# 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         33         3
##   virginica       0          2        30
```

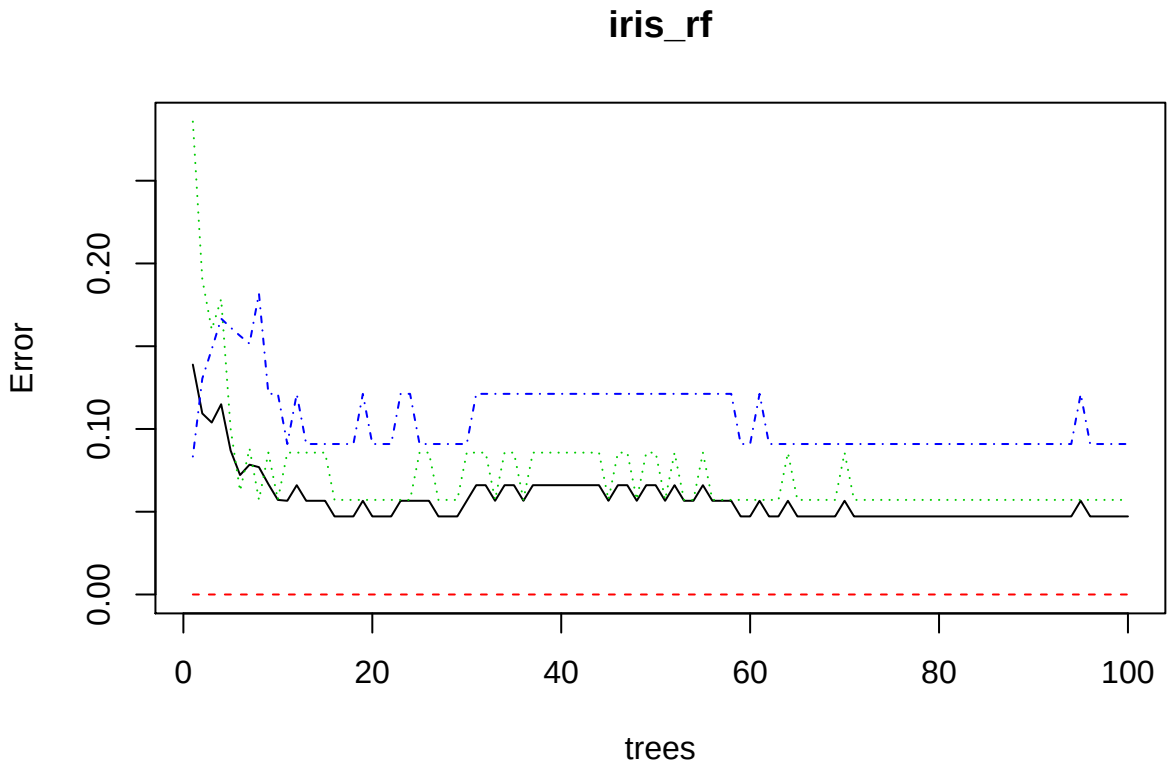
3. Try to print Random Forest model and see the importance features

```
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: 4.72%
## Confusion matrix:
##            setosa versicolor virginica class.error
## setosa         38          0         0  0.00000000
## versicolor      0         33         2  0.05714286
```

```
## virginica       0          3         30  0.09090909
```

```
plot(iris_rf)
```

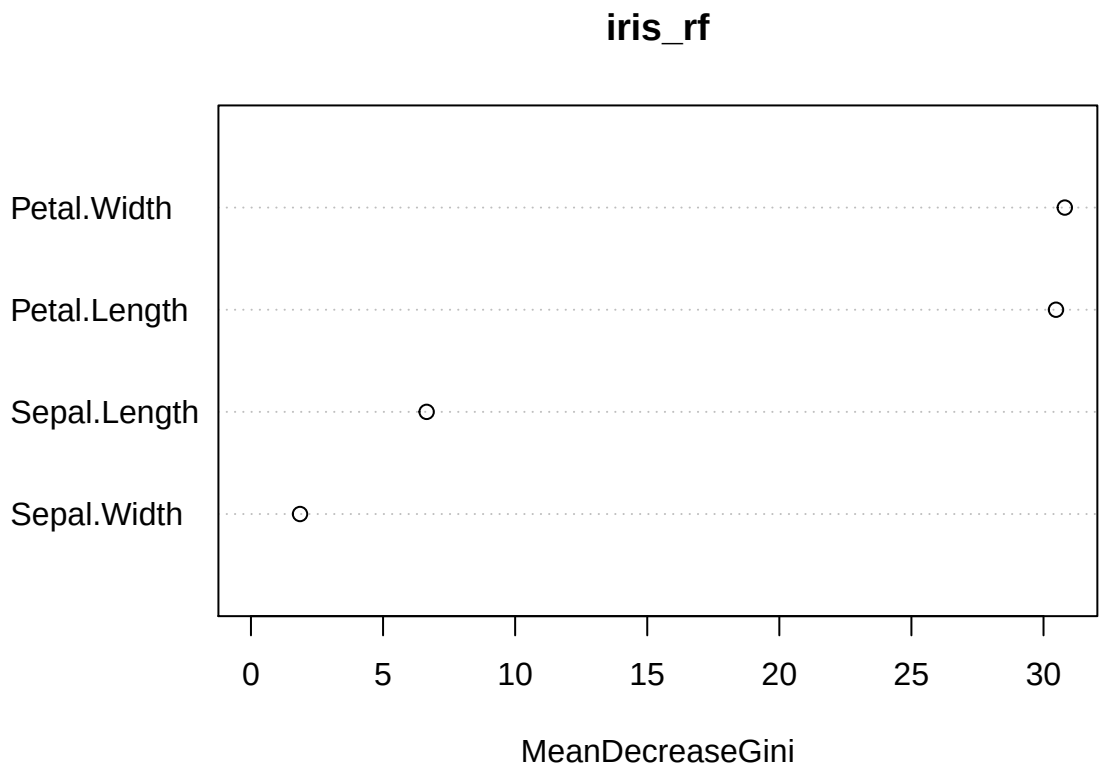



4. Assessing model fit

```
importance(iris_rf)
```

```
##              MeanDecreaseGini
## Sepal.Length         6.649314
## Sepal.Width          1.856770
## Petal.Length        30.472205
## Petal.Width         30.805861
```

```
varImpPlot(iris_rf)
```




5. Predict the class labels for test data

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

```
##          7         11         15         21         23         30
##     setosa     setosa     setosa     setosa     setosa     setosa
##         33         37         39         40         48         49
##     setosa     setosa     setosa     setosa     setosa     setosa
##         52         53         54         74         76         78
## versicolor versicolor versicolor versicolor versicolor  virginica
##         85         86         87         89         92         96
## versicolor versicolor versicolor versicolor versicolor versicolor
##         98         99        100        103        106        113
## versicolor versicolor versicolor  virginica  virginica  virginica
##        114        116        119        120        126        128
##  virginica  virginica  virginica versicolor  virginica  virginica
##        131        136        137        138        139        140
##  virginica  virginica  virginica  virginica  virginica  virginica
##        143        145
##  virginica  virginica
## Levels: setosa versicolor virginica
```

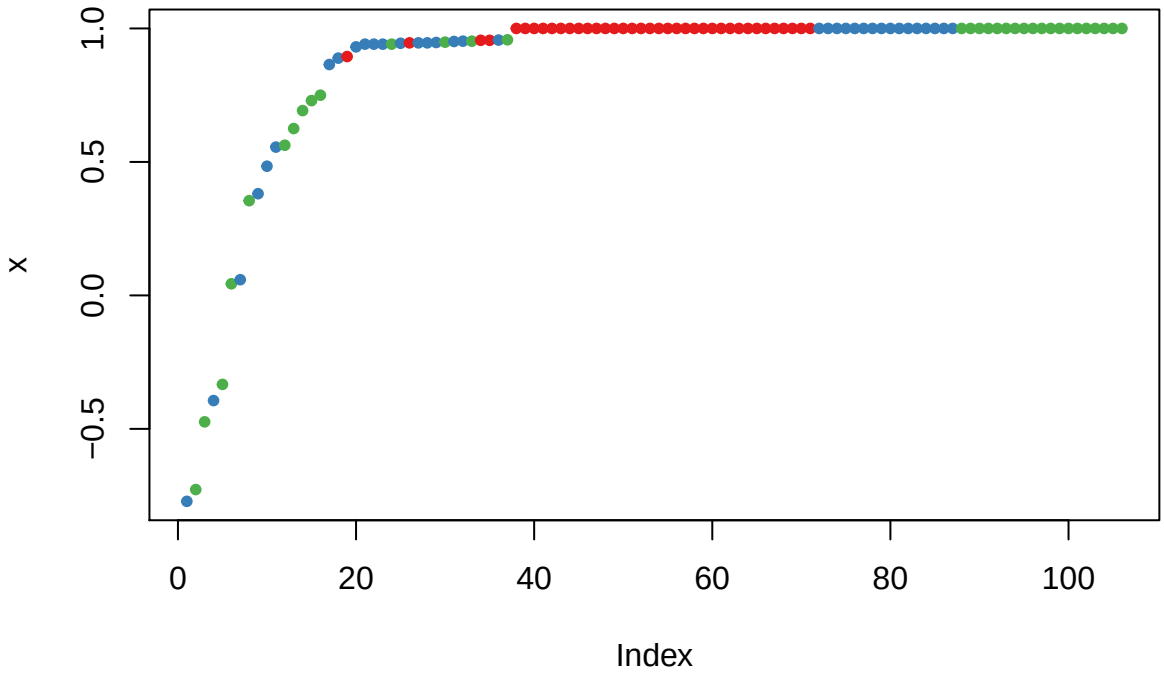
```
table(irisPred, testData$Species)
```

```
##
## irisPred     setosa versicolor virginica
##   setosa         12          0         0
```

```
##    versicolor      0         14         1
##    virginica       0          1        16
```

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

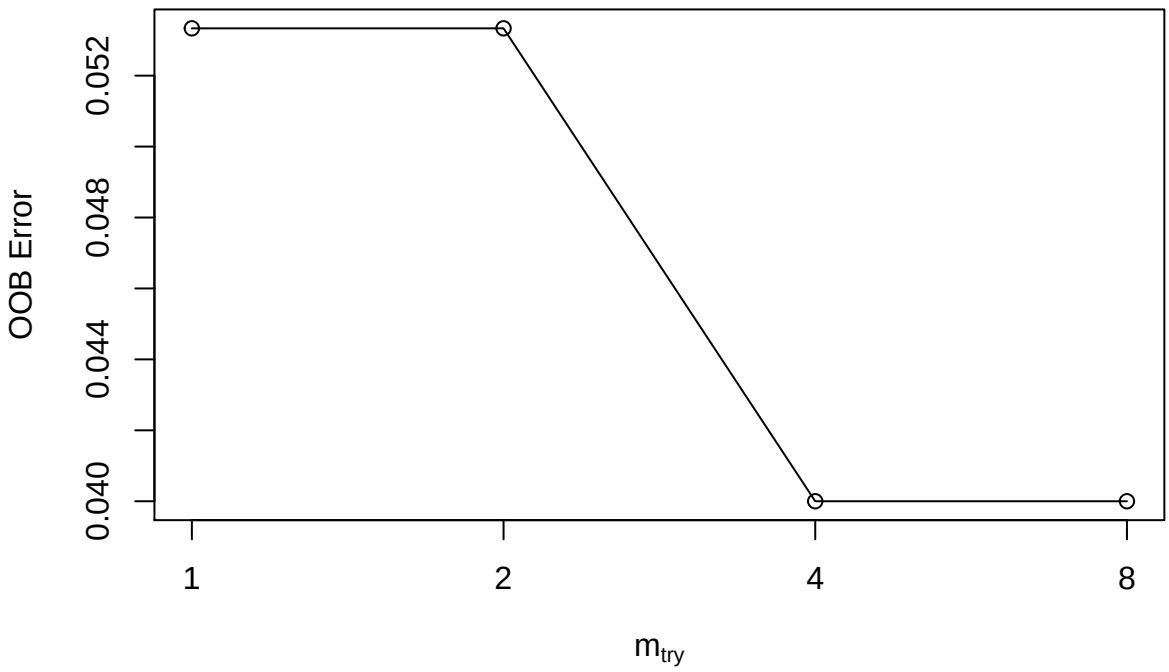
method 1

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

```
## mtry = 2  OOB error = 5.33%
## Searching left ...
## mtry = 4     OOB error = 4%
## 0.25 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 = 5.33%
## -0.3333333 0.05
```




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

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

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: 5.66%
## Confusion matrix:
##            setosa versicolor virginica class.error
## setosa         38          0         0  0.00000000
```

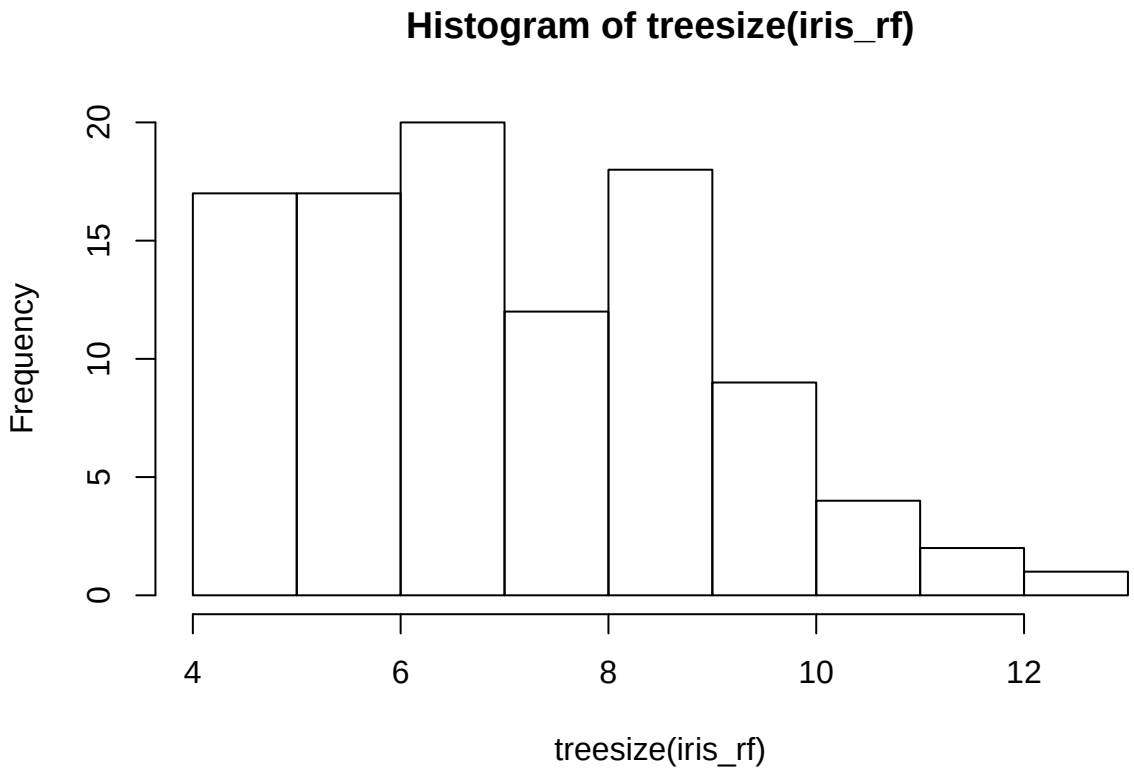
```
## versicolor      0         33         2  0.05714286
## virginica       0          4        29  0.12121212
```

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

```
##             predictions
##              setosa versicolor virginica
##   setosa         12          0         0
##   versicolor      0         14         1
##   virginica       0          1        16
```

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
```

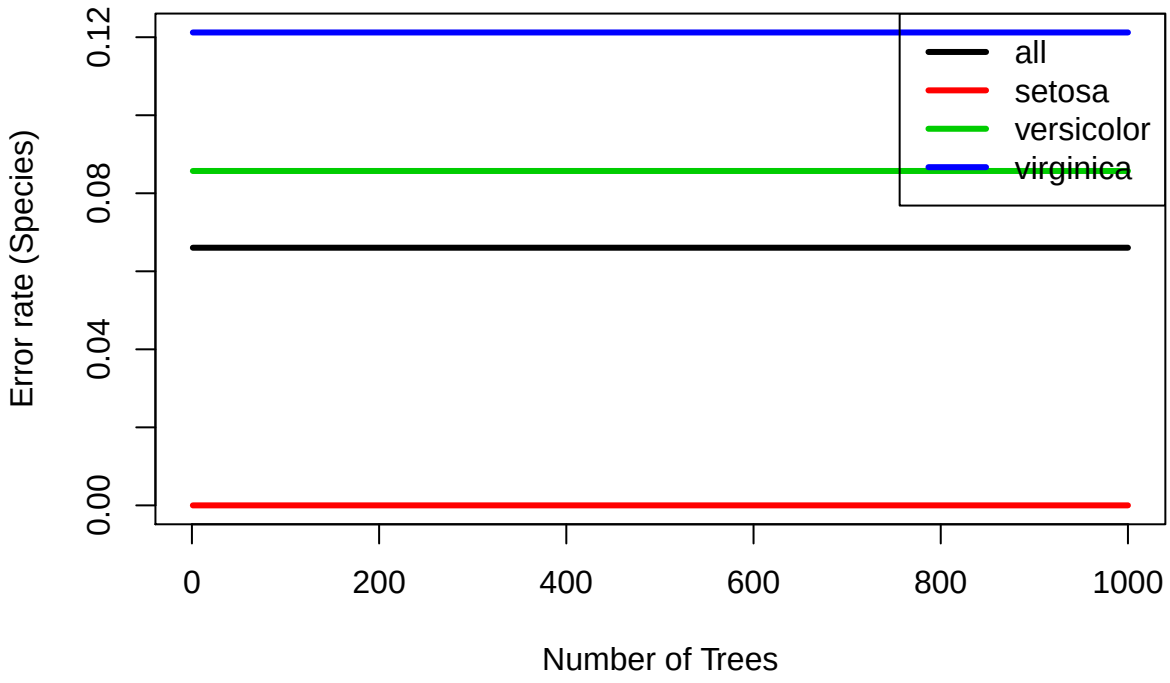
```
##                          Sample size: 106
##            Frequency of class labels: 38, 35, 33
##                      Number of trees: 1000
##            Forest terminal node size: 1
##        Average no. of terminal nodes: 7.23
## No. of variables tried at each split: 2
##               Total no. of variables: 4
##                             Analysis: RF-C
##                               Family: class
##                       Splitting rule: gini
##               Normalized Brier score: 12.89
##                           Error rate: 0.07, 0, 0.09, 0.12
##
## Confusion matrix:
##
##             predicted
##   observed   setosa versicolor virginica class.error
##   setosa         38          0         0      0.0000
##   versicolor      0         32         3      0.0857
##   virginica       0          4        29      0.1212
##
##  Overall error rate: 6.6%
```

```
summary(iris.obj)
```

```
##                 Length Class      Mode
## call               3   -none-     call
## family             1   -none-     character
## n                  1   -none-     numeric
## ntree              1   -none-     numeric
## nimpute            1   -none-     numeric
## mtry               1   -none-     numeric
## nodesize           1   -none-     numeric
## nodedepth          1   -none-     numeric
## nsplit             1   -none-     numeric
## yvar             106   factor     numeric
## yvar.names         1   -none-     character
## xvar               4   data.frame list
## xvar.names         4   -none-     character
## xvar.wt            4   -none-     numeric
## split.wt           4   -none-     numeric
## cause.wt           0   -none-     NULL
## leaf.count      1000   -none-     numeric
```

```
## proximity          0   -none-     NULL
## forest            24   rfsrc      list
## forest.wt          0   -none-     NULL
## membership         0   -none-     NULL
## splitrule          1   -none-     character
## inbag              0   -none-     NULL
## var.used           0   -none-     NULL
## imputed.indv       0   -none-     NULL
## imputed.data       0   -none-     NULL
## split.depth        0   -none-     NULL
## node.stats         0   -none-     NULL
## node.mtry.stats    0   -none-     NULL
## node.mtry.index    0   -none-     NULL
## node.ytry.index    0   -none-     NULL
## tree.err           1   -none-     logical
## predicted        318   -none-     numeric
## class            106   factor     numeric
## predicted.oob    318   -none-     numeric
## class.oob        106   factor     numeric
## err.rate        4000   -none-     numeric
```

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




```
# obtain class labels
```

2. predict based on the results of rfsrc

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

predictions <- rfsrcpred$predicted
predictions
```

```
##       setosa versicolor  virginica
##  [1,]  1.000 0.00000000 0.00000000
##  [2,]  0.999 0.00100000 0.00000000
##  [3,]  0.972 0.02800000 0.00000000
##  [4,]  0.996 0.00400000 0.00000000
##  [5,]  1.000 0.00000000 0.00000000
##  [6,]  1.000 0.00000000 0.00000000
##  [7,]  1.000 0.00000000 0.00000000
##  [8,]  0.991 0.00900000 0.00000000
##  [9,]  1.000 0.00000000 0.00000000
## [10,]  1.000 0.00000000 0.00000000
## [11,]  1.000 0.00000000 0.00000000
## [12,]  0.999 0.00100000 0.00000000
## [13,]  0.003 0.97383333 0.02316667
## [14,]  0.001 0.52016667 0.47883333
## [15,]  0.000 0.99600000 0.00400000
## [16,]  0.000 0.97433333 0.02566667
## [17,]  0.000 0.99700000 0.00300000
## [18,]  0.000 0.00000000 1.00000000
## [19,]  0.035 0.89700000 0.06800000
## [20,]  0.046 0.84633333 0.10766667
## [21,]  0.001 0.95383333 0.04516667
## [22,]  0.002 0.99600000 0.00200000
## [23,]  0.000 0.98733333 0.01266667
## [24,]  0.002 0.99700000 0.00100000
## [25,]  0.000 0.98700000 0.01300000
## [26,]  0.010 0.96600000 0.02400000
## [27,]  0.000 0.99800000 0.00200000
## [28,]  0.000 0.00000000 1.00000000
## [29,]  0.000 0.00000000 1.00000000
## [30,]  0.000 0.00000000 1.00000000
## [31,]  0.000 0.06800000 0.93200000
## [32,]  0.000 0.00400000 0.99600000
## [33,]  0.000 0.00300000 0.99700000
## [34,]  0.000 0.62433333 0.37566667
## [35,]  0.000 0.02600000 0.97400000
## [36,]  0.000 0.09666667 0.90333333
## [37,]  0.000 0.00000000 1.00000000
## [38,]  0.000 0.00000000 1.00000000
## [39,]  0.002 0.00000000 0.99800000
## [40,]  0.000 0.01900000 0.98100000
## [41,]  0.000 0.45085000 0.54915000
## [42,]  0.000 0.00000000 1.00000000
## [43,]  0.000 0.04983333 0.95016667
## [44,]  0.000 0.00100000 0.99900000
```

```r
maxid <- function(arr) {
    # to select the maximum probability
    return(which(arr == max(arr)))
```

```
}

Labels <- colnames(predictions)
idx <- apply(predictions, c(1), maxid)
idx  # this gives the column id corresponds to the maximum probability in the predictions dataframe
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 2 3
## [36] 3 3 3 3 3 3 3 3 3
```

```
prediction.label <- Labels[idx]
prediction.label
```

```
##  [1] "setosa"     "setosa"     "setosa"     "setosa"     "setosa"
##  [6] "setosa"     "setosa"     "setosa"     "setosa"     "setosa"
## [11] "setosa"     "setosa"     "versicolor" "versicolor" "versicolor"
## [16] "versicolor" "versicolor" "virginica"  "versicolor" "versicolor"
## [21] "versicolor" "versicolor" "versicolor" "versicolor" "versicolor"
## [26] "versicolor" "versicolor" "virginica"  "virginica"  "virginica"
## [31] "virginica"  "virginica"  "virginica"  "versicolor" "virginica"
## [36] "virginica"  "virginica"  "virginica"  "virginica"  "virginica"
## [41] "virginica"  "virginica"  "virginica"  "virginica"
```

```
table(prediction.label)
```

```
## prediction.label
##     setosa versicolor  virginica
##         12         15         17
```

3. Plotting commands

```
library("ggRandomForests")
```

```
##
## Attaching package: 'ggRandomForests'
```

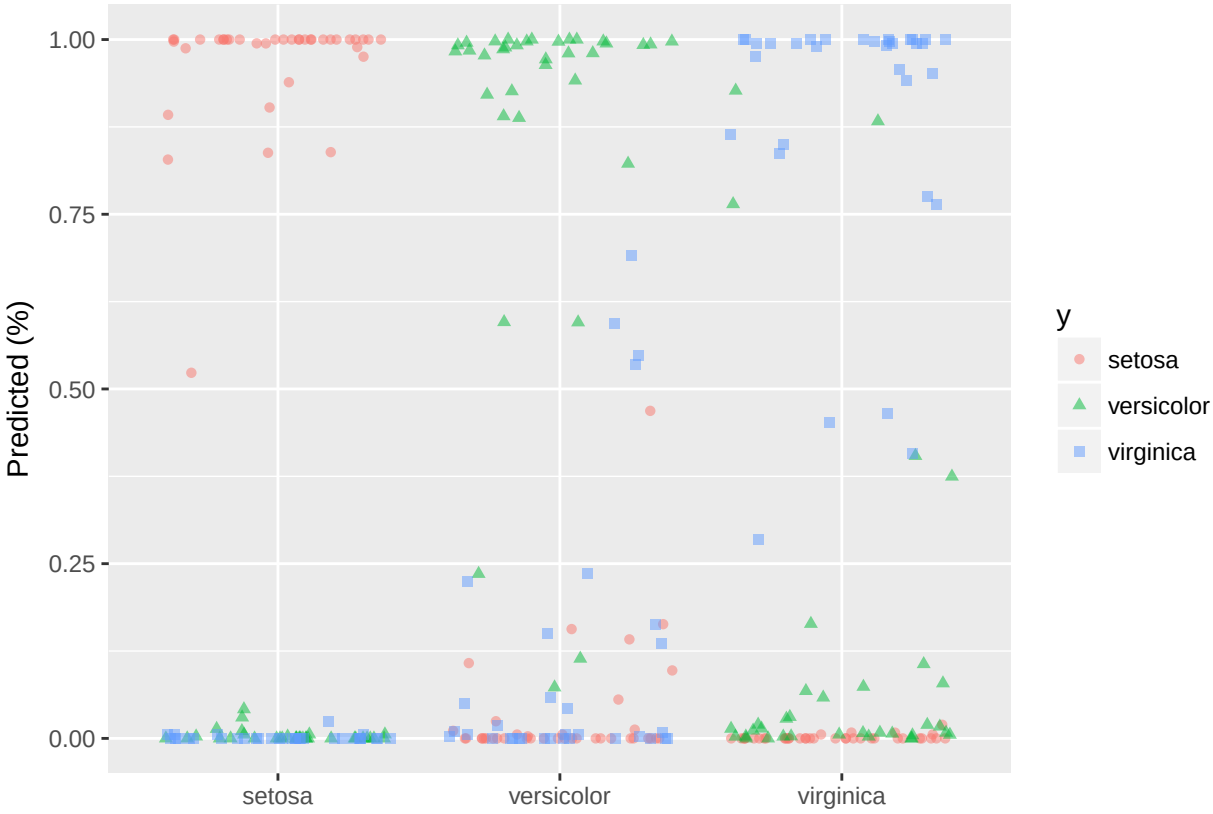
```
## The following object is masked from 'package:randomForestSRC':
##
##     partial.rfsrc
```

```
data(iris.obj, package = "ggRandomForests")
```

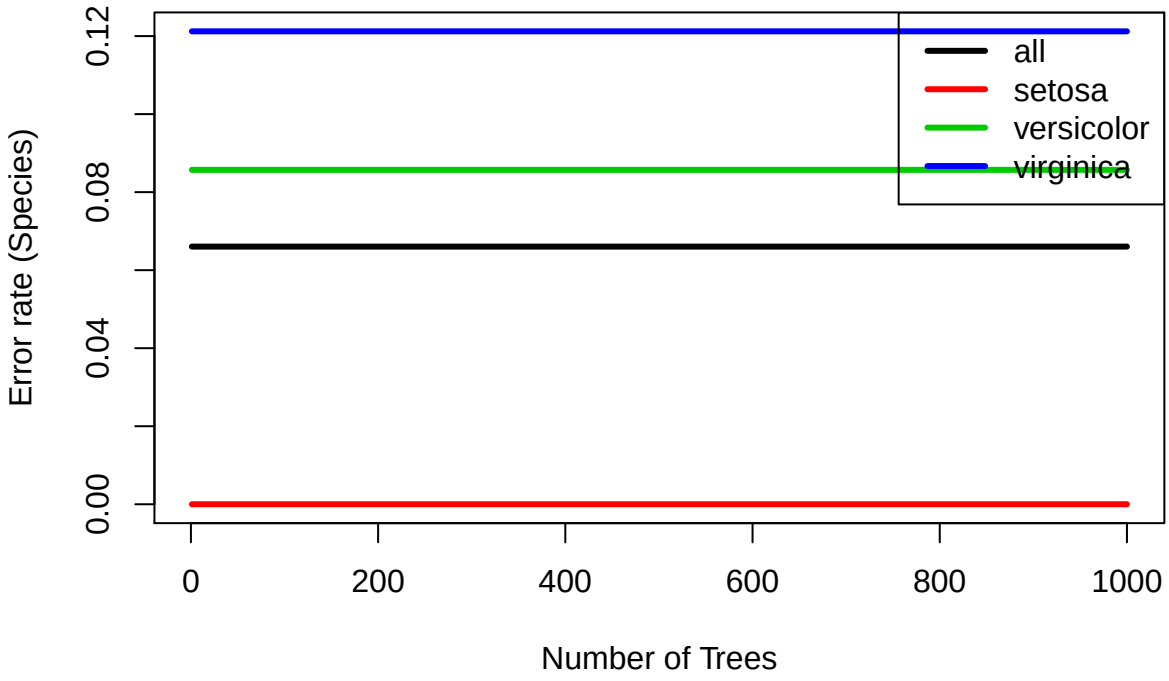
```
## Warning in data(iris.obj, package = "ggRandomForests"): data set 'iris.obj'
## not found
```

```
gg_dta <- gg_rfsrc(iris.obj)

plot(gg_dta)
```




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




### 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)
}
```