

# Package ‘Boruta’

February 22, 2012

**Title** Boruta -- a tool for finding significant attributes in information systems

**Version** 1.6

**Date** 2010-11-25

**Depends** R (>= 2.0.0), randomForest

**Suggests** mlbench

**Author** Algorithm by Witold R. Rudnicki <W.Rudnicki@icm.edu.pl>, R  
implementation by Miron B. Kursa <M.Kursa@icm.edu.pl>.

**Description** Boruta is a feature selection algorithm based on a  
randomForest classifier. It selects the full set of all relevant attributes by comparing original at-  
tributes’ importances with importance achievable at random estimated  
using their randomised copies.

**Maintainer** Miron B. Kursa <M.Kursa@icm.edu.pl>

**License** GPL (>= 2)

**Repository** CRAN

**Date/Publication** 2012-02-22 14:09:03

## R topics documented:

attStats . . . . .	2
Boruta . . . . .	3
Boruta plot . . . . .	5
Get formula . . . . .	7
plotZHistory . . . . .	8
TentativeRoughFix . . . . .	9

<b>Index</b>	<b>10</b>
--------------	-----------

---

`attStats`*Attribute statistics*

---

**Description**

`attStats` shows a summary of a Boruta run in an attribute-centred way. It produces a data frame containing some ZScore stats as well as the number of hits that attribute scored and the decision it was given.

**Usage**

```
attStats(x)
```

**Arguments**

`x` an object of class Boruta.

**Value**

A data frame containing, for each attribute that was originally in information system, mean, median, maximal and minimal ZScore, number of hits normalised to number of random forest runs performed and the decision copied from `finalDecision`.

**Note**

When using a Boruta object generated by a [TentativeRoughFix](#), the resulting data frame will consist a rough fixed decision.

**Author(s)**

Miron B. Kursa

**Examples**

```
## Not run:
library(mlbench); data(Sonar);
#Takes some time, so be patient
Boruta(Class~.,data=Sonar,doTrace=2)->Bor.son;
print(Bor.son);
stats<-attStats(Bor.son);
print(stats);
plot(normHits~meanZ,col=stats$decision,data=stats);

## End(Not run)
```

Boruta

*Important attribute search using Boruta algorithm***Description**

Boruta is an algorithm of finding important attributes in information systems by iterative learning of the randomForest classifier.

**Usage**

```
## S3 method for class 'formula'
Boruta(formula,data=.GlobalEnv,...)
## Default S3 method:
Boruta(x,y,confidence=0.999,maxRuns=100,light=TRUE,doTrace=0,getImp=getImpRf,...)
## S3 method for class 'Boruta'
print(x,...)
```

**Arguments**

x, formula	data frame of predictors or a formula describing model to be analysed.
data	data frame containing model variables. Global environment is default.
y	response vector; factor for classification, numeric vector for regression.
confidence	confidence level. Default value should be used. Lower value may reduce computation time of test runs.
maxRuns	maximal number of randomForest runs in the final round. You may increase it to resolve attributes left Tentative.
doTrace	0 means no tracing, 1 means printing a "." sign after each randomForest run, 2 means same as 1, plus consecutive reporting of test results.
getImp	Function used to obtain attribute importance. The default is getImpRf, which runs randomForest and gathers Z-scores of mean decrease accuracy measure.
light	if set to TRUE, Boruta runs in standard, light mode. If set to FALSE, Boruta runs in more restrictive, force mode.
...	additional parameters that will be passed to getImp function.

**Details**

Boruta iteratively compares ZScores of attributes with ZScores of shadow attributes, created by shuffling original ones. Attributes that have significantly worst importance than shadow ones are being consecutively dropped. On the other hand, attributes that are significantly better than shadows are admitted to be Confirmed. If algorithm is run in default light mode, unimportant attributes are being dropped along with their random shadows, while in the force mode all shadow attributes are preserved during the whole Boruta run. Algorithm stops when only Confirmed attributes are left, or when it reaches maxRuns randomForest runs in the last round. If the second scenario occurs, some attributes may be left without a decision. They are claimed Tentative. You may try to extend maxRuns or lower confidence to clarify them, but in some cases their ZScores do fluctuate

too much for Boruta to converge. Instead, you can use `TentativeRoughFix` function, which will perform other, weaker test to make a final decision, or simply treat them as undecided in further analysis.

### Value

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

<code>finalDecision</code>	a factor of three value: <code>Confirmed</code> , <code>Rejected</code> or <code>Tentative</code> , containing final result of feature selection.
<code>ZScoreHistory</code>	a data frame of ZScores of attributes gathered in each <code>randomForest</code> run. Beside predictors' ZScores contains maximal, mean and minimal ZScore of shadow attributes in each run. Rejected attributes have <code>-Inf</code> ZScore assumed.
<code>timeTaken</code>	time taken by the computation.
<code>impSource</code>	string describing the source of importance, equal to a comment attribute of the <code>getImp</code> argument.
<code>call</code>	the original call of the <code>Boruta</code> function.

### Note

`Boruta` was originally based on a Z-score of `randomForest`'s mean decrease of accuracy score. While the current version of the package allows one to use arbitrary importance source using the `getImp` argument, the documentation and function/element naming still follow the original implementation because of the compatibility issues.

### Author(s)

Miron B. Kursa, based on the idea & original code by Witold R. Rudnicki.

### References

Miron B. Kursa, Witold R. Rudnicki (2010). Feature Selection with the Boruta Package. *Journal of Statistical Software*, 36(11), p. 1-13. URL: <http://www.jstatsoft.org/v36/i11/>

### Examples

```
set.seed(777);
#Add some nonsense attributes to iris dataset by shuffling original attributes
iris.extended<-data.frame(iris,apply(iris[,-5],2,sample));
names(iris.extended)[6:9]<-paste("Nonsense",1:4,sep="");
#Run Boruta on this data
Boruta(Species~.,data=iris.extended,doTrace=2)->Boruta.iris.extended
#Nonsense attributes should be rejected
print(Boruta.iris.extended);

## Not run:
#Boruta using rFerns' importance (rFerns package must be installed!)
#Definition of ferns' importance adapter
getImpFerns<-function(x,y,...){
  f<-rFerns(x,y,saveForest=FALSE,importance=TRUE,...);
```

```

    f$importance[,1]
  }
  #Those are optional
  attr(getImpFerns,"toLoad")<-"rFerns";
  comment(getImpFerns)<-"rFerns importance"
  #Running altered Boruta on the Iris data
  Boruta(Species~.,data=iris.extended,getImp=getImpFerns)->Boruta.ferns.irisE
  print(Boruta.ferns.irisE);

## End(Not run)

## Not run:
#Boruta on the Ozone data from mlbench
library(mlbench); data(Ozone);
na.omit(Ozone)->ozo;
#Takes some time, so be patient
Boruta(V4~.,data=ozo,doTrace=2)->Bor.ozo;
cat('Random forest run on all attributes:\n');
print(randomForest(V4~.,data=ozo));
cat('Random forest run only on confirmed attributes:\n');
print(randomForest(getConfirmedFormula(Bor.ozo),data=ozo));

## End(Not run)

## Not run:
#Boruta on the HouseVotes84 data from mlbench
library(mlbench); data(HouseVotes84);
na.omit(HouseVotes84)->hvo;
#Takes some time, so be patient
Boruta(Class~.,data=hvo,doTrace=2)->Bor.hvo;
print(Bor.hvo);
plot(Bor.hvo);

## End(Not run)

## Not run:
#Boruta on the Sonar data from mlbench
library(mlbench); data(Sonar);
#Takes some time, so be patient
Boruta(Class~.,data=Sonar,doTrace=2)->Bor.son;
print(Bor.son);
#Shows important bands
plot(Bor.son,sort=FALSE);

## End(Not run)

```

**Description**

`plot.Boruta` visualises `ZScoreHistory` element of the `Boruta` object. Each attribute (including randomised meta-attributes, `randMax`, `randMean`, `randMin`) is represented as a boxplot. Its colour corresponds to a Boruta decision; see `colCode` argument note. Boxplots are ordered due to their median `ZScore`.

**Usage**

```
## S3 method for class 'Boruta'
plot(x,colCode=c('green','yellow','red','blue'),sort=TRUE,
      whichRand=c(TRUE,TRUE,TRUE),col=NULL,
      xlab='Attributes',ylab='Importance',...)
```

**Arguments**

<code>x</code>	an object of class <code>Boruta</code> .
<code>colCode</code>	a vector containing colour codes for attribute decisions, respectively <code>Confirmed</code> , <code>Tentative</code> , <code>Rejected</code> , <code>Random</code> .
<code>sort</code>	controls weather boxplots should be ordered, or left in original order.
<code>whichRand</code>	a vector controlling which randomised meta-attributes should be drawn; respectively <code>randMax</code> , <code>randMean</code> , <code>randMin</code> .
<code>col</code>	standard <code>col</code> attribute. If given, suppresses <code>colCode</code> .
<code>xlab,ylab,...</code>	additional graphical parameters that will be passed to <code>boxplot</code> .

**Value**

None.

**Note**

If `col` is given and `sort` is `TRUE`, the `col` will be permuted, so that its order corresponds to attribute order in `ZScoreHistory`.

**Author(s)**

Miron B. Kursa

**Examples**

```
## Not run:
library(mlbench); data(HouseVotes84);
na.omit(HouseVotes84)->hvo;
#Takes some time, so be patient
Boruta(Class~.,data=hvo,doTrace=2)->Bor.hvo;
print(Bor.hvo);
plot(Bor.hvo);

## End(Not run)
```

**Description**

This two functions convert Boruta result into a convenient formula object, that may be easily passed to other R functions without additional parsing. `getConfirmedFormula` includes Confirmed attributes in formula only, `getNonRejectedFormula` includes both Confirmed and Tentative.

**Usage**

```
getConfirmedFormula(x)
getNonRejectedFormula(x)
```

**Arguments**

`x` an object of class Boruta.

**Value**

A formula corresponding to `Boruta$finalDecision`.

**Note**

Those functions work only if Boruta has been run using formula object as a model definition. They return error otherwise.

**Author(s)**

Miron B. Kursa

**Examples**

```
## Not run:
library(mlbench); data(Ozone);
na.omit(Ozone)->ozo;
#Takes some time, so be patient
Boruta(V4~.,data=ozo,doTrace=2)->Bor.ozo;
cat('Random forest run on all attributes:\n');
print(randomForest(V4~.,data=ozo));
cat('Random forest run only on confirmed attributes:\n');
print(randomForest(getConfirmedFormula(Bor.ozo),data=ozo));

## End(Not run)
```

---

plotZHistory

*Plotting history of importance*


---

### Description

plotZHistory visualises ZScoreHistory element of the Boruta object in some more straightforward way than plot.Boruta does. Namely, it plots each attributes' (including randomised meta-attributes, randMax, randMean, randMin) importance as a function of Random Forest run it was obtained in. The colour of each line corresponds to the final Boruta decision; see colCode argument note.

### Usage

```
plotZHistory(x,colCode=c('green','yellow','red','blue'),showRounds=TRUE,
col=NULL,type="l",lty=1,pch=0,
xlab='Random Forest run',ylab='Importance',...)
```

### Arguments

x	an object of class Boruta.
colCode	a vector containing colour codes for attribute decisions, respectively Confirmed, Tentative, Rejected, Random.
showRounds	if set, to TRUE, gray lines separating round will be drawn.
col	standard col attribute. If given, suppresses colCode.
type,lty,pch,xlab,ylab,...	additional graphical parameters that will be passed to matplot.

### Value

None.

### Author(s)

Miron B. Kursa

### Examples

```
## Not run:
library(mlbench); data(Sonar);
#Takes some time, so be patient
Boruta(Class~.,data=Sonar,doTrace=2)->Bor.son;
print(Bor.son);
plotZHistory(Bor.son);

## End(Not run)
```

---

TentativeRoughFix	<i>Rough test for tentative attributes</i>
-------------------	--

---

**Description**

In some circumstances (too short Boruta run, unfortunate mixing of shadow attributes, tricky dataset...), Boruta can leave some attributes Tentative. TentativeRoughFix performs a simplified, weaker test for judging such attributes.

**Usage**

```
TentativeRoughFix(x, averageOver='finalRound')
```

**Arguments**

x	an object of class Boruta.
averageOver	Either number of last randomForest runs to average over, 'finalRound' for averaging over last (final) round or 'allRounds' for averaging over whole Boruta run.

**Details**

Function claims as Confirmed those attributes that have median ZScore higher than the median ZScore of maximal shadow attribute (maxRand), and the rest as Rejected. Depending of the user choice, medians for the test are count over last round, all rounds or N last randomForest runs.

**Value**

A Boruta class object with modified finalDecision element. Such object has few additional elements:

originalDecision	Original finalDecision.
averageOver	Copy of averageOver parameter.

**Note**

This function should be used only when strict decision is highly desired, because this test is much weaker than Boruta and can lower the confidence of the final result.

**Author(s)**

Miron B. Kursa

# Index

\*Topic **hplot**

Boruta plot, 5  
plotZHistory, 8

\*Topic **tree**

Boruta, 3

attStats, 2

Boruta, 3

Boruta plot, 5

Get formula, 7

getConfirmedFormula (Get formula), 7

getNonRejectedFormula (Get formula), 7

plot.Boruta (Boruta plot), 5

plotZHistory, 8

print.Boruta (Boruta), 3

TentativeRoughFix, 2, 4, 9