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         39          0         0
##   versicolor      0         32         2
##   virginica       0          2        35
```

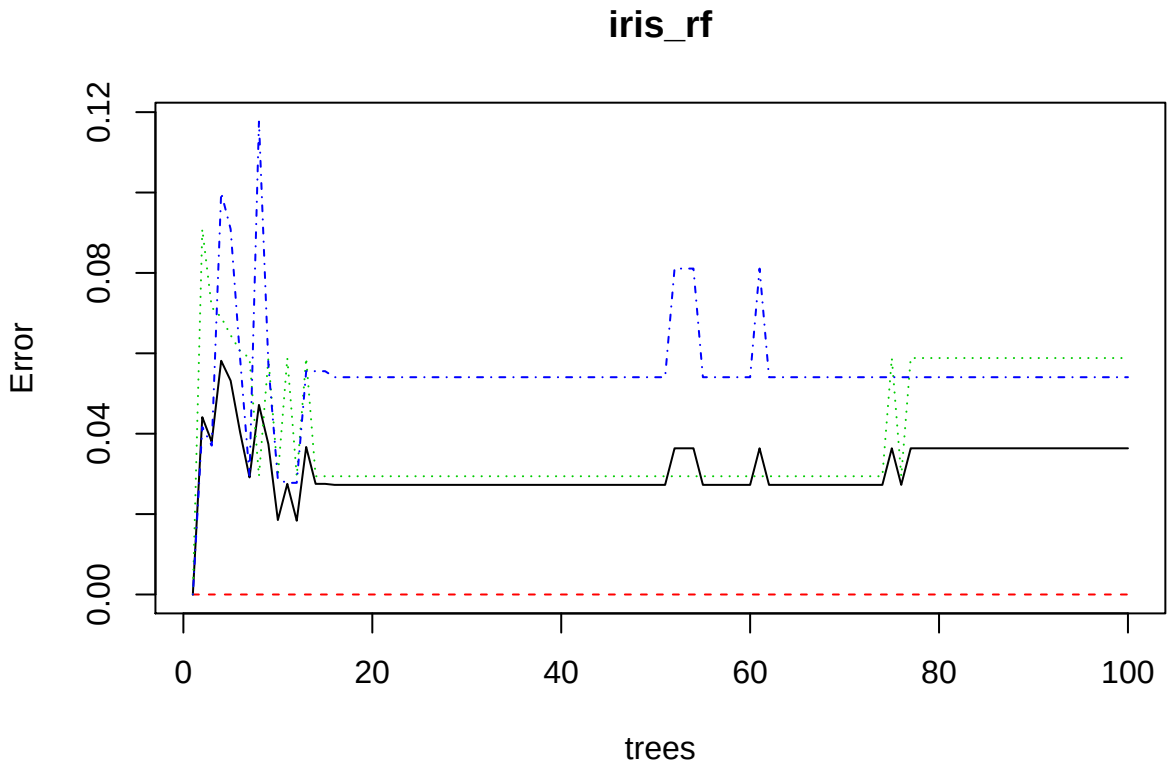
3. Try to print Random Forest model and see the importance features

```
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: 3.64%
## Confusion matrix:
##            setosa versicolor virginica class.error
## setosa         39          0         0  0.00000000
## versicolor      0         32         2  0.05882353
```

```
## virginica       0          2         35  0.05405405
```

```
plot(iris_rf)
```

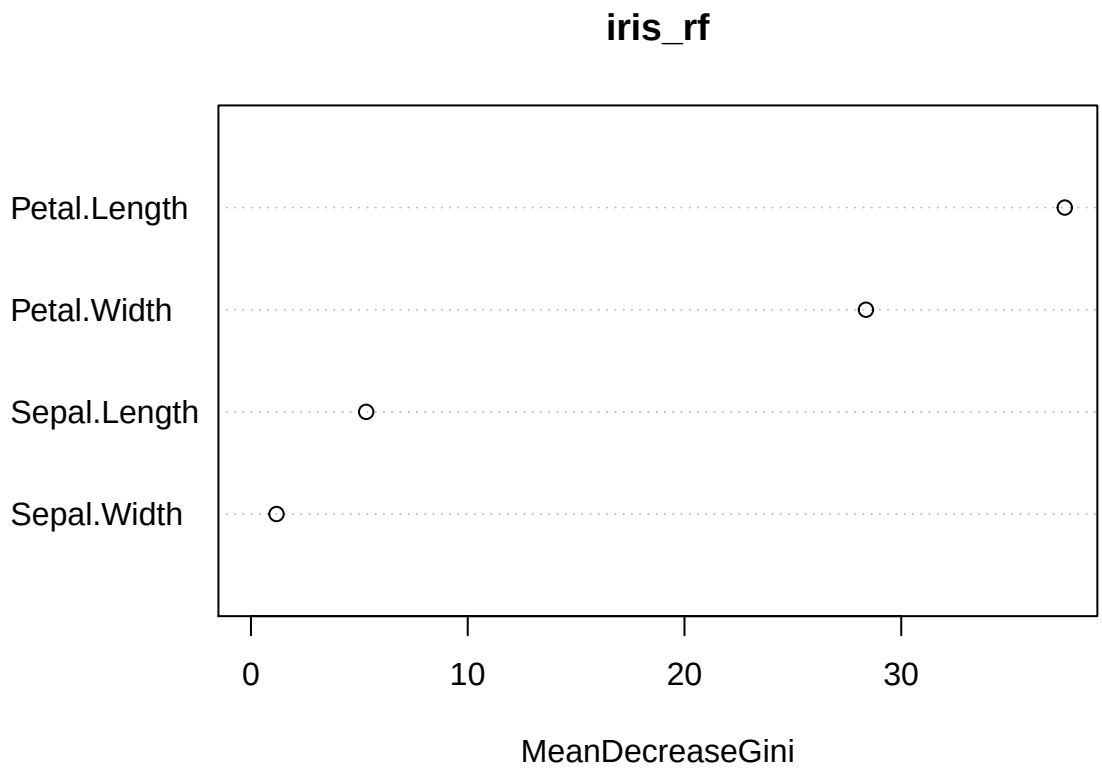


4. Assessing model fit

```
importance(iris_rf)
```

```
##              MeanDecreaseGini
## Sepal.Length         5.315161
## Sepal.Width          1.178239
## Petal.Length        37.545719
## Petal.Width         28.371912
```

```
varImpPlot(iris_rf)
```



5. Predict the class labels for test data

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

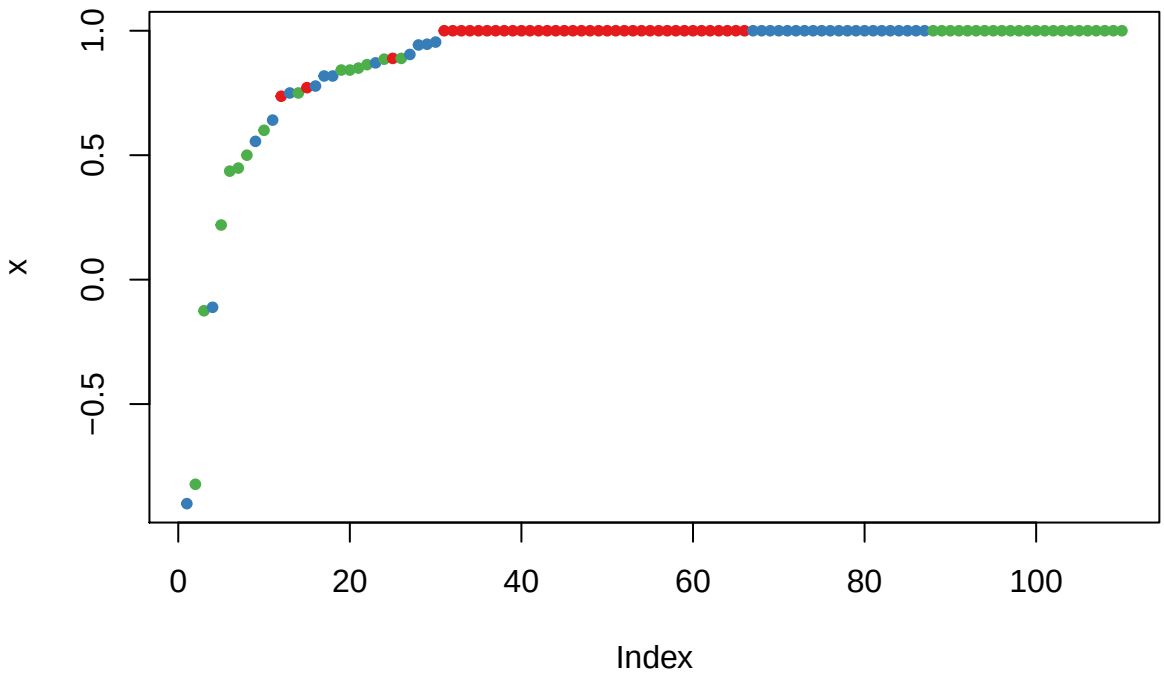
```
##          1          2          6         12         16         18
##     setosa     setosa     setosa     setosa     setosa     setosa
##         19         20         36         48         49         51
##     setosa     setosa     setosa     setosa     setosa versicolor
##         56         60         67         69         73         74
## versicolor versicolor versicolor versicolor versicolor versicolor
##         78         84         85         90         91         92
##  virginica  virginica versicolor versicolor versicolor versicolor
##         95         98         99        104        109        110
## versicolor versicolor versicolor  virginica  virginica  virginica
##        113        121        124        125        131        135
##  virginica  virginica  virginica  virginica  virginica  virginica
##        137        143        144        147
##  virginica  virginica  virginica  virginica
## Levels: setosa versicolor virginica
```

```
table(irisPred, testData$Species)
```

```
##
## irisPred     setosa versicolor virginica
##   setosa         11          0         0
##   versicolor      0         14         0
##   virginica       0          2        13
```

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

method 1

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

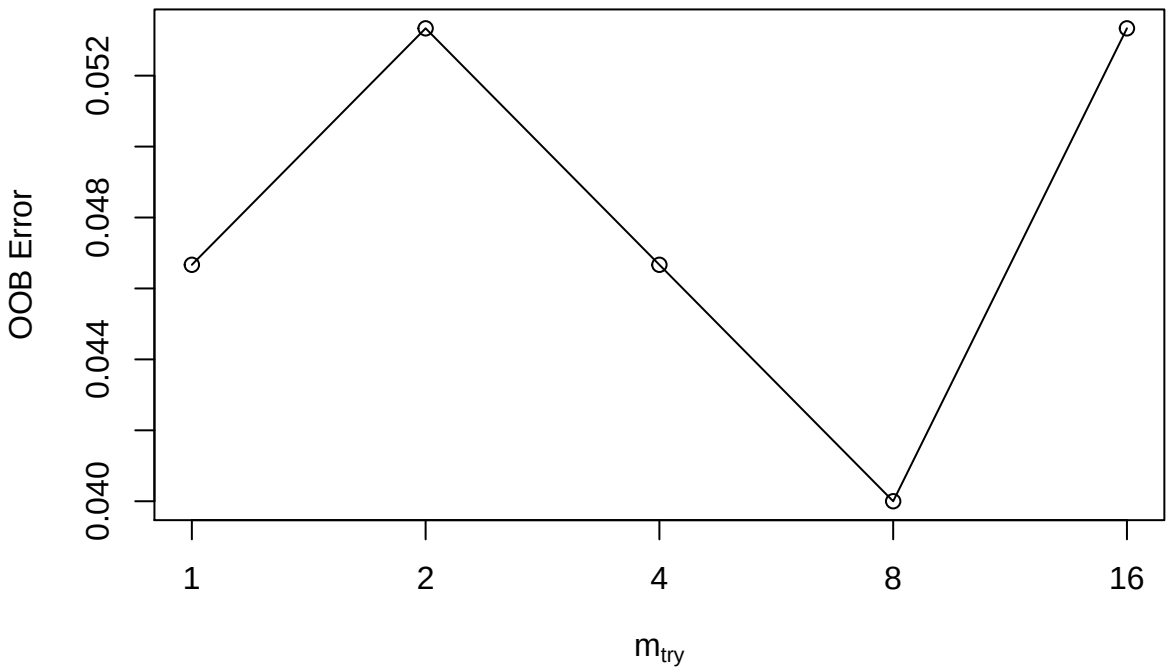
```
## mtry = 2  OOB error = 5.33%
## Searching left ...
## mtry = 4     OOB error = 4.67%
## 0.125 0.05
```

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

```
## mtry = 8     OOB error = 4%
## 0.1428571 0.05
```

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

```
## mtry = 16    OOB error = 5.33%
## -0.3333333 0.05
## Searching right ...
## mtry = 1     OOB error = 4.67%
## -0.1666667 0.05
```




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

```
##        mtry   OOBError
## 1.OOB     1 0.04666667
## 2.OOB     2 0.05333333
## 4.OOB     4 0.04666667
## 8.OOB     8 0.04000000
## 16.OOB   16 0.05333333
```

method 2

We can also tune the structure, ie, finding the best hyperparameters of the method via grid search

```
library(e1071)
tuned.r <- tune(randomForest, train.x = Species ~ ., data = trainData, validation.x = testData)
best.model <- tuned.r$best.model
best.model
```

```
##
## Call:
##  best.tune(method = randomForest, train.x = Species ~ ., data = trainData,      validation.x = testDa
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 3.64%
## Confusion matrix:
##            setosa versicolor virginica class.error
```

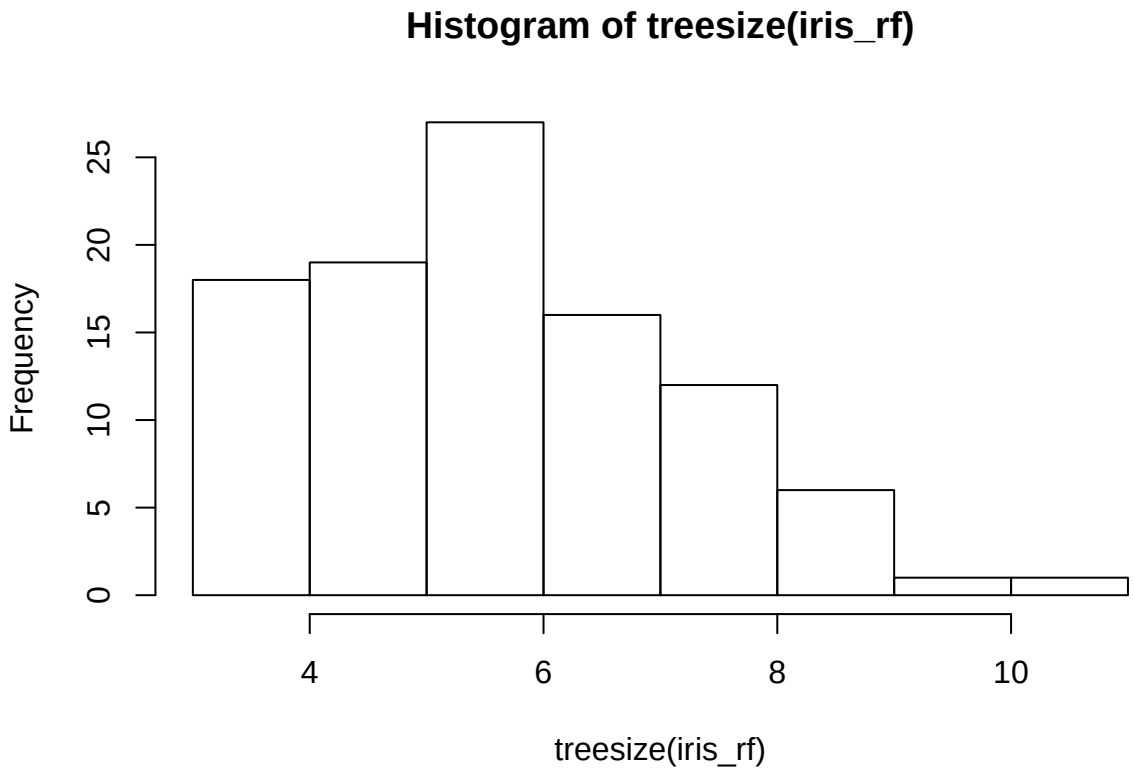
```
## setosa         39          0         0  0.00000000
## versicolor      0         32         2  0.05882353
## virginica       0          2        35  0.05405405
```

```
predictions <- predict(best.model, testData)
table.random.forest <- table(testData$Species, predictions)
table.random.forest
```

```
##             predictions
##              setosa versicolor virginica
##   setosa         11          0         0
##   versicolor      0         14         2
##   virginica       0          0        13
```

8. Tree size

```
hist(treesize(iris_rf))
```



The randomForestSRC Package

```
library(randomForestSRC)
```

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

```
##
```

```
##
## Attaching package: 'randomForestSRC'
```

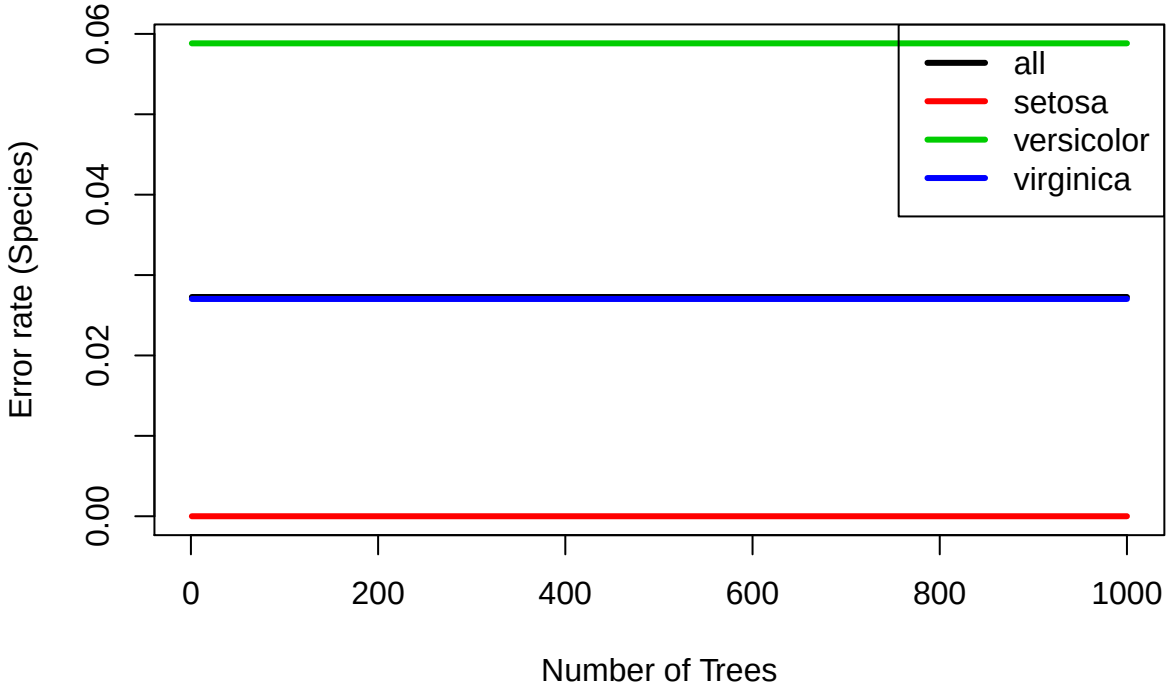
```
## The following object is masked from 'package:e1071':
##
##     impute
```

1. Fitting a random forest

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

```
##                          Sample size: 110
##            Frequency of class labels: 39, 34, 37
##                      Number of trees: 1000
##            Forest terminal node size: 1
##        Average no. of terminal nodes: 6.548
## No. of variables tried at each split: 2
##               Total no. of variables: 4
##                             Analysis: RF-C
##                               Family: class
##                       Splitting rule: gini
##               Normalized Brier score: 9.03
##                           Error rate: 0.03, 0, 0.06, 0.03
##
## Confusion matrix:
##
##             predicted
##   observed   setosa versicolor virginica class.error
##   setosa         39          0         0      0.0000
##   versicolor      0         32         2      0.0588
##   virginica       0          1        36      0.0270
##
##  Overall error rate: 2.73%
```

```
plot(iris.obj)
```




```
# obtain class labels
```

2. predict based on the results of rfsrc

```
rfsrcpred <- predict(iris.obj, testData)

predictions <- rfsrcpred$predicted
predictions
```

```
##       setosa  versicolor virginica
##  [1,]  1.000 0.000000000 0.0000000
##  [2,]  1.000 0.000000000 0.0000000
##  [3,]  0.990 0.010000000 0.0000000
##  [4,]  1.000 0.000000000 0.0000000
##  [5,]  0.876 0.123000000 0.0010000
##  [6,]  1.000 0.000000000 0.0000000
##  [7,]  0.870 0.129000000 0.0010000
##  [8,]  1.000 0.000000000 0.0000000
##  [9,]  1.000 0.000000000 0.0000000
## [10,]  1.000 0.000000000 0.0000000
## [11,]  1.000 0.000000000 0.0000000
## [12,]  0.007 0.890000000 0.1030000
## [13,]  0.000 0.978000000 0.0220000
## [14,]  0.050 0.874000000 0.0760000
## [15,]  0.001 0.960000000 0.0390000
## [16,]  0.000 0.854166667 0.1458333
## [17,]  0.000 0.517666667 0.4823333
```

```
## [18,]  0.000 0.972250000 0.0277500
## [19,]  0.000 0.002000000 0.9980000
## [20,]  0.000 0.097000000 0.9030000
## [21,]  0.052 0.843000000 0.1050000
## [22,]  0.012 0.956000000 0.0320000
## [23,]  0.021 0.929000000 0.0500000
## [24,]  0.000 0.986250000 0.0137500
## [25,]  0.000 0.997000000 0.0030000
## [26,]  0.000 0.989000000 0.0110000
## [27,]  0.024 0.916000000 0.0600000
## [28,]  0.000 0.001000000 0.9990000
## [29,]  0.000 0.003000000 0.9970000
## [30,]  0.001 0.001000000 0.9980000
## [31,]  0.000 0.003000000 0.9970000
## [32,]  0.000 0.001333333 0.9986667
## [33,]  0.000 0.083900000 0.9161000
## [34,]  0.000 0.000000000 1.0000000
## [35,]  0.000 0.001000000 0.9990000
## [36,]  0.000 0.286000000 0.7140000
## [37,]  0.000 0.001500000 0.9985000
## [38,]  0.000 0.004000000 0.9960000
## [39,]  0.000 0.003000000 0.9970000
## [40,]  0.000 0.000000000 1.0000000
```

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)
}
```