randomForest

From randomForest v4.6-14
by Andy Liaw

99.99th
Percentile

Classification And Regression With Random Forest

`randomForest` implements Breiman's random forest algorithm (based on Breiman and Cutler's original Fortran code) for classification and regression. It can also be used in unsupervised mode for assessing proximities among data points.

Keywords regression, classif, tree

Usage

```
# S3 method for formula
randomForest(formula, data=NULL, ..., subset, na.action=na.fail)
# S3 method for default
randomForest(x, y=NULL,  xtest=NULL, ytest=NULL, ntree=500,
             mtry=if (!is.null(y) && !is.factor(y))
             max(floor(ncol(x)/3), 1) else floor(sqrt(ncol(x))),
             replace=TRUE, classwt=NULL, cutoff, strata,
             sampsize = if (replace) nrow(x) else ceiling(.632*nrow(x)),
             nodesize = if (!is.null(y) && !is.factor(y)) 5 else 1,
             maxnodes = NULL,
             importance=FALSE, localImp=FALSE, nPerm=1,
             proximity, oob.prox=proximity,
             norm.votes=TRUE, do.trace=FALSE,
             keep.forest=!is.null(y) && is.null(xtest), corr.bias=FALSE,
             keep.inbag=FALSE, ...)
# S3 method for randomForest
print(x, ...)
```

data	an optional data frame containing the variables in the model. By default the variables are taken from the environment which `randomForest` is called from.
subset	an index vector indicating which rows should be used. (NOTE: If given, this argument must be named.)
na.action	A function to specify the action to be taken if NAs are found. (NOTE: If given, this argument must be named.)
x, formula	a data frame or a matrix of predictors, or a formula describing the model to be fitted (for the `print` method, an `randomForest` object).
y	A response vector. If a factor, classification is assumed, otherwise regression is assumed. If omitted, `randomForest` will run in unsupervised mode.
xtest	a data frame or matrix (like `x`) containing predictors for the test set.
ytest	response for the test set.
ntree	Number of trees to grow. This should not be set to too small a number, to ensure that every input row gets predicted at least a few times.
mtry	Number of variables randomly sampled as candidates at each split. Note that the default values are different for classification (sqrt(p) where p is number of variables in `x`) and regression (p/3)
replace	Should sampling of cases be done with or without replacement?
classwt	Priors of the classes. Need not add up to one. Ignored for regression.
cutoff	(Classification only) A vector of length equal to number of classes. The `winning' class for an observation is the one with the

	maximum ratio of proportion of votes to cutoff. Default is 1/k where k is the number of classes (i.e., majority vote wins).
strata	A (factor) variable that is used for stratified sampling.
sampsize	Size(s) of sample to draw. For classification, if sampsize is a vector of the length the number of strata, then sampling is stratified by strata, and the elements of sampsize indicate the numbers to be drawn from the strata.
nodesize	Minimum size of terminal nodes. Setting this number larger causes smaller trees to be grown (and thus take less time). Note that the default values are different for classification (1) and regression (5).
maxnodes	Maximum number of terminal nodes trees in the forest can have. If not given, trees are grown to the maximum possible (subject to limits by `nodesize`). If set larger than maximum possible, a warning is issued.
importance	Should importance of predictors be assessed?
localImp	Should casewise importance measure be computed? (Setting this to `TRUE` will override `importance` .)
nPerm	Number of times the OOB data are permuted per tree for assessing variable importance. Number larger than 1 gives slightly more stable estimate, but not very effective. Currently only implemented for regression.
proximity	Should proximity measure among the rows be calculated?
oob.prox	Should proximity be calculated only on ``out-of-bag'' data?
norm.votes	If `TRUE` (default), the final result of votes are expressed as fractions. If `FALSE` , raw vote counts are returned (useful for combining results from different runs). Ignored for regression.
do.trace	If set to `TRUE` , give a more verbose output as `randomForest` is

	run. If set to some integer, then running output is printed for every `do.trace` trees.
keep.forest	If set to `FALSE`, the forest will not be retained in the output object. If `xtest` is given, defaults to `FALSE`.
corr.bias	perform bias correction for regression? Note: Experimental. Use at your own risk.
keep.inbag	Should an `n` by `ntree` matrix be returned that keeps track of which samples are ``in-bag'' in which trees (but not how many times, if sampling with replacement)
...	optional parameters to be passed to the low level function `randomForest.default`.

Value

An object of class `randomForest`, which is a list with the following components:

call

the original call to `randomForest`

type

one of `regression`, `classification`, or `unsupervised`.

predicted

the predicted values of the input data based on out-of-bag samples.

importance

a matrix with `nclass` + 2 (for classification) or two (for regression) columns. For classification, the first `nclass` columns are the class-specific measures computed as mean descrease in accuracy. The `nclass` + 1st column is the mean descrease in accuracy over all classes. The last column is the mean decrease in Gini index. For Regression, the first column is the mean decrease in accuracy and the second the mean decrease in MSE. If `importance=FALSE`, the last measure is still returned as a vector.

importanceSD

The ``standard errors'' of the permutation-based importance measure. For classification, a `p` by `nclass + 1` matrix corresponding to the first `nclass + 1` columns of the importance matrix. For regression, a length `p` vector.

localImp

a p by n matrix containing the casewise importance measures, the [i,j] element of which is the importance of i-th variable on the j-th case. `NULL` if `localImp=FALSE`.

ntree

number of trees grown.

mtry

number of predictors sampled for spliting at each node.

forest

(a list that contains the entire forest; `NULL` if `randomForest` is run in unsupervised mode or if `keep.forest=FALSE`.

err.rate

(classification only) vector error rates of the prediction on the input data, the i-th element being the (OOB) error rate for all trees up to the i-th.

confusion

(classification only) the confusion matrix of the prediction (based on OOB data).

votes

(classification only) a matrix with one row for each input data point and one column for each class, giving the fraction or number of (OOB) \`votes' from the random forest.

oob.times

number of times cases are \`out-of-bag' (and thus used in computing OOB error estimate)

proximity

if `proximity=TRUE` when `randomForest` is called, a matrix of proximity measures among the input (based on the frequency that pairs of data points are in the same terminal nodes).

mse

(regression only) vector of mean square errors: sum of squared residuals divided by `n` .

rsq

(regression only) \`\`pseudo R-squared'': 1 - `mse` / Var(y).

test

if test set is given (through the `xtest` or additionally `ytest` arguments), this component is a list which contains the corresponding `predicted` , `err.rate` , `confusion` , `votes` (for classification) or `predicted` , `mse` and `rsq` (for regression) for the test set. If `proximity=TRUE` , there is also a component, `proximity` , which contains the proximity among the test set as well as proximity between test and training data.

Note

The `forest` structure is slightly different between classification and regression. For details on how the trees are stored, see the help page for `getTree` .

If `xtest` is given, prediction of the test set is done ``in place'' as the trees are grown. If `ytest` is also given, and `do.trace` is set to some positive integer, then for every `do.trace` trees, the test set error is printed. Results for the test set is returned in the `test` component of the resulting `randomForest` object. For classification, the `votes` component (for training or test set data) contain the votes the cases received for the classes. If `norm.votes=TRUE` , the fraction is given, which can be taken as predicted probabilities for the classes.

For large data sets, especially those with large number of variables, calling `randomForest` via the formula interface is not advised: There may be too much overhead in handling the formula.

The ``local'' (or casewise) variable importance is computed as follows: For classification, it is the increase in percent of times a case is OOB and misclassified when the variable is permuted. For regression, it is the average increase in squared OOB residuals when the variable is permuted.

References

Breiman, L. (2001), *Random Forests*, Machine Learning 45(1), 5-32.

Breiman, L (2002), ``Manual On Setting Up, Using, And Understanding Random Forests V3.1'', https://www.stat.berkeley.edu/~breiman/Using_random_forests_V3.1.pdf.

See Also

`predict.randomForest` , `varImpPlot`

Examples

```
# NOT RUN {
## Classification:
##data(iris)
set.seed(71)
iris.rf <- randomForest(Species ~ ., data=iris, importance=TRUE,
                        proximity=TRUE)
print(iris.rf)
## Look at variable importance:
round(importance(iris.rf), 2)
## Do MDS on 1 - proximity:
iris.mds <- cmdscale(1 - iris.rf$proximity, eig=TRUE)
op <- par(pty="s")
pairs(cbind(iris[,1:4], iris.mds$points), cex=0.6, gap=0,
      col=c("red", "green", "blue")[as.numeric(iris$Species)],
      main="Iris Data: Predictors and MDS of Proximity Based on RandomFore
par(op)
print(iris.mds$GOF)

## The `unsupervised' case:
set.seed(17)
iris.urf <- randomForest(iris[, -5])
MDSplot(iris.urf, iris$Species)

## stratified sampling: draw 20, 30, and 20 of the species to grow each tr
(iris.rf2 <- randomForest(iris[1:4], iris$Species,
                          sampsize=c(20, 30, 20)))

## Regression:
## data(airquality)
set.seed(131)
ozone.rf <- randomForest(Ozone ~ ., data=airquality, mtry=3,
                         importance=TRUE, na.action=na.omit)
print(ozone.rf)
## Show "importance" of variables: higher value mean more important:
round(importance(ozone.rf), 2)

## "x" can be a matrix instead of a data frame:
set.seed(17)
x <- matrix(runif(5e2), 100)
y <- gl(2, 50)
(myrf <- randomForest(x, y))
(predict(myrf, x))
```

```
## "complicated" formula:
(swiss.rf <- randomForest(sqrt(Fertility) ~ . - Catholic + I(Catholic < 50
                          data=swiss))
(predict(swiss.rf, swiss))
## Test use of 32-level factor as a predictor:
set.seed(1)
x <- data.frame(x1=gl(53, 10), x2=runif(530), y=rnorm(530))
(rf1 <- randomForest(x[-3], x[[3]], ntree=10))

## Grow no more than 4 nodes per tree:
(treesize(randomForest(Species ~ ., data=iris, maxnodes=4, ntree=30)))

## test proximity in regression
iris.rrf <- randomForest(iris[-1], iris[[1]], ntree=101, proximity=TRUE, o
str(iris.rrf$proximity)
# }
```

Community examples

Looks like there are no examples yet.

Post a new example:

~~~
## New example
Use markdown to format your example

R code blocks are runnable and interactive:
```r
a <- 2
print(a)
```


You can also display normal code blocks
```

var a = b
```
~~~

~~~
## New example
Use markdown to format your example

R code blocks are runnable and interactive:
```r
a <- 2
print(a)
```


You can also display normal code blocks
```

var a = b
```
~~~

Submit your example