Different Random Forest Packages in R

Thiyanga Talagala

14 November 2017

```
library(knitr)
opts_chunk$set(tidy = TRUE)
```

randomForest Package

Data pre-processing

Split iris data to Training data and testing data

```
ind <- sample(2, nrow(iris), replace = TRUE, prob = c(0.7, 0.3))
trainData <- iris[ind == 1, ]
testData <- iris[ind == 2, ]
```

1. Load randomForest

```
library(randomForest)
```

```
## randomForest 4.6-12
```

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

2. Generate Random Forest learning treee

```
iris_rf <- randomForest(Species ~ ., data = trainData, ntree = 100, proximity = TRUE)
table(predict(iris_rf), trainData$Species)
```

```
##
##              setosa versicolor virginica
##   setosa         38          0         0
##   versicolor      0         38         2
##   virginica       0          2        32
```

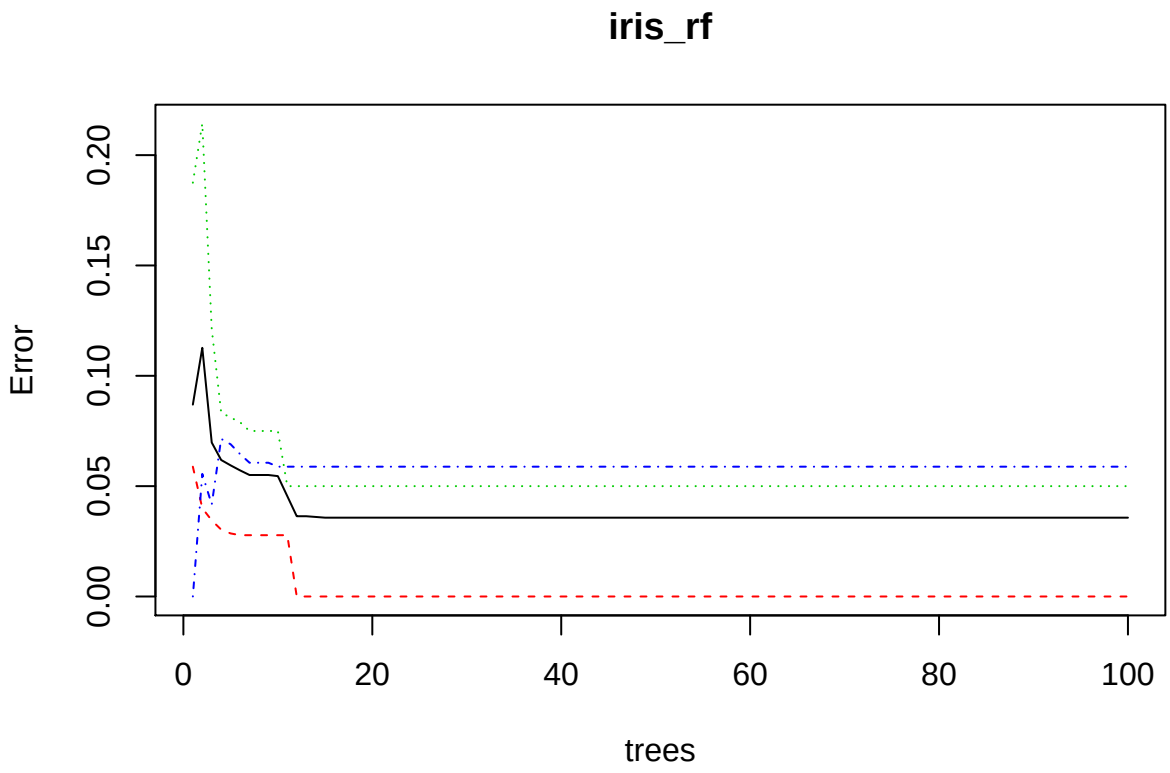
3. Try to print Random Forest model and see the importance features

```
print(iris_rf)
```

```
##
## Call:
##  randomForest(formula = Species ~ ., data = trainData, ntree = 100,      proximity = TRUE)
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 3.57%
## Confusion matrix:
##            setosa versicolor virginica class.error
## setosa         38          0         0  0.00000000
## versicolor      0         38         2  0.05000000
```

```
## virginica       0          2         32  0.05882353
```

```
plot(iris_rf)
```

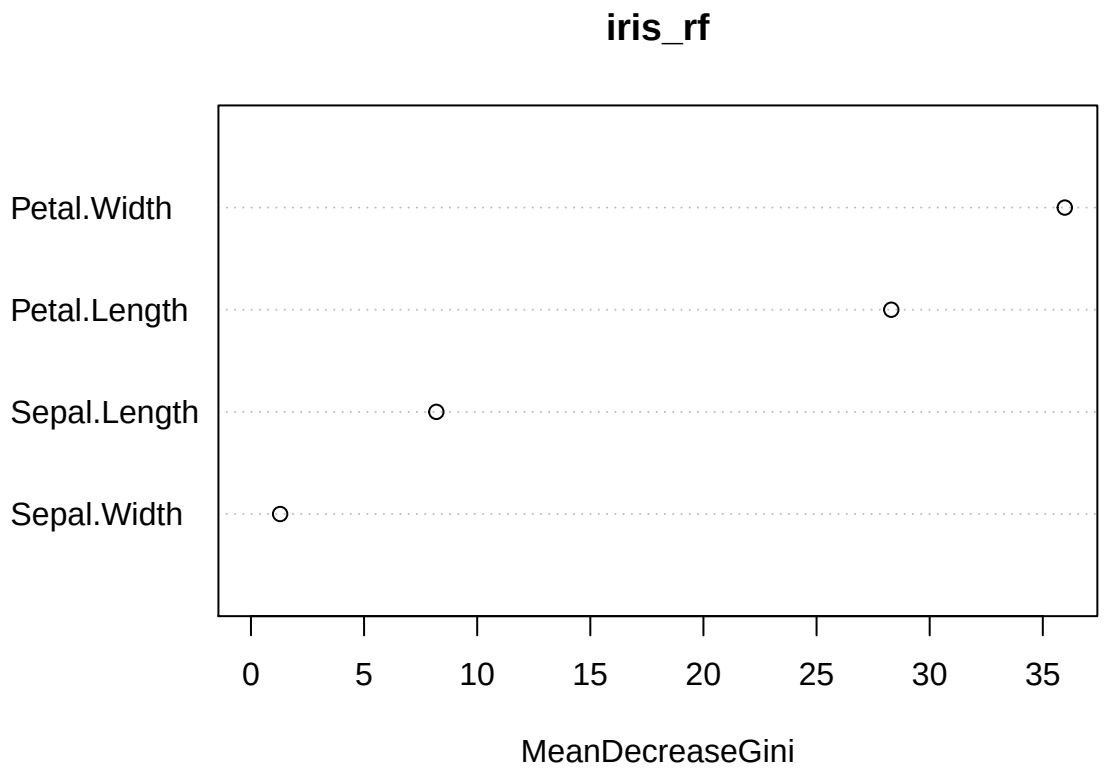


4. Assessing model fit

```
importance(iris_rf)
```

```
##              MeanDecreaseGini
## Sepal.Length         8.193935
## Sepal.Width          1.289160
## Petal.Length        28.302124
## Petal.Width         35.972579
```

```
varImpPlot(iris_rf)
```



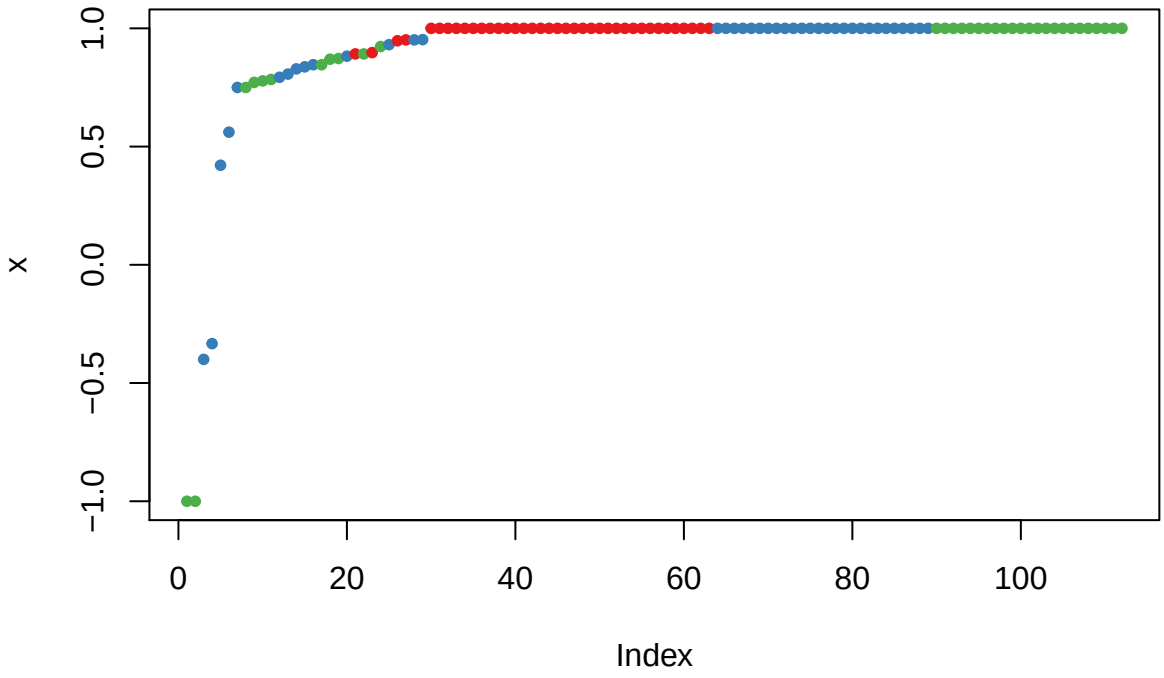
5. Predict the class labels for test data

```
irisPred <- predict(iris_rf, newdata = testData)
table(irisPred, testData$Species)
```

```
##
## irisPred     setosa versicolor virginica
##   setosa         12          0         0
##   versicolor      0          9         2
##   virginica       0          1        14
```

6. Try to see the margin, positive or negative, if positif it means correct classification

```
plot(margin(iris_rf, testData$Species))
```



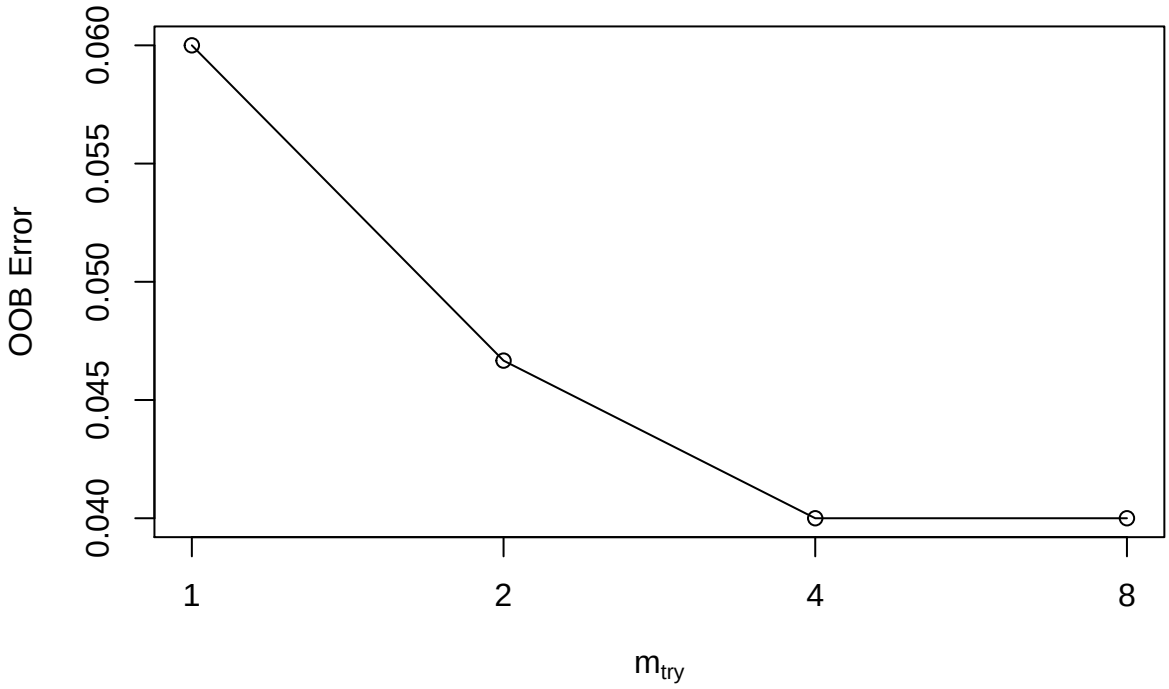
7. Tune randomForest for the optimal mtry parameter

```
tune.rf <- tuneRF(iris[, -5], iris[, 5], stepFactor = 0.5)
```

```
## mtry = 2  OOB error = 4.67%
## Searching left ...
## mtry = 4     OOB error = 4%
## 0.1428571 0.05
```

```
## Warning in randomForest.default(x, y, mtry = mtryCur, ntree = ntreeTry, :
## invalid mtry: reset to within valid range
```

```
## mtry = 8     OOB error = 4%
## 0 0.05
## Searching right ...
## mtry = 1     OOB error = 6%
## -0.5 0.05
```




```
print(tune.rf)
```

```
##       mtry   OOBError
## 1.OOB    1 0.06000000
## 2.OOB    2 0.04666667
## 4.OOB    4 0.04000000
## 8.OOB    8 0.04000000
```

The randomForestSRC Package

```
library(randomForestSRC)
```

```
##
##  randomForestSRC 2.5.1
##
##  Type rfsrc.news() to see new features, changes, and bug fixes.
##
```

1. Fitting a random forest

```
## Edgar Anderson's iris data iris.obj <- rfsrc(Species ~., data = trainData)
```

Parallel execution with random forest

```
library(randomForest)
library(doParallel)
```

```
## Loading required package: foreach
```

```
## Loading required package: iterators
```

```
## Loading required package: parallel
```

```
workers <- detectCores()
workers
```

```
## [1] 8
```

```
cl <- makePSOCKcluster(workers)
registerDoParallel(cl)

x <- matrix(runif(500), 100)
y <- gl(2, 50)
ntree <- 1000

rf <- foreach(n = rep(ceiling(ntree/workers), workers), .combine = combine,
    .multicombine = TRUE, .packages = "randomForest") %dopar% {
    randomForest(x, y, ntree = n)
}
```