Package 'varSelRF'

January 31, 2026

Version 0.7-9

Date 2026-01-30

Title Variable Selection using Random Forests

Maintainer Ramon Diaz-Uriarte <rdiaz02@gmail.com>

Depends R (>= 2.0.0), randomForest, parallel

Description Variable selection from random forests using both backwards variable elimination (for the selection of small sets of non-redundant variables) and selection based on the importance spectrum (somewhat similar to scree plots; for the selection of large, potentially highly-correlated variables). Main applications in high-dimensional data (e.g., microarray data, and other genomics and proteomics applications).

LazyLoad Yes

License GPL (>= 2)

URL <https://ligarto.org/rdiaz/Software/Software.html>, <https://github.com/rdiaz02/varSelRF>

Repository CRAN

Date/Publication 2026-01-30 23:40:02 UTC

NeedsCompilation no

Author Ramon Diaz-Uriarte [aut, cre]

Contents

---

plot.varSelRF   *Plot a varSelRF object*

Description

Plots a varSelRF object, showing the initial variable importances, and the change in OOB error with the number of variables.

Usage

```
## S3 method for class 'varSelRF'
plot(x, nvar = NULL, which = c(1, 2), ...)
```

Arguments

x	The varSelRF object.
nvar	The number of variables for which the initial variable importances should be shown. By default, only the 30 with the largest importance are shown.
which	which plots should be drawn, either 1 (for the initial variable importance plot), 2 (for the change in OOB error with the number of variables) or c(1,2) for drawing both plots
...	Not used.

Value

This function is only used for its side effect of producing plots.

Warning

The OOB Error rate is biased down (and can be severely biased down) because we do (potentially many) rounds of reducing the set of predictor variables until we minimize this OOB error rate.

Author(s)

Ramon Diaz-Uriarte <rdiaz02@gmail.com>

References

Diaz-Uriarte, R. , Alvarez de Andres, S. (2006) Gene selection and classification of microarray data using random forest. *BMC Bioinformatics*, **7**, doi:10.1186/1471210573

See Also

varSelRF, randomForest, importance

Examples

```
x <- matrix(rnorm(25 * 30), ncol = 30)
x[1:10, 1:2] <- x[1:10, 1:2] + 2
cl <- factor(c(rep("A", 10), rep("B", 15)))

rf.vs1 <- varSelRF(x, cl, ntree = 200, ntreeIterat = 100,
                   vars.drop.frac = 0.2)
rf.vs1
plot(rf.vs1)
```

---

plot.varSelRFBoot   *plot a varSelRFBoot object*

Description

Plots of out-of-bag predictions and OOB error vs. number of variables.

Usage

```
## S3 method for class 'varSelRFBoot'
plot(x,  oobProb = TRUE,
                  oobProbBoxPlot = FALSE,
                  ErrorNum = TRUE,
                  subject.names = NULL,
                  class.to.plot = NULL,...)
```

Arguments

x	An object of class varSelRFBoot, such as returned by function varSelRFBoot.
oobProb	If TRUE plot (average) out-of-bag predictions. See prob.predictions in varSelRFBoot for more details about the out-of-bag predictions.
oobProbBoxPlot	If TRUE plot a box-plot of out-of-bag predictions.
ErrorNum	If TRUE plot OOB error (as returned by random forest) vs. the number of variables.
subject.names	If not NULL, a vector, of the same length as the number of cases (samples or subjects) with IDs for the cases/samples/subjects, that will be shown to the left of the average out-of-bag prediction.
class.to.plot	If not NULL, an integer or a vector of integers. These integers are those class levels for which out-of-bag predictions plots will be returned.
...	Not used.

Value

This function is only used for its side effects of producing plots.

Warning

The OOB Error rate is biased down (and can be severely biased down) because we do (potentially many) rounds of reducing the set of predictor variables until we minimize this OOB error rate. Note, however, that this is NOT the error rate reported as the estimate of the error rate for the procedure (for which we use the .632+ bootstrap rule).

Note

When plotting the out-of-bag predictions, we show one plot for each class. This is an overkill for two-class problems, but not necessarily for problems with more than two classes. Use class.to.plot to plot only those classes that interest you.

Author(s)

Ramon Diaz-Uriarte <rdiaz02@gmail.com>

References

Breiman, L. (2001) Random forests. *Machine Learning*, **45**, 5–32.

Diaz-Uriarte, R. , Alvarez de Andres, S. (2006) Gene selection and classification of microarray data using random forest. *BMC Bioinformatics*, **7**, doi:10.1186/1471210573

Efron, B. & Tibshirani, R. J. (1997) Improvements on cross-validation: the .632+ bootstrap method. *J. American Statistical Association*, **92**, 548–560.

See Also

randomForest, varSelRF, summary.varSelRFBoot, varSelRFBoot

Examples

```
## Not run:
## This is a small example, but can take some time.

x <- matrix(rnorm(25 * 30), ncol = 30)
x[1:10, 1:2] <- x[1:10, 1:2] + 2
cl <- factor(c(rep("A", 10), rep("B", 15)))

rf.vs1 <- varSelRF(x, cl, ntree = 200, ntreeIterat = 100,
                   vars.drop.frac = 0.2)
rf.vsb <- varSelRFBoot(x, cl,
                       bootnumber = 10,
                       usingCluster = FALSE,
                       srf = rf.vs1)
rf.vsb
summary(rf.vsb)
plot(rf.vsb)

## End(Not run)
```

---

randomVarImpsRF   *Variable importances from random forest on permuted class labels*

Description

Return variable importances from random forests fitted to data sets like the original except class labels have been randomly permuted.

Usage

```
randomVarImpsRF(xdata, Class, forest, numrandom = 100,
                whichImp = "impsUnscaled", usingCluster = TRUE,
                TheCluster = NULL, ...)
```

Arguments

xdata	A data frame or matrix, with subjects/cases in rows and variables in columns. NAs not allowed.
Class	The dependent variable; must be a factor.
forest	A previously fitted random forest (see randomForest).
numrandom	The number of random permutations of the class labels.
whichImp	A vector of one or more of impsUnscaled, impsScaled, impsGini, that correspond, respectively, to the (unscaled) mean decrease in accuracy, the scaled mean decrease in accuracy, and the Gini index. See below and randomForest, importance and the references for further explanations of the measures of variable importance.
usingCluster	If TRUE use a cluster to parallelize the calculations.
TheCluster	The name of the cluster, if one is used.
...	Not used.

Details

The measure of variable importance most often used is based on the decrease of classification accuracy when values of a variable in a node of a tree are permuted randomly (see references); we use the unscaled version —see our paper and supplementary material. Note that, by default, importance returns the scaled version.

Value

An object of class randomVarImpsRF, which is a list with one to three named components. The name of each component corresponds to the types of variable importance measures selected (i.e., impsUnscaled, impsScaled, impsGini).

Each component is a matrix, of dimensions number of variables by numrandom; each element (i,j) of this matrix is the variable importance for variable i and random permutation j.

Author(s)

Ramon Diaz-Uriarte <rdiaz02@gmail.com>

References

Breiman, L. (2001) Random forests. *Machine Learning*, **45**, 5–32.

Diaz-Uriarte, R. , Alvarez de Andres, S. (2006) Gene selection and classification of microarray data using random forest. *BMC Bioinformatics*, **7**, doi:10.1186/1471210573

Svetnik, V., Liaw, A. , Tong, C & Wang, T. (2004) Application of Breiman's random forest to modeling structure-activity relationships of pharmaceutical molecules. Pp. 334-343 in *F. Roli, J. Kittler, and T. Windeatt* (eds.). *Multiple Classier Systems, Fifth International Workshop*, MCS 2004, Proceedings, 9-11 June 2004, Cagliari, Italy. Lecture Notes in Computer Science, vol. 3077. Berlin: Springer.

See Also

randomForest, varSelRF, varSelRFBoot, varSelImpSpecRF, randomVarImpsRFplot

Examples

```
x <- matrix(rnorm(45 * 30), ncol = 30)
x[1:20, 1:2] <- x[1:20, 1:2] + 2
cl <- factor(c(rep("A", 20), rep("B", 25)))

rf <- randomForest(x, cl, ntree = 200, importance = TRUE)
rf.rvi <- randomVarImpsRF(x, cl,
                          rf,
                          numrandom = 20,
                          usingCluster = FALSE)

randomVarImpsRFplot(rf.rvi, rf)
```

---

randomVarImpsRFplot   *Plot random random variable importances*

Description

Plot variable importances from random permutations of class labels and the variable importances from the original data set.

Usage

```
randomVarImpsRFplot(randomImportances, forest,
                    whichImp = "impsUnscaled", nvars = NULL,
                    show.var.names = FALSE, vars.highlight = NULL,
                    main = NULL, screeRandom = TRUE,
                    lwdBlack = 1.5,
```

```
                              lwdRed = 2,
                              lwdLightblue = 1,
                              cexPoint = 1,
                              overlayTrue = FALSE,
                              xlab = NULL,
                              ylab = NULL, ...)
```

Arguments

randomImportances

A list with a structure such as the object return by randomVarImpsRF

.

forest   A random forest fitted to the original data. This forest must have been fitted with
                  importances = TRUE.

whichImp   The importance measue to use. One (only one) of impsUnscaled, impsScaled,
                  impsGini, that correspond, respectively, to the (unscaled) mean decrease in ac-
                  curacy, the scaled mean decrease in accuracy, and the Gini index. See below and
                  randomForest, importance and the references for further explanations of the
                  measures of variable importance.

nvars   If NULL will show the plot for the complete range of variables. If an integer,
                  will plot only the most important nvars.

show.var.names   If TRUE, show the variable names in the plot. Unless you are plotting few
                  variables, it probably won't be of any use.

vars.highlight   A vector indicating the variables to highlight in the plot with a vertical blue seg-
                  ment. You need to pass here a vector of variable names, not variable positions.

main   The title for the plot.

screeRandom   If TRUE, order all the variable importances (i.e., those from both the original
                  and the permuted class labels data sets) from largest to smallest before plotting.
                  The plot will thus resemble a usual "scree plot".

lwdBlack   The width of the line to use for the importances from the original data set.

lwdRed   The width of the line to use for the average of the importances for the permuted
                  data sets.

lwdLightblue   The width of the line for the importances for the individual permuted data sets.

cexPoint   cex argument for the points for the importances from the original data set.

overlayTrue   If TRUE, the variable importance from the original data set will be plotted last,
                  so you can see it even if buried in the middle of many gree lines; can be of help
                  when the plot does not allow you to see the black line.

xlab   The title for the x-axis (see xlab).

ylab   The title for the y-axis (see ylab).

...   Additional arguments to plot.

Value

Only used for its side effects of producing plots. In particular, you will see lines of three colors:

black	Connects the variable importances from the original simulated data.
green	Connect the variable importances from the data sets with permuted class labels; there will be as many lines as numrandom where used when [randomVarImpsRF](#) was called to obtain the random importances.
red	Connects the average of the importances from the permuted data sets.

Additionally, if you used a valid set of values for vars.highlight, these will be shown with a vertical blue segment.

Note

These plots resemble the scree plots commonly used with principal component analysis, and the actual choice of colors was taken from the importance spectrum plots of *Friedman & Meulman*.

Author(s)

Ramon Diaz-Uriarte <rdiaz02@gmail.com>

References

Breiman, L. (2001) Random forests. *Machine Learning*, **45**, 5–32.

Diaz-Uriarte, R. , Alvarez de Andres, S. (2006) Gene selection and classification of microarray data using random forest. *BMC Bioinformatics*, **7**, [doi:10.1186/1471210573](#)

Friedman, J., Meulman, J. (2005) Clustering objects on subsets of attributes (with discussion). *J. Royal Statistical Society, Series B*, **66**, 815–850.

See Also

[randomForest](#), [varSelRF](#), [varSelRFBoot](#), [varSelImpSpecRF](#), [randomVarImpsRF](#)

Examples

```
x <- matrix(rnorm(45 * 30), ncol = 30)
x[1:20, 1:2] <- x[1:20, 1:2] + 2
colnames(x) <- paste0("V", seq.int(ncol(x)))
cl <- factor(c(rep("A", 20), rep("B", 25)))

rf <- randomForest(x, cl, ntree = 200, importance = TRUE)
rf.rvi <- randomVarImpsRF(x, cl,
                          rf,
                          numrandom = 20,
                          usingCluster = FALSE)

randomVarImpsRFplot(rf.rvi, rf)
op <- par(las = 2)
randomVarImpsRFplot(rf.rvi, rf, show.var.names = TRUE)
par(op)
```

```
## Not run:
## identical, but using a cluster
## make a small cluster, for the sake of illustration
psockCL <- makeCluster(2, "PSOCK")
clusterSetRNGStream(psockCL, iseed = 789)
clusterEvalQ(psockCL, library(varSelRF))

rf.rvi <- randomVarImpsRF(x, cl,
                          rf,
                          numrandom = 20,
                          usingCluster = TRUE,
                          TheCluster = psockCL)

randomVarImpsRFplot(rf.rvi, rf)
stopCluster(psockCL)

## End(Not run)
```

---

selProbPlot   *Selection probability plot for variable importance from random forests*

Description

Plot, for the top ranked k variables from the original sample, the probability that each of these variables is included among the top ranked k genes from the bootstrap samples.

Usage

```
selProbPlot(object, k = c(20, 100),
            color = TRUE,
            legend = FALSE,
            xlegend = 68,
            ylegend = 0.93,
            cexlegend = 1.4,
            main = NULL,
            xlab = "Rank of gene",
            ylab = "Selection probability",
            pch = 19, ...)
```

Arguments

object   An object of class varSelRFBoot such as returned by the [varSelRFBoot](#) function.

k   A two-component vector with the k-th upper variables for which you want the plots.

color	If TRUE a color plot; if FALSE, black and white.
legend	If TRUE, show a legend.
xlegend	The x-coordinate for the legend.
ylegend	The y-coordinate for the legend.
cexlegend	The cex argument for the legend.
main	main for the plot.
xlab	xlab for the plot.
ylab	ylab for the plot.
pch	pch for the plot.
...	Additional arguments to plot.

Details

Pepe et al., 2003 suggested the use of selection probability plots to evaluate the stability and confidence on our selection of "relevant genes." This paper also presents several more sophisticated ideas not implemented here.

Value

Used for its side effects of producing a plot. In a single plot show the "selection probability plot" for the upper (largest variable importance) kth variables. By default, show the upper 20 and the upper 100 colored blue and red respectively.

Note

This function is in very rudimentary shape and could be used for more general types of data. I wrote specifically to produce Fig.\ 4 of the paper.

Author(s)

Ramon Diaz-Uriarte <rdiaz02@gmail.com>

References

Breiman, L. (2001) Random forests. *Machine Learning*, **45**, 5–32.

Diaz-Uriarte, R. , Alvarez de Andres, S. (2006) Gene selection and classification of microarray data using random forest. *BMC Bioinformatics*, **7**, doi:10.1186/1471210573

Pepe, M. S., Longton, G., Anderson, G. L. & Schummer, M. (2003) Selecting differentially expressed genes from microarray experiments. *Biometrics*, **59**, 133–142.

Svetnik, V., Liaw, A. , Tong, C & Wang, T. (2004) Application of Breiman's random forest to modeling structure-activity relationships of pharmaceutical molecules. Pp. 334-343 in *F. Roli, J. Kittler, and T. Windeatt* (eds.). *Multiple Classier Systems, Fifth International Workshop*, MCS 2004, Proceedings, 9-11 June 2004, Cagliari, Italy. Lecture Notes in Computer Science, vol. 3077. Berlin: Springer.

See Also

randomForest, varSelRF, varSelRFBoot, randomVarImpsRFplot, randomVarImpsRF

Examples

```
## This is a small example, but can take some time.

x <- matrix(rnorm(25 * 15), ncol = 15)
x[1:10, 1:2] <- x[1:10, 1:2] + 2
cl <- factor(c(rep("A", 10), rep("B", 15)))

rf.vs1 <- varSelRF(x, cl, ntree = 200, ntreeIterat = 100,
                   vars.drop.frac = 0.2)
rf.vsb <- varSelRFBoot(x, cl,
                       bootnumber = 10,
                       usingCluster = FALSE,
                       srf = rf.vs1)
selProbPlot(rf.vsb, k = c(5, 10), legend = TRUE,
            xlegend = 8, ylegend = 0.8)
```

---

summary.varSelRFBoot   *Summary of a varSelRFBoot object*

Description

Returns error rate and stability measures of a varSelRFBoot object.

Usage

```
## S3 method for class 'varSelRFBoot'
summary(object, return.model.freqs = FALSE,
                     return.class.probs = TRUE,
                     return.var.freqs.b.models = TRUE, ...)
```

Arguments

object	An object of class varSelRFBoot, as returned from varSelRFBoot.
return.model.freqs	
	If TRUE return a table with the frequencies of the final "models" (sets of selected variables) over all bootstrap replications.
return.class.probs	
	If TRUE return average class probabilities for each sample based on the out-of-bag probabilites (see varSelRFBoot, the prob.predictions component).
return.var.freqs.b.models	
	If TRUE return the frequencies of all variables selected from the bootstrap replicates.
...	Not used.

Value

If `return.class.probs = TRUE` a matrix with the average class probabilities for each sample based on the out-of-bag probabilites.

Regardless of that setting, print out several summaries:

`Summaries related to the "simplified" random forest on the original data`

Such as the number and identity of the variables selected.

`Summaries related to the error rate estimate`

Such as the .632+ estimate, and some of its components

`Summaries related to the stability (uniqueness) of the results obtained`

Such as the frequency of the selected variables in the bootstrap runs, the frequency of the selected variables in the boostrap runs that are also among the variables selected from the complete run, the overlap of the bootstrap forests with the forest from the original data set (see `varSelRF` for the definition of overlap), and (optionally) the frequency of the "models", where a model is the set of variables selected in any particular run.

Author(s)

Ramon Diaz-Uriarte <rdiaz02@gmail.com>

References

Breiman, L. (2001) Random forests. *Machine Learning*, **45**, 5–32.

Diaz-Uriarte, R. , Alvarez de Andres, S. (2006) Gene selection and classification of microarray data using random forest. *BMC Bioinformatics*, **7**, doi:10.1186/1471210573

Efron, B. & Tibshirani, R. J. (1997) Improvements on cross-validation: the .632+ bootstrap method. *J. American Statistical Association*, **92**, 548–560.

See Also

randomForest, varSelRF, varSelRFBoot, plot.varSelRFBoot,

Examples

```
## Not run:
## This is a small example, but can take some time.

x <- matrix(rnorm(25 * 30), ncol = 30)
x[1:10, 1:2] <- x[1:10, 1:2] + 2
cl <- factor(c(rep("A", 10), rep("B", 15)))

rf.vs1 <- varSelRF(x, cl, ntree = 200, ntreeIterat = 100,
                   vars.drop.frac = 0.2)
rf.vsb <- varSelRFBoot(x, cl,
                       bootnumber = 10,
                       usingCluster = FALSE,
```

```
                                        srf = rf.vs1)
rf.vsb
summary(rf.vsb)
plot(rf.vsb)

## End(Not run)
```

---

varSelImpSpecRF   *Variable selection using the "importance spectrum"*

Description

Perform variable selection based on a simple heuristic using the importance spectrum of the original data compared to the importance spectra from the same data with the class labels randomly permuted.

Usage

```
varSelImpSpecRF(forest, xdata = NULL, Class = NULL,
                randomImps = NULL,
                threshold = 0.1,
                numrandom = 20,
                whichImp = "impsUnscaled",
                usingCluster = TRUE,
                TheCluster = NULL, ...)
```

Arguments

forest	A previously fitted random forest (see randomForest).
xdata	A data frame or matrix, with subjects/cases in rows and variables in columns. NAs not allowed.
Class	The dependent variable; must be a factor.
randomImps	A list with a structure such as the object return by randomVarImpsRF .
threshold	The threshold for the selection of variables. See details.
numrandom	The number of random permutations of the class labels.
whichImp	One of impsUnscaled, impsScaled, impsGini, that correspond, respectively, to the (unscaled) mean decrease in accuracy, the scaled mean decrease in accuracy, and the Gini index. See below and randomForest, importance and the references for further explanations of the measures of variable importance.
usingCluster	If TRUE use a cluster to parallelize the calculations.
TheCluster	The name of the cluster, if one is used.
...	Not used.

Details

You can either pass as arguments a valid object for randomImps, obtained from a previous call to randomVarImpsRF OR you can pass a covariate data frame and a dependent variable, and these will be used to obtain the random importances. The former is preferred for normal use, because this function will not returned the computed random variable importances, and this computation can be lengthy. If you pass both randomImps, xdata, and Class, randomImps will be used.

To select variables, start by ordering from largest ($i = 1$) to smallest ($i = p$, where p is the number of variables), the variable importances from the original data and from each of the data sets with permuted class labels. (So the ordering is done in each data set independently). Compute q_i, the $1 - threshold$ quantile of the ordered variable importances from the permuted data at ordered postion i. Then, starting from $i = 1$, let i_a be the first i for which the variable importance from the original data is smaller than q_i. Select all variables from $i = 1$ to $i = i_a - 1$.

Value

A vector with the names of the selected variables, ordered by decreasing importance.

Note

The name of this function is related to the idea of "importance spectrum plot", which is the term that *Friedman & Meulman, 2005* use in their paper.

Author(s)

Ramon Diaz-Uriarte <rdiaz02@gmail.com>

References

Breiman, L. (2001) Random forests. *Machine Learning*, **45**, 5–32.

Diaz-Uriarte, R. , Alvarez de Andres, S. (2006) Gene selection and classification of microarray data using random forest. *BMC Bioinformatics*, **7**, doi:10.1186/1471210573

Friedman, J., Meulman, J. (2005) Clustering objects on subsets of attributes (with discussion). *J. Royal Statistical Society, Series B*, **66**, 815–850.

See Also

randomForest, varSelRF, varSelRFBoot, randomVarImpsRFplot, randomVarImpsRF

Examples

```
x <- matrix(rnorm(45 * 30), ncol = 30)
x[1:20, 1:2] <- x[1:20, 1:2] + 2
cl <- factor(c(rep("A", 20), rep("B", 25)))

rf <- randomForest(x, cl, ntree = 200, importance = TRUE)
rf.rvi <- randomVarImpsRF(x, cl,
                          rf,
                          numrandom = 20,
                          usingCluster = FALSE)
```

```
varSelImpSpecRF(rf, randomImps = rf.rvi)



## Not run:
## Identical, but using a cluster
psockCL <- makeCluster(2, "PSOCK")
clusterSetRNGStream(psockCL, iseed = 456)
clusterEvalQ(psockCL, library(varSelRF))

rf.rvi <- randomVarImpsRF(x, cl,
                          rf,
                          numrandom = 20,
                          usingCluster = TRUE,
                          TheCluster = psockCL)
varSelImpSpecRF(rf, randomImps = rf.rvi)
stopCluster(psockCL)


## End(Not run)
```

---

varSelRF   *Variable selection from random forests using OOB error*

Description

Using the OOB error as minimization criterion, carry out variable elimination from random forest, by successively eliminating the least important variables (with importance as returned from random forest).

Usage

```
varSelRF(xdata, Class, c.sd = 1, mtryFactor = 1, ntree = 5000,
         ntreeIterat = 2000, vars.drop.num = NULL, vars.drop.frac = 0.2,
         whole.range = TRUE, recompute.var.imp = FALSE, verbose = FALSE,
         returnFirstForest = TRUE, fitted.rf = NULL, keep.forest = FALSE)
```

Arguments

xdata
: A data frame or matrix, with subjects/cases in rows and variables in columns. NAs not allowed.

Class
: The dependent variable; must be a factor.

c.sd
: The factor that multiplies the sd. to decide on stopping the tierations or choosing the final solution. See reference for details.

mtryFactor	The multiplication factor of $\sqrt{number.of.variables}$ for the number of variables to use for the ntry argument of randomForest.
ntree	The number of trees to use for the first forest; same as ntree for randomForest.
ntreeIterat	The number of trees to use (ntree of randomForest) for all additional forests.
vars.drop.num	The number of variables to exclude at each iteration.
vars.drop.frac	The fraction of variables, from those in the previous forest, to exclude at each iteration.
whole.range	If TRUE continue dropping variables until a forest with only two variables is built, and choose the best model from the complete series of models. If FALSE, stop the iterations if the current OOB error becomes larger than the initial OOB error (plus c.sd*OOB standard error) or if the current OOB error becoems larger than the previous OOB error (plus c.sd*OOB standard error).
recompute.var.imp	
	If TRUE recompute variable importances at each new iteration.
verbose	Give more information about what is being done.
returnFirstForest	
	If TRUE the random forest from the complete set of variables is returned.
fitted.rf	An (optional) object of class randomForest previously fitted. In this case, the ntree and mtryFactor arguments are obtained from the fitted object, not the arguments to this function.
keep.forest	Same argument as in randomForest function. If the forest is kept, it will be returned as part of the "rf.model" component of the output. Beware that setting this to TRUE can lead to very large memory consumption.

Details

With the default parameters, we examine all forest that result from eliminating, iteratively, a fraction, vars.drop.frac, of the least important variables used in the previous iteration. By default, vars.frac.drop = 0.2 which allows for relatively fast operation, is coherent with the idea of an "aggressive variable selection" approach, and increases the resolution as the number of variables considered becomes smaller. By default, we do not recalculate variable importances at each step (recompute.var.imp = FALSE) as *Svetnik et al. 2004* mention severe overfitting resulting from recalculating variable importances. After fitting all forests, we examine the OOB error rates from all the fitted random forests. We choose the solution with the smallest number of genes whose error rate is within c.sd standard errors of the minimum error rate of all forests. (The standard error is calculated using the expression for a biomial error count [$\sqrt{p(1-p) * 1/N}$]). Setting c.sd = 0 is the same as selecting the set of genes that leads to the smallest error rate. Setting c.sd = 1 is similar to the common "1 s.e. rule", used in the classification trees literature; this strategy can lead to solutions with fewer genes than selecting the solution with the smallest error rate, while achieving an error rate that is not different, within sampling error, from the "best solution".

The use of ntree = 5000 and ntreeIterat = 2000 is discussed in longer detail in the references. Essentially, more iterations rarely seem to lead (with 9 different microarray data sets) to improved solutions.

The measure of variable importance used is based on the decrease of classification accuracy when values of a variable in a node of a tree are permuted randomly (see references); we use the unscaled version —see our paper and supplementary material.

Value

An object of class "varSelRF": a list with components:

selec.history   A data frame where the selection history is stored. The components are:

> **Number.Variables** The number of variables examined.
>
> **Vars.in.Forest** The actual variables that were in the forest at that stage.
>
> **OOB** Out of bag error rate.
>
> **sd.OOB** Standard deviation of the error rate.

rf.model   The final, selected, random forest (only if whole.range = FALSE). (If you set whole.range = TRUE, the final model always contains exactly two variables. This is unlikely to be the forest that interests you).

selected.vars   The variables finally selected.

selected.model   Same as above, but ordered alphabetically and concatenated with a "+" for easier display.

best.model.nvars
          The number of variables in the finally selected model.

initialImportance
          The importances of variables, before any variable deletion.

initialOrderedImportances
          Same as above but ordered in by decreasing importance.

ntree   The ntree argument.

ntreeIterat   The ntreeIterat argument.

mtryFactor   The mtryFactor argument.

firstForest   The first forest (before any variable selection) fitted.

Author(s)

Ramon Diaz-Uriarte <rdiaz02@gmail.com>

References

Breiman, L. (2001) Random forests. *Machine Learning*, **45**, 5–32.

Diaz-Uriarte, R. , Alvarez de Andres, S. (2006) Gene selection and classification of microarray data using random forest. *BMC Bioinformatics*, **7**, doi:10.1186/1471210573

Svetnik, V., Liaw, A. , Tong, C & Wang, T. (2004) Application of Breiman's random forest to modeling structure-activity relationships of pharmaceutical molecules. Pp. 334-343 in *F. Roli, J. Kittler, and T. Windeatt* (eds.). *Multiple Classier Systems, Fifth International Workshop*, MCS 2004, Proceedings, 9-11 June 2004, Cagliari, Italy. Lecture Notes in Computer Science, vol. 3077. Berlin: Springer.

See Also

randomForest, plot.varSelRF, varSelRFBoot

Examples

```
set.seed(1)
x <- matrix(rnorm(25 * 30), ncol = 30)
colnames(x) <- paste("v", 1:30, sep = "")
x[1:10, 1:2] <- x[1:10, 1:2] + 1
x[1:4, 5] <- x[1:4, 5] - 1.5
x[5:10, 8] <- x[5:10, 8] + 1.4

cl <- factor(c(rep("A", 10), rep("B", 15)))
rf.vs1 <- varSelRF(x, cl, ntree = 500, ntreeIterat = 300,
                   vars.drop.frac = 0.2)
rf.vs1
plot(rf.vs1)


## Note you can use tiny vars.drop.frac
## though you'll rarely want this
rf.vs1tiny <- varSelRF(x, cl, ntree = 500, ntreeIterat = 300,
                   vars.drop.frac = 0.01)

#### Using the final, fitted model to predict other data

## Simulate new data
set.seed(2)
x.new <- matrix(rnorm(25 * 30), ncol = 30)
colnames(x.new) <- paste("v", 1:30, sep = "")
x.new[1:10, 1:2] <- x.new[1:10, 1:2] + 1
x.new[1:10, 5] <- x.new[1:10, 5] - 0.5


## Fit with whole.range = FALSE and keep.forest = TRUE
set.seed(3)
rf.vs2 <- varSelRF(x, cl, ntree = 3000, ntreeIterat = 2000,
                   vars.drop.frac = 0.3, whole.range = FALSE,
                   keep.forest = TRUE)


## To obtain predictions from a data set, you must specify the
## same variables as those used in the final model

rf.vs2$selected.vars


predict(rf.vs2$rf.model,
        newdata = subset(x.new, select = rf.vs2$selected.vars))
predict(rf.vs2$rf.model,
        newdata = subset(x.new, select = rf.vs2$selected.vars),
        type = "prob")


## If you had not kept the forest (keep.forest) you could also try
randomForest(y = cl, x = subset(x, select = rf.vs2$selected.vars),
```

```
                ntree = rf.vs2$ntreeIterat,
                xtest = subset(x, select = rf.vs2$selected.vars))$test

## but here the forest is built new (with only the selected variables)
## so results need not be the same



## CAUTION: You will NOT want this (these are similar to resubstitution
##    predictions)

predict(rf.vs2$rf.model, newdata = subset(x, select = rf.vs2$selected.vars))

## nor these (read help of predict.randomForest for why these
## predictions are different from those from previous command)

predict(rf.vs2$rf.model)
```

---

varSelRFBoot   *Bootstrap the variable selection procedure in varSelRF*

Description

Use the bootstrap to estimate the prediction error rate (wuth the .632+ rule) and the stability of the
variable selection procedure implemented in [varSelRF](#).

Usage

```
varSelRFBoot(xdata, Class, c.sd = 1,
             mtryFactor = 1, ntree = 5000, ntreeIterat = 2000,
             vars.drop.frac = 0.2, bootnumber = 200,
             whole.range = TRUE,
             recompute.var.imp = FALSE,
             usingCluster = TRUE,
             TheCluster = NULL, srf = NULL, verbose = TRUE, ...)
```

Arguments

Most arguments are the same as for [varSelRFBoot](#).

xdata	A data frame or matrix, with subjects/cases in rows and variables in columns. NAs not allowed.
Class	The dependent variable; must be a factor.
c.sd	The factor that multiplies the sd. to decide on stopping the tierations or choosing the final solution. See reference for details.

mtryFactor	The multiplication factor of $\sqrt{number.of.variables}$ for the number of variables to use for the ntry argument of randomForest.
ntree	The number of trees to use for the first forest; same as ntree for randomForest.
ntreeIterat	The number of trees to use (ntree of randomForest) for all additional forests.
vars.drop.frac	The fraction of variables, from those in the previous forest, to exclude at each iteration.
whole.range	If TRUE continue dropping variables until a forest with only two variables is built, and choose the best model from the complete series of models. If FALSE, stop the iterations if the current OOB error becomes larger than the initial OOB error (plus c.sd*OOB standard error) or if the current OOB error becoems larger than the previous OOB error (plus c.sd*OOB standard error).

recompute.var.imp

If TRUE recompute variable importances at each new iteration.

bootnumber	The number of bootstrap samples to draw.
usingCluster	If TRUE use a cluster to parallelize the calculations.
TheCluster	The name of the cluster, if one is used.
srf	An object of class varSelRF. If used, the ntree and mtryFactor parameters are taken from this object, not from the arguments to this function. If used, it allows to skip carrying out a first iteration to build the random forest to the complete, original data set.
verbose	Give more information about what is being done.
...	Not used.

Details

If a cluster is used for the calculations, it will be used for the embarrisingly parallelizable task of building as many random forests as bootstrap samples.

Value

An object of class varSelRFBoot, which is a list with components:

number.of.bootsamples

The number of bootstrap replicates.

bootstrap.pred.error

The .632+ estimate of the prediction error.

leave.one.out.bootstrap

The leave-one-out estimate of the error rate (used when computing the .632+ estimate).

all.data.randomForest

A random forest built from all the data, but after the variable selection. Thus, beware because the OOB error rate is severely biased down.

all.data.vars	The variables selected in the run with all the data.
all.data.run	An object of class varSelRF; the one obtained from a run of varSelRF on the original, complete, data set. See [varSelRF]().

class.predictions

> The out-of-bag predictions from the bootstrap, of type "response".See `predict.randomForest`. This is an array, with dimensions number of cases by number of bootstrap replicates.

prob.predictions

> The out-of-bag predictions from the bootstrap, of type "class probability". See `predict.randomForest`. This is a 3-way array; the last dimension is the bootstrap replication; for each bootstrap replication, the 2D array has dimensions case by number of classes, and each value is the probability of belonging to that class.

number.of.vars A vector with the number of variables selected for each bootstrap sample.

overlap The "overlap" between the variables selected from the run in original sample and the variables returned from a bootstrap sample. Overlap between the sets of variables A and B is defined as

$$\frac{|variables.in.A \cap variables.in.B|}{\sqrt{|variables.in.A||variables.in.B|}}$$

> or size (cardinality) of intersection between the two sets / sqrt(product of size of each set).

all.vars.in.solutions

> A vector with all the genes selected in the runs on all the bootstrap samples. If the same gene is selected in several bootstrap runs, it appears multiple times in this vector.

all.solutions Each solutions is a character vector with all the variables in a particular solution concatenated by a "+". Thus, all.solutions is a vector, with length equal to `number.of.bootsamples`, of the solution from each bootstrap run.

Class The original class argument.

allBootRuns A list of length `number.of.bootsamples`. Each component of this list is an element of class `varSelRF` and stores the results from the runs on each bootstrap sample.

Note

The out-of-bag predictions stored in `class.predictions` and `prob.predictions` are NOT the OOB votes from random forest itself for a given run. These are predictions from the out-of-bag samples for each bootstrap replication. Thus, these are samples that have not been used at all in any of the variable selection procedures in the given bootstrap replication.

Author(s)

Ramon Diaz-Uriarte <rdiaz02@gmail.com>

References

Breiman, L. (2001) Random forests. *Machine Learning*, **45**, 5–32.

Diaz-Uriarte, R. , Alvarez de Andres, S. (2006) Gene selection and classification of microarray data using random forest. *BMC Bioinformatics*, **7**, doi:10.1186/1471210573

Efron, B. & Tibshirani, R. J. (1997) Improvements on cross-validation: the .632+ bootstrap method. *J. American Statistical Association*, **92**, 548–560.

Svetnik, V., Liaw, A. , Tong, C & Wang, T. (2004) Application of Breiman's random forest to modeling structure-activity relationships of pharmaceutical molecules. Pp. 334-343 in *F. Roli, J. Kittler, and T. Windeatt* (eds.). *Multiple Classier Systems, Fifth International Workshop*, MCS 2004, Proceedings, 9-11 June 2004, Cagliari, Italy. Lecture Notes in Computer Science, vol. 3077. Berlin: Springer.

See Also

[randomForest](), [varSelRF](), [summary.varSelRFBoot](), [plot.varSelRFBoot](),

Examples

```
## Not run:
## This is a small example, but can take some time.

## make a small cluster, for the sake of illustration
forkCL <- makeForkCluster(2)
clusterSetRNGStream(forkCL, iseed = 123)
clusterEvalQ(forkCL, library(varSelRF))


x <- matrix(rnorm(25 * 30), ncol = 30)
x[1:10, 1:2] <- x[1:10, 1:2] + 2
cl <- factor(c(rep("A", 10), rep("B", 15)))

rf.vs1 <- varSelRF(x, cl, ntree = 200, ntreeIterat = 100,
                   vars.drop.frac = 0.2)
rf.vsb <- varSelRFBoot(x, cl,
                       bootnumber = 10,
                       usingCluster = TRUE,
                       srf = rf.vs1,
                       TheCluster = forkCL)
rf.vsb
summary(rf.vsb)
plot(rf.vsb)
stopCluster(forkCL)

## End(Not run)
```

Index