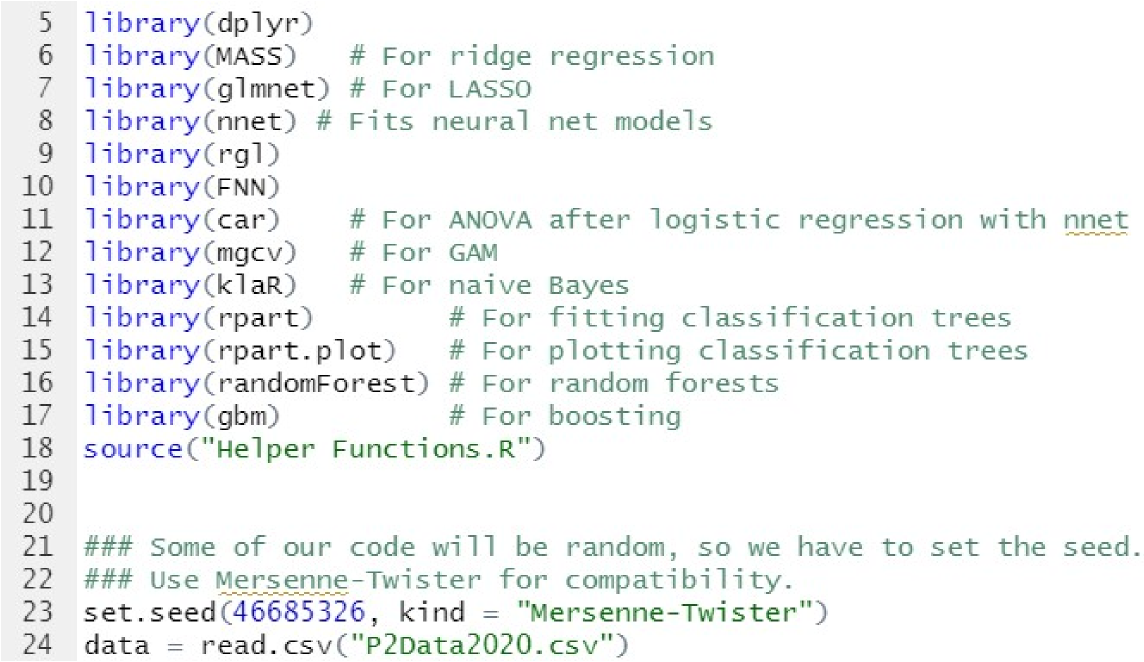
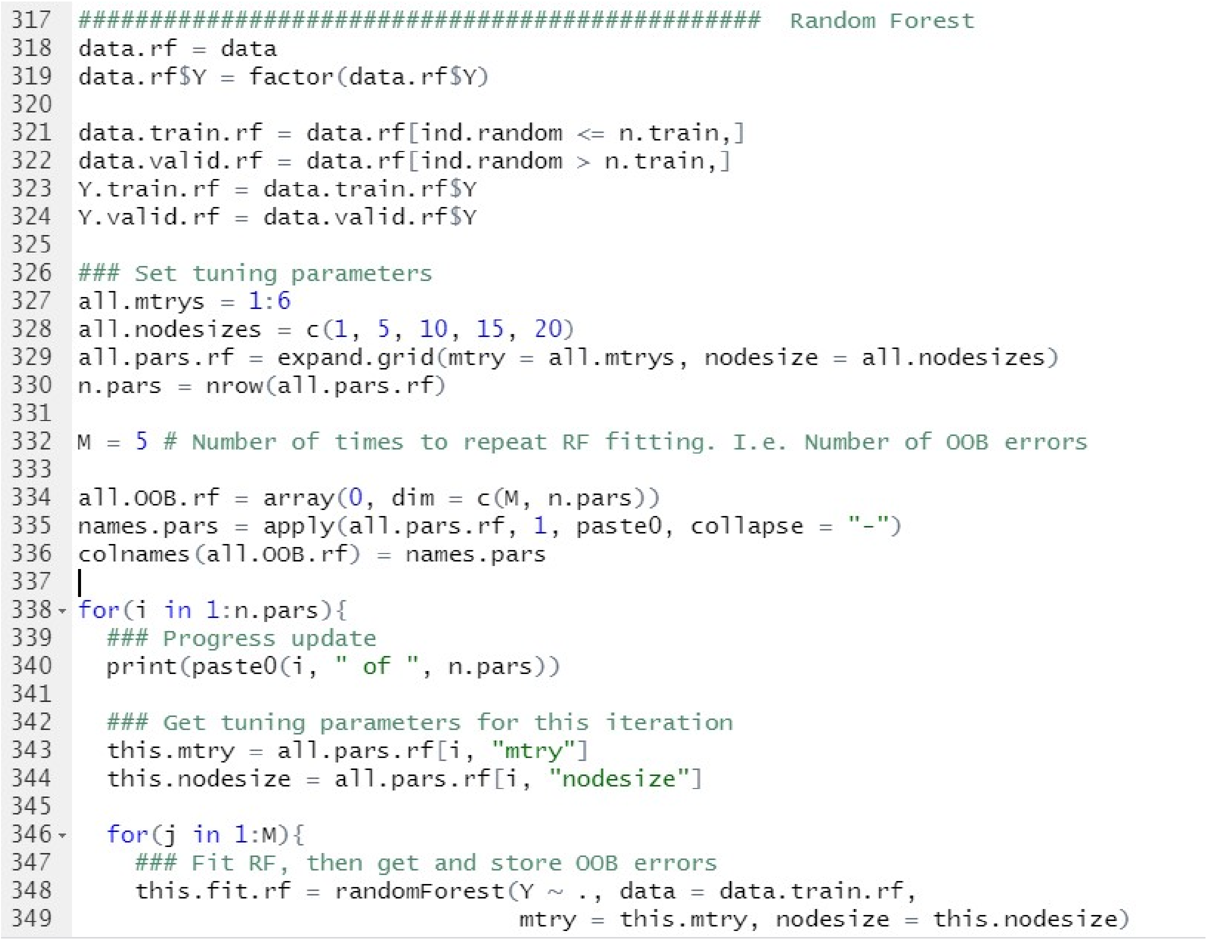
5. *What was your chosen prediction machine?* **Paste the code that produced your predicted values, including all values of tuning parameters if any, random number seeds, and explaining any variable names that are not obvious.** I should be able to run your code and produce the same results (or extremely similar if randomization is used). If I try and it doesn't work, there will be a major deduction.

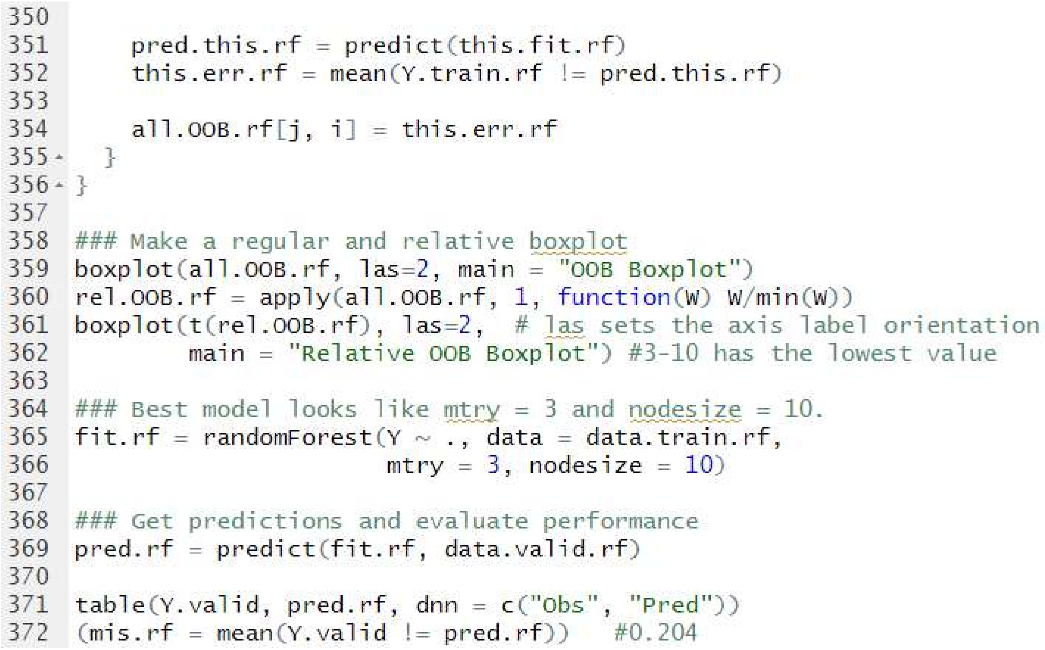
- ➔ I choose randomforest as my prediction model.



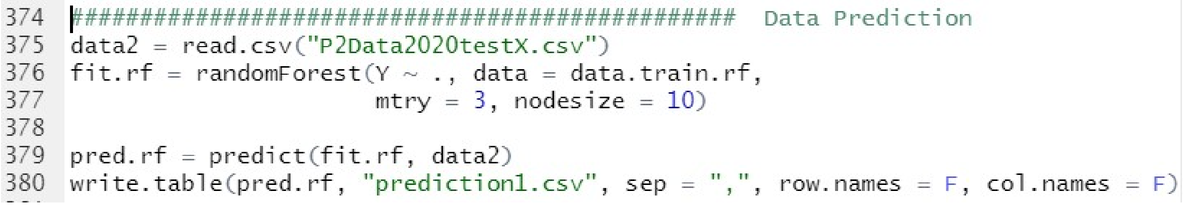
```r
library(dplyr)
library(MASS)   # For ridge regression
library(glmnet) # For LASSO
library(nnet) # Fits neural net models
library(rgl)
library(FNN)
library(car)    # For ANOVA after logistic regression with nnet
library(mgcv)   # For GAM
library(klaR)   # For naive Bayes
library(rpart)        # For fitting classification trees
library(rpart.plot)   # For plotting classification trees
library(randomForest) # For random forests
library(gbm)          # For boosting
source("Helper Functions.R")


### Some of our code will be random, so we have to set the seed.
### Use Mersenne-Twister for compatibility.
set.seed(46685326, kind = "Mersenne-Twister")
data = read.csv("P2Data2020.csv")
```



```
##################################################  Random Forest
data.rf = data
data.rf$Y = factor(data.rf$Y)

data.train.rf = data.rf[ind.random <= n.train,]
data.valid.rf = data.rf[ind.random > n.train,]
Y.train.rf = data.train.rf$Y
Y.valid.rf = data.valid.rf$Y

### Set tuning parameters
all.mtrys = 1:6
all.nodesizes = c(1, 5, 10, 15, 20)
all.pars.rf = expand.grid(mtry = all.mtrys, nodesize = all.nodesizes)
n.pars = nrow(all.pars.rf)

M = 5 # Number of times to repeat RF fitting. I.e. Number of OOB errors

all.OOB.rf = array(0, dim = c(M, n.pars))
names.pars = apply(all.pars.rf, 1, paste0, collapse = "-")
colnames(all.OOB.rf) = names.pars

for(i in 1:n.pars){
  ### Progress update
  print(paste0(i, " of ", n.pars))

  ### Get tuning parameters for this iteration
  this.mtry = all.pars.rf[i, "mtry"]
  this.nodesize = all.pars.rf[i, "nodesize"]

  for(j in 1:M){
    ### Fit RF, then get and store OOB errors
    this.fit.rf = randomForest(Y ~ ., data = data.train.rf,
                               mtry = this.mtry, nodesize = this.nodesize)
```



```

    pred.this.rf = predict(this.fit.rf)
    this.err.rf = mean(Y.train.rf != pred.this.rf)

    all.OOB.rf[j, i] = this.err.rf
  }
}

### Make a regular and relative boxplot
boxplot(all.OOB.rf, las=2, main = "OOB Boxplot")
rel.OOB.rf = apply(all.OOB.rf, 1, function(W) W/min(W))
boxplot(t(rel.OOB.rf), las=2,  # las sets the axis label orientation
        main = "Relative OOB Boxplot") #3-10 has the lowest value

### Best model looks like mtry = 3 and nodesize = 10.
fit.rf = randomForest(Y ~ ., data = data.train.rf,
                      mtry = 3, nodesize = 10)

### Get predictions and evaluate performance
pred.rf = predict(fit.rf, data.valid.rf)

table(Y.valid, pred.rf, dnn = c("Obs", "Pred"))
(mis.rf = mean(Y.valid != pred.rf))   #0.204
```



```r
################################################  Data Prediction
data2 = read.csv("P2Data2020testX.csv")
fit.rf = randomForest(Y ~ ., data = data.train.rf,
                      mtry = 3, nodesize = 10)

pred.rf = predict(fit.rf, data2)
write.table(pred.rf, "prediction1.csv", sep = ",", row.names = F, col.names = F)
```