Package 'VSURF'

January 20, 2025

Type Package

Title Variable Selection Using Random Forests

Version 1.2.0

Date 2022-12-15

Depends R (>= 4.2.0)

Description Three steps variable selection procedure based on random forests. Initially developed to handle high dimensional data (for which number of variables largely exceeds number of observations), the package is very versatile and can treat most dimensions of data, for regression and supervised classification problems. First step is dedicated to eliminate irrelevant variables from the dataset. Second step aims to select all variables related to the response for interpretation purpose. Third step refines the selection by eliminating redundancy in the set of variables selected by the second step, for prediction purpose.
Genuer, R. Poggi, J.-M. and Tuleau-Malot, C. (2015)
<https://journal.r-project.org/archive/2015-2/genuer-poggi-tuleaumalot.pdf>.

License GPL (>= 2)

LazyData true

URL https://github.com/robingenuer/VSURF

BugReports https://github.com/robingenuer/VSURF/issues

Imports doParallel, foreach, parallel, randomForest, rpart

Suggests testthat, ranger, Rborist

RoxygenNote 7.1.2

NeedsCompilation no

Author Robin Genuer [aut, cre],
Jean-Michel Poggi [aut],
Christine Tuleau-Malot [aut]

Maintainer Robin Genuer <Robin.Genuer@u-bordeaux.fr>

Repository CRAN

Date/Publication 2022-12-15 13:20:02 UTC

Contents

`plot.VSURF` *Plot of VSURF results*

Description

This function plots 4 graphs illustrating VSURF results.

Usage

```
## S3 method for class 'VSURF'
plot(
  x,
  step = "all",
  var.names = FALSE,
  imp.mean = TRUE,
  imp.sd = TRUE,
  nvar.imp.mean = length(x$imp.mean.dec),
  nvar.imp.sd = length(x$imp.sd.dec),
  nvar.interp = length(x$varselect.thres),
  nvar.pred = length(x$varselect.pred),
  ...
)

## S3 method for class 'VSURF_thres'
plot(
  x,
  var.names = FALSE,
  imp.mean = TRUE,
  imp.sd = TRUE,
  nvar.imp.mean = length(x$imp.mean.dec),
  nvar.imp.sd = length(x$imp.sd.dec),
  ...
```

```
)

## S3 method for class 'VSURF_interp'
plot(x, var.names = FALSE, nvar.interp = length(x$varselect.thres), ...)

## S3 method for class 'VSURF_pred'
plot(x, var.names = FALSE, nvar.pred = length(x$varselect.pred), ...)
```

Arguments

x	An object of class VSURF, VSURF_thres, VSURF_interp or VSURF_pred, which is the result of the VSURF function (or resp. VSURF_thres, VSURF_interp or VSURF_pred).
step	A character string indicating which step must be plotted (default is "all"). Available choices are "thres", "interp", "pred".
var.names	If FALSE (default) xticks are the numbering given by the sorting of VI mean, if TRUE they are the variables names.
imp.mean	If TRUE (default) VI mean is plotted, if FALSE it is not.
imp.sd	If TRUE (default) VI standard deviation is plotted, if FALSE it is not.
nvar.imp.mean	The number of variables to be kept for the VI mean plot.
nvar.imp.sd	The number of variables to be kept for the VI standard deviation plot.
nvar.interp	The number of variables to be kept for the "interp" plot.
nvar.pred	The number of variables to be kept for the "pred" plot.
...	Arguments to be passed to par (they will affect all plots) or to others methods of plot.

Details

The 2 graphs of the top row correspond to the "thresholding step" (and only these 2 graphs are plotted by the plot.VSURF_thres function). The top left graph plots the mean variable importance in decreasing order (black curve). The red horizontal line represent the value of the threshold. The top right graph plots the standard deviation of variable importance with variables ordered according to their mean variable importance in decreasing order (black curve). The green line represents the predictions given by a CART tree fitted to the black curve (the standard deviations). Finally, the dotted horizontal red line represents the minimum value of the CART predictions, which actually is the value of the threshold.

The bottom left graph corresponds to the "interpretation step" (and only this graph is plotted by the plot.VSURF_interp function). It plots the mean OOB error rate of embedded random forests models (from the one with only one variable as predictor, to the one with all variables kept after the "thresholding step"). The vertical red line indicates the retained model.

The bottom right graph corresponds to the "prediction step" (and only this graph is plotted by the plot.VSURF_pred function). It plots the mean OOB error rate of embedded random forests models (the difference, here, being that variables are added to the model in a step-wise manner). The retained model is the final one.

Author(s)

Robin Genuer, Jean-Michel Poggi and Christine Tuleau-Malot

References

Genuer, R. and Poggi, J.M. and Tuleau-Malot, C. (2010), *Variable selection using random forests*, Pattern Recognition Letters 31(14), 2225-2236

Genuer, R. and Poggi, J.M. and Tuleau-Malot, C. (2015), *VSURF: An R Package for Variable Selection Using Random Forests*, The R Journal 7(2):19-33

See Also

VSURF, summary.VSURF

Examples

```
## Not run:
data(iris)
iris.vsurf <- VSURF(iris[,1:4], iris[,5])
plot(iris.vsurf)
plot(iris.vsurf, var.names=TRUE)
plot(iris.vsurf, step="thres")

# A more interesting example with toys data (see \code{\link{toys}})
# (a few minutes to execute) and intermediate functions
data(toys)
toys.vsurf <- VSURF(toys$x, toys$y)
plot(toys.vsurf)
plot(toys.vsurf, nvar.imp.mean = 50, nvar.imp.sd = 50)
toys.thres <- VSURF_thres(toys$x, toys$y)
plot(toys.thres)
plot(toys.thres, nvar.imp.mean = 70, imp.sd = FALSE)
toys.interp <- VSURF_interp(toys$x, toys$y, vars = toys.thres$varselect.thres)
plot(toys.interp, var.names = TRUE)
toys.pred <- VSURF_pred(toys$x, toys$y, err.interp = toys.interp$err.interp,
                        varselect.interp = toys.interp$varselect.interp)
plot(toys.pred, var.names = TRUE)

## End(Not run)
```

PM10 *Real-world data on PM10 pollution in Rouen area, France*

Description

These data are TEOM (Tapered Element Oscillating Microbalance) PM10 concentrations from 2004 to 2006 (1096 days) measured by Air Normand, and the associated weather data provided by Meteo France, the French national meteorological service, using six different monitoring sites.

Usage

```
ail

gcm

gui

hri

jus

rep
```

Format

Each object is a data frame.

The description of the 18 variables is the following (note that for `gcm` station, only the pollutant SO2 is available in addition to PM10, and for `ail` station, there is no other pollutant in addition to PM10):

PM10 Daily mean concentration of PM10, in $\mu g/m^3$

NO, NO2, SO2 Daily mean concentration of NO, NO2 , SO2, in $\mu g/m^3$

T.min, T.max, T.moy Daily minimum, maximum and mean temperature, in degree Celsius

DV.maxvv, DV.dom Daily maximum speed and dominant wind direction, in degree (for wind direction, 0 degree corresponds to north)

VV.max, VV.moy Daily maximum and mean wind speed, in m/s

PL.som Daily rainfall, in mm

HR.min, HR.max, HR.moy Daily minimum, maximum and mean relative humidity, in %

PA.moy Daily mean air pressure, in hPa

GTrouen, GTlehavre Daily temperature gradient, in degree Celsius

An object of class `data.frame` with 1096 rows and 15 columns.

An object of class `data.frame` with 1096 rows and 16 columns.

An object of class `data.frame` with 1096 rows and 18 columns.

An object of class `data.frame` with 1096 rows and 18 columns.

An object of class `data.frame` with 1096 rows and 18 columns.

An object of class `data.frame` with 1096 rows and 18 columns.

Details

Six different monitoring stations of the Rouen (Haute Normandie, France) area are considered. The urban station `jus`, the traffic station `gui`, the second most polluted in the region, and `gcm` which is located in an industrial area. In Le Havre, are considered the stations `rep` (the most polluted in the region) and `hri` located at the seaside. Lastly, the station `ail` near Dieppe, because it is rural and

coastal, and a priori hardly influenced by social and industrial activity. Grouping by categories: `jus` and `hri` are background urban monitoring sites, `gui` and `rep` are urban sites close to traffic, `gcm` is industrial and `ail` is rural.

Source

F.-X. Jollois, J.-M. Poggi, B. Portier, *Three non-linear statistical methods to analyze PM10 pollution in Rouen area*. CSBIGS 3(1): 1-17, 2009

`predict.VSURF` *Predict method for VSURF object*

Description

This function predicts new data with random forests, using variables selected by VSURF only.

Usage

```
## S3 method for class 'VSURF'
predict(object, newdata, step = c("interp", "pred"), ...)
```

Arguments

`object`	An object of class `VSURF`, which is the result of the `VSURF` function.
`newdata`	A data frame or matrix containing new data. (Note: If not given, the out-of-bag predictions of the randomForest object is returned.)
`step`	A character string indicating which variable set must be used to train the `randomForest` object (default is c("interp", "pred")). Available choices are "thres", "interp", "pred".
`...`	further parameters passed to `randomForest` or `predict.randomForest` functions (depending on their names).

Details

This method applies for a VSURF object. VSURF selects two sets of variables during its two last steps. For each set of variables, a random forest object is created, by running `randomForest` on training data using this set of variables only. Then the `predict.randomForest` function is used to predict new data.

Value

If only one step is indicated in `step`, a vector of predicted values.

If two or more steps are indicated in `step`, a data frame of predicted values (each column corresponding to a variable set).

Author(s)

Robin Genuer, Jean-Michel Poggi and Christine Tuleau-Malot

References

Genuer, R. and Poggi, J.M. and Tuleau-Malot, C. (2010), *Variable selection using random forests*, Pattern Recognition Letters 31(14), 2225-2236

Genuer, R. and Poggi, J.M. and Tuleau-Malot, C. (2015), *VSURF: An R Package for Variable Selection Using Random Forests*, The R Journal 7(2):19-33

See Also

VSURF

Examples

```
## Not run:
data(iris)
iris.learn <- sample(1:nrow(iris), nrow(iris)/2)
iris.vsurf <- VSURF(iris[iris.learn, 1:4], iris[iris.learn, 5], ntree = 100, nfor.thres = 20,
                    nfor.interp = 10, nfor.pred = 10)
iris.predictions <- predict(iris.vsurf, newdata = iris[-iris.learn, 1:4])

# A more interesting example with toys data (see \code{\link{toys}})
# (a few minutes to execute)
data(toys)
toys.learn <- 1:(nrow(toys$x) / 2)
toys.vsurf <- VSURF(toys$x[toys.learn, ], toys$y[toys.learn])
toys.predictions <- predict(toys.vsurf, newdata = toys$x[-toys.learn, ])
## End(Not run)
```

---

print.VSURF *Print of VSURF results*

Description

This function display a small description of VSURF results

Usage

```
## S3 method for class 'VSURF'
print(x, ...)
```

Arguments

x	An object of class VSURF, which is the result of the VSURF function.
...	Not used.

Author(s)

Robin Genuer, Jean-Michel Poggi and Christine Tuleau-Malot

References

Genuer, R. and Poggi, J.M. and Tuleau-Malot, C. (2010), *Variable selection using random forests*, Pattern Recognition Letters 31(14), 2225-2236

Genuer, R. and Poggi, J.M. and Tuleau-Malot, C. (2015), *VSURF: An R Package for Variable Selection Using Random Forests*, The R Journal 7(2):19-33

See Also

VSURF, plot.VSURF, summary.VSURF

Examples

```
## Not run:
data(iris)
iris.vsurf <- VSURF(iris[,1:4], iris[,5], ntree = 100, nfor.thres = 20,
                    nfor.interp = 10, nfor.pred = 10)
iris.vsurf

## End(Not run)
```

---

summary.VSURF *Summary of VSURF results*

Description

This function displays a summary of VSURF results

Usage

```
## S3 method for class 'VSURF'
summary(object, ...)
```

Arguments

object	An object of class VSURF, which is the result of the VSURF function.
...	Not used.

Details

This function prints the total computation time of VSURF. It also gives the number of selected variables (and the computation time) at each step of VSURF. In addition, it gives the number of cores and the type of cluster if the parallel version of VSURF was used.

Author(s)

Robin Genuer, Jean-Michel Poggi and Christine Tuleau-Malot

References

Genuer, R. and Poggi, J.M. and Tuleau-Malot, C. (2010), *Variable selection using random forests*, Pattern Recognition Letters 31(14), 2225-2236

Genuer, R. and Poggi, J.M. and Tuleau-Malot, C. (2015), *VSURF: An R Package for Variable Selection Using Random Forests*, The R Journal 7(2):19-33

See Also

VSURF, plot.VSURF

Examples

```
## Not run:
data(iris)
iris.vsurf <- VSURF(iris[,1:4], iris[,5], ntree = 100, nfor.thres = 20,
                    nfor.interp = 10, nfor.pred = 10)
summary(iris.vsurf)

# A more interesting example with toys data (see \code{\link{toys}})
# (a few minutes to execute)
data(toys)
toys.vsurf <- VSURF(toys$x, toys$y)
summary(toys.vsurf)
## End(Not run)
```

---

toys *A simulated dataset called toys data*

Description

toys is a simple simulated dataset of a binary classification problem, introduced by Weston et.al..

Format

The format is a list of 2 components:

x a dataframe containing input variables: with 100 obs. of 200 variables

y output variable: a factor with 2 levels "-1" and "1"

Details

It is an equiprobable two class problem, Y belongs to {-1,1}, with six true variables, the others being some noise. The simulation model is defined through the conditional distribution of the X_i for Y=y:

- with probability 0.7, X^j ~ N(yj,1) for j=1,2,3 and X^j ~ N(0,1) for j=4,5,6 ;
- with probability 0.3, X^j ~ N(0,1) for j=1,2,3 and X^j ~ N(y(j-3),1) for j=4,5,6 ;
- the other variables are noise, X^j ~ N(0,1) for j=7,...,p.

After simulation, the obtained variables are finally standardized.

Source

Weston, J., Elisseff, A., Schoelkopf, B., Tipping, M. (2003), *Use of the zero norm with linear models and Kernel methods*, J. Machine Learn. Res. 3, 1439-1461

Examples

```
data(toys)
toys.rf <- randomForest::randomForest(toys$x, toys$y)
toys.rf

## Not run:
# VSURF applied for toys data:
# (a few minutes to execute)
data(toys)
toys.vsurf <- VSURF(toys$x, toys$y)
toys.vsurf

## End(Not run)
```

---

tune *Tuning of the thresholding and interpretation steps of VSURF*

Description

This function allows to tune the "thresholding" and "interpretation step" of VSURF, without rerunning all computations.

Usage

```
tune(x, ...)

## S3 method for class 'VSURF_thres'
tune(x, nmin = 1, ...)

## S3 method for class 'VSURF_interp'
tune(x, nsd = 1, ...)
```

Arguments

x	An object of class VSURF_thres or VSURF_interp, which is the result of the VSURF_thres or VSURF_interp function.
...	Not used.
nmin	Number of times the "minimum value" is multiplied to set threshold value. See details below.
nsd	Number of times the standard deviation of the minimum value of err.interp is multiplied. See details below.

Details

In VSURF_thres function, the actual threshold is performed like this: only variables with a mean VI larger than nmin * min.thres are kept. The function tune.VSURF_thres allows you to change the value of nmin (which multiply the estimated threshold value min.thres), without rerunning all computations. To get a softer threshold than default, choose a value of nmin less than 1, and to get a harder one, choose a value larger than 1.

In VSURF_interp function, the smallest model (and hence its corresponding variables) having a mean OOB error rate less than err.min + nsd * sd.min is selected. The function tune.VSURF_interp allows to change the value of nsd (which multiply the standard deviation of the minimum OOB error rate sd.min), without rerunning all computations. To get a larger model than default, choose a value of nsd less than 1, and to get a smaller one, choose a value larger than 1.

Value

An object with the same structure than the original output (from VSURF_thres or VSURF_interp).

Author(s)

Robin Genuer, Jean-Michel Poggi and Christine Tuleau-Malot

References

Genuer, R. and Poggi, J.M. and Tuleau-Malot, C. (2010), *Variable selection using random forests*, Pattern Recognition Letters 31(14), 2225-2236

Genuer, R. and Poggi, J.M. and Tuleau-Malot, C. (2015), *VSURF: An R Package for Variable Selection Using Random Forests*, The R Journal 7(2):19-33

See Also

VSURF, VSURF_thres, VSURF_interp

Examples

```
## Not run:
data(iris)
iris.thres <- VSURF_thres(iris[,1:4], iris[,5], ntree = 100, nfor.thres = 20)
iris.thres.tuned <- tune(iris.thres, nmin = 10)
iris.thres.tuned
```

```
iris.interp <- VSURF_interp(iris[,1:4], iris[,5], vars = iris.thres$varselect.thres,
                            nfor.interp = 10)
iris.interp.tuned <- tune(iris.interp, nsd = 10)
iris.interp.tuned

## End(Not run)
```

VSURF *Variable Selection Using Random Forests*

Description

Three steps variable selection procedure based on random forests for supervised classification and regression problems. First step ("thresholding step") is dedicated to eliminate irrelevant variables from the dataset. Second step ("interpretation step") aims to select all variables related to the response for interpretation purpose. Third step ("prediction step") refines the selection by eliminating redundancy in the set of variables selected by the second step, for prediction purpose.

Usage

```
VSURF(x, ...)

## Default S3 method:
VSURF(
  x,
  y,
  mtry = max(floor(ncol(x)/3), 1),
  ntree.thres = 500,
  nfor.thres = 20,
  nmin = 1,
  ntree.interp = 100,
  nfor.interp = 10,
  nsd = 1,
  ntree.pred = 100,
  nfor.pred = 10,
  nmj = 1,
  RFimplem = "randomForest",
  parallel = FALSE,
  ncores = detectCores() - 1,
  clusterType = "PSOCK",
  verbose = TRUE,
  ntree = 2000,
  ...
)

## S3 method for class 'formula'
VSURF(formula, data, ..., na.action = na.fail)
```

Arguments

x, formula	A data frame or a matrix of predictors, the columns represent the variables. Or a formula describing the model to be fitted.
...	others parameters to be passed on to the randomForest function (see ?randomForest for further information).
y	A response vector (must be a factor for classification problems and numeric for regression ones).
mtry	Number of variables randomly sampled as candidates at each split. Standard parameter of randomForest.
ntree.thres	Number of trees of each forest grown for "thresholding step" (first of the three steps).
nfor.thres	Number of forests grown for "thresholding step".
nmin	Number of times the "minimum value" is multiplied to set threshold value. See details below.
ntree.interp	Number of trees of each forest grown for "interpretation step" (second of the three steps).
nfor.interp	Number of forests grown for "interpretation step".
nsd	Number of times the standard deviation of the minimum value of err.interp is multiplied. See details below.
ntree.pred	Number of trees of each forest grown for "prediction step" (last of the three steps).
nfor.pred	Number of forests grown for "prediction step".
nmj	Number of times the mean jump is multiplied. See details below.
RFimplem	Choice of the random forests implementation to use : "randomForest" (default), "ranger" or "Rborist" (not that if "Rborist" is chosen, "randoForest" will still be used for the first step VSURF_thres). If a vector of length 3 is given, each coordinate is passed to each intermediate function: VSURF_thres, VSURF_interp, VSURF_pred, in this order.
parallel	A logical indicating if you want VSURF to run in parallel on multiple cores (default to FALSE). If a vector of length 3 is given, each coordinate is passed to each intermediate function: VSURF_thres, VSURF_interp, VSURF_pred, in this order.
ncores	Number of cores to use. Default is set to the number of cores detected by R minus 1.
clusterType	Type of the multiple cores cluster used to run VSURF in parallel. Must be chosen among "PSOCK" (default: SOCKET cluster available locally on all OS), "FORK" (local too, only available for Linux and Mac OS), "MPI" (can be used on a remote cluster, which needs snow and Rmpi packages installed), "ranger" and "Rborist" for internal parallelizations of those packages (not that if "Rborist" is chosen, "SOCKET" will still be used for the first step VSURF_thres). If a vector of length 2 is given, each coordinate is passed to each intermediate function: VSURF_thres, VSURF_interp, in this order.

verbose	A logical indicating if information about method's progress (included progress bars for each step) must be printed (default to TRUE). Adds a small extra overload.
ntree	(deprecated) Number of trees in each forest grown for "thresholding step".
data	a data frame containing the variables in the model.
na.action	A function to specify the action to be taken if NAs are found. (NOTE: If given, this argument must be named, and as randomForest it is only used with the formula-type call.)

Details

- First step ("thresholding step"): first, nfor.thres random forests are computed using the function randomForest with arguments importance=TRUE, and our choice of default values for ntree and mtry (which are higher than default in randomForest to get a more stable variable importance measure). Then variables are sorted according to their mean variable importance (VI), in decreasing order. This order is kept all along the procedure. Next, a threshold is computed: min.thres, the minimum predicted value of a pruned CART tree fitted to the curve of the standard deviations of VI. Finally, the actual "thresholding step" is performed: only variables with a mean VI larger than nmin * min.thres are kept.
- Second step ("interpretation step"): the variables selected by the first step are considered. nfor.interp embedded random forests models are grown, starting with the random forest build with only the most important variable and ending with all variables selected in the first step. Then, err.min the minimum mean out-of-bag (OOB) error of these models and its associated standard deviation sd.min are computed. Finally, the smallest model (and hence its corresponding variables) having a mean OOB error less than err.min + nsd * sd.min is selected.

  Note that for this step (and the next one), the mtry parameter of randomForest is set to its default value (see randomForest) if nvm, the number of variables in the model, is not greater than the number of observations, while it is set to nvm/3 otherwise. This is to ensure quality of OOB error estimations along embedded RF models.
- Third step ("prediction step"): the starting point is the same than in the second step. However, now the variables are added to the model in a stepwise manner. mean.jump, the mean jump value is calculated using variables that have been left out by the second step, and is set as the mean absolute difference between mean OOB errors of one model and its first following model. Hence a variable is included in the model if the mean OOB error decrease is larger than nmj * mean.jump.

  As for interpretation step, the mtry parameter of randomForest is set to its default value if nvm, the number of variables in the model, is not greater than the number of observations, while it is set to nvm/3 otherwise.

VSURF is able to run using multiple cores in parallel (see parallel, clusterType and ncores arguments).

Value

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

`varselect.thres`
: A vector of indexes of variables selected after "thresholding step", sorted according to their mean VI, in decreasing order.

`varselect.interp`
: A vector of indexes of variables selected after "interpretation step".

`varselect.pred`
: A vector of indexes of variables selected after "prediction step".

`nums.varselect`
: A vector of the 3 numbers of variables selected resp. by "thresholding step", "interpretation step" and "prediction step".

`imp.varselect.thres`
: A vector of importance of the `varselect.thres` variables.

`min.thres`
: The minimum predicted value of a pruned CART tree fitted to the curve of the standard deviations of VI.

`imp.mean.dec`
: A vector of the variables importance means (over `nfor.thres` runs), in decreasing order.

`imp.mean.dec.ind`
: The ordering index vector associated to the sorting of variables importance means.

`imp.sd.dec`
: A vector of standard deviations of all variables importance. The order is given by `imp.mean.dec.ind`.

`mean.perf`
: Mean OOB error rate, obtained by a random forests build on all variables.

`pred.pruned.tree`
: Predictions of the CART tree fitted to the curve of the standard deviations of VI.

`err.interp`
: A vector of the mean OOB error rates of the embedded random forests models build during the "interpretation step".

`sd.min`
: The standard deviation of OOB error rates associated to the random forests model attaining the minimum mean OOB error rate during the "interpretation step".

`err.pred`
: A vector of the mean OOB error rates of the random forests models build during the "prediction step".

`mean.jump`
: The mean jump value computed during the "prediction step".

`nmin`, `nsd`, `nmj`
: Corresponding parameters values.

`overall.time`
: Overall computation time.

`comput.times`
: A list of the 3 computation times respectively associated with the 3 steps: "thresholding", "interpretation" and "prediction".

`RFimplem`
: The RF implementation used to run `VSURF`, among "randomForest" (default), "ranger" and "Rborist" or a vector of length 3 with those.

`ncores`
: The number of cores used to run `VSURF` in parallel (NULL if VSURF did not run in parallel).

`clusterType`
: The type of the cluster used to run `VSURF` in parallel (NULL if VSURF did not run in parallel).

`call`
: The original call to `VSURF`.

`terms`
: Terms associated to the formula (only if formula-type call was used).

`na.action`
: Method used to deal with missing values (only if formula-type call was used).

Author(s)

Robin Genuer, Jean-Michel Poggi and Christine Tuleau-Malot

References

Genuer, R. and Poggi, J.M. and Tuleau-Malot, C. (2010), *Variable selection using random forests*, Pattern Recognition Letters 31(14), 2225-2236

Genuer, R. and Poggi, J.M. and Tuleau-Malot, C. (2015), *VSURF: An R Package for Variable Selection Using Random Forests*, The R Journal 7(2):19-33

See Also

plot.VSURF, summary.VSURF, VSURF_thres, VSURF_interp, VSURF_pred, tune

Examples

```
data(iris)
iris.vsurf <- VSURF(iris[,1:4], iris[,5])
iris.vsurf

## Not run:
# A more interesting example with toys data (see \code{\link{toys}})
# (a few minutes to execute)
data(toys)
toys.vsurf <- VSURF(toys$x, toys$y)
toys.vsurf

# VSURF run on 2 cores in parallel (using a SOCKET cluster):
data(toys)
toys.vsurf.parallel <- VSURF(toys$x, toys$y, parallel = TRUE, ncores = 2)

## End(Not run)
```

VSURF_interp *Interpretation step of VSURF*

Description

Interpretation step aims to select all variables related to the response for interpretation purpose. This is the second step of the VSURF function. It is designed to be executed after the thresholding step VSURF_thres.

Usage

```
VSURF_interp(x, ...)

## Default S3 method:
VSURF_interp(
  x,
  y,
  vars,
  ntree.interp = 100,
  nfor.interp = 25,
  nsd = 1,
  RFimplem = "randomForest",
  parallel = FALSE,
  ncores = detectCores() - 1,
  clusterType = "PSOCK",
  verbose = TRUE,
  ntree = NULL,
  ...
)

## S3 method for class 'formula'
VSURF_interp(formula, data, ..., na.action = na.fail)
```

Arguments

x, formula	A data frame or a matrix of predictors, the columns represent the variables. Or a formula describing the model to be fitted.
...	others parameters to be passed on to the randomForest function (see ?randomForest for further information).
y	A response vector (must be a factor for classification problems and numeric for regression ones).
vars	A vector of variable indices. Typically, indices of variables selected by thresholding step (see value varselect.thres of VSURF_thres function).
ntree.interp	Number of trees of each forest grown.
nfor.interp	Number of forests grown.
nsd	Number of times the standard deviation of the minimum value of err.interp is multiplied. See details below.
RFimplem	Choice of the random forests implementation to use : "randomForest" (default), "ranger" or "Rborist" (not that if "Rborist" is chosen, "randoForest" will still be used for the first step VSURF_thres). If a vector of length 3 is given, each coordinate is passed to each intermediate function: VSURF_thres, VSURF_interp, VSURF_pred, in this order.
parallel	A logical indicating if you want VSURF to run in parallel on multiple cores (default to FALSE). If a vector of length 3 is given, each coordinate is passed to each intermediate function: VSURF_thres, VSURF_interp, VSURF_pred, in this order.

ncores	Number of cores to use. Default is set to the number of cores detected by R minus 1.
clusterType	Type of the multiple cores cluster used to run VSURF in parallel. Must be chosen among "PSOCK" (default: SOCKET cluster available locally on all OS), "FORK" (local too, only available for Linux and Mac OS), "MPI" (can be used on a remote cluster, which needs snow and Rmpi packages installed), "ranger" and "Rborist" for internal parallelizations of those packages (not that if "Rborist" is chosen, "SOCKET" will still be used for the first step VSURF_thres). If a vector of length 2 is given, each coordinate is passed to each intermediate function: VSURF_thres, VSURF_interp, in this order.
verbose	A logical indicating if information about method's progress (included progress bars for each step) must be printed (default to TRUE). Adds a small extra overload.
ntree	(deprecated) Number of trees in each forest grown for "thresholding step".
data	a data frame containing the variables in the model.
na.action	A function to specify the action to be taken if NAs are found. (NOTE: If given, this argument must be named, and as randomForest it is only used with the formula-type call.)

Details

nfor.interp embedded random forests models are grown, starting with the random forest build with only the most important variable and ending with all variables. Then, err.min the minimum mean out-of-bag (OOB) error rate of these models and its associated standard deviation sd.min are computed. Finally, the smallest model (and hence its corresponding variables) having a mean OOB error less than err.min + nsd * sd.min is selected.

Note that, the mtry parameter of randomForest is set to its default value (see randomForest) if nvm, the number of variables in the model, is not greater than the number of observations, while it is set to nvm/3 otherwise. This is to ensure quality of OOB error estimations along embedded RF models.

Value

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

varselect.interp	A vector of indices of selected variables.
err.interp	A vector of the mean OOB error rates of the embedded random forests models.
sd.min	The standard deviation of OOB error rates associated to the random forests model attaining the minimum mean OOB error rate.
num.varselect.interp	The number of selected variables.
varselect.thres	A vector of indexes of variables selected after "thresholding step", sorted according to their mean VI, in decreasing order.
nsd	Value of the parameter in the call.

comput.time	Computation time.
RFimplem	The RF implementation used to run VSURF_interp.
ncores	The number of cores used to run VSURF_interp in parallel (NULL if VSURF_interp did not run in parallel).
clusterType	The type of the cluster used to run VSURF_interp in parallel (NULL if VSURF_interp did not run in parallel).
call	The original call to VSURF.
terms	Terms associated to the formula (only if formula-type call was used).

Author(s)

Robin Genuer, Jean-Michel Poggi and Christine Tuleau-Malot

References

Genuer, R. and Poggi, J.M. and Tuleau-Malot, C. (2010), *Variable selection using random forests*, Pattern Recognition Letters 31(14), 2225-2236

Genuer, R. and Poggi, J.M. and Tuleau-Malot, C. (2015), *VSURF: An R Package for Variable Selection Using Random Forests*, The R Journal 7(2):19-33

See Also

VSURF, tune

Examples

```
data(iris)
iris.thres <- VSURF_thres(iris[,1:4], iris[,5])
iris.interp <- VSURF_interp(iris[,1:4], iris[,5],
  vars = iris.thres$varselect.thres)
iris.interp

## Not run:
# A more interesting example with toys data (see \code{\link{toys}})
# (a few minutes to execute)
data(toys)
toys.thres <- VSURF_thres(toys$x, toys$y)
toys.interp <- VSURF_interp(toys$x, toys$y,
  vars = toys.thres$varselect.thres)
toys.interp
## End(Not run)
```

VSURF_pred *Prediction step of VSURF*

Description

Prediction step refines the selection of interpretation step VSURF_interp by eliminating redundancy in the set of variables selected, for prediction purpose. This is the third step of the VSURF function.

Usage

```
VSURF_pred(x, ...)

## Default S3 method:
VSURF_pred(
  x,
  y,
  err.interp,
  varselect.interp,
  ntree.pred = 100,
  nfor.pred = 10,
  nmj = 1,
  RFimplem = "randomForest",
  parallel = FALSE,
  ncores = detectCores() - 1,
  verbose = TRUE,
  ntree = NULL,
  ...
)

## S3 method for class 'formula'
VSURF_pred(formula, data, ..., na.action = na.fail)
```

Arguments

x, formula	A data frame or a matrix of predictors, the columns represent the variables. Or a formula describing the model to be fitted.
...	others parameters to be passed on to the randomForest function (see ?randomForest for further information).
y	A response vector (must be a factor for classification problems and numeric for regression ones).
err.interp	A vector of the mean OOB error rates of the embedded random forests models build during interpretation step (value err.interp of function VSURF_interp).
varselect.interp	A vector of indices of variables selected after interpretation step.
ntree.pred	Number of trees of each forest grown.

nfor.pred	Number of forests grown.
nmj	Number of times the mean jump is multiplied. See details below.
RFimplem	Choice of the random forests implementation to use : "randomForest" (default), "ranger" or "Rborist" (not that if "Rborist" is chosen, "randoForest" will still be used for the first step VSURF_thres). If a vector of length 3 is given, each coordinate is passed to each intermediate function: VSURF_thres, VSURF_interp, VSURF_pred, in this order.
parallel	A logical indicating if you want VSURF to run in parallel on multiple cores (default to FALSE). If a vector of length 3 is given, each coordinate is passed to each intermediate function: VSURF_thres, VSURF_interp, VSURF_pred, in this order.
ncores	Number of cores to use. Default is set to the number of cores detected by R minus 1.
verbose	A logical indicating if information about method's progress (included progress bars for each step) must be printed (default to TRUE). Adds a small extra overload.
ntree	(deprecated) Number of trees in each forest grown for "thresholding step".
data	a data frame containing the variables in the model.
na.action	A function to specify the action to be taken if NAs are found. (NOTE: If given, this argument must be named, and as randomForest it is only used with the formula-type call.)

Details

nfor.pred embedded random forests models are grown, starting with the random forest build with only the most important variable. Variables are added to the model in a stepwise manner. The mean jump value mean.jump is calculated using variables that have been left out by interpretation step, and is set as the mean absolute difference between mean OOB errors of one model and its first following model. Hence a variable is included in the model if the mean OOB error decrease is larger than nmj * mean.jump.

Note that, the mtry parameter of randomForest is set to its default value (see randomForest) if nvm, the number of variables in the model, is not greater than the number of observations, while it is set to nvm/3 otherwise. This is to ensure quality of OOB error estimations along embedded RF models.

Value

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

varselect.pred	A vector of indices of variables selected after "prediction step".
err.pred	A vector of the mean OOB error rates of the random forests models build during the "prediction step".
mean.jump	The mean jump value computed during the "prediction step".
num.varselect.pred	The number of selected variables.

nmj	Value of the parameter in the call.
comput.time	Computation time.
RFimplem	The RF implementation used to run VSURF_pred.
call	The original call to VSURF.
terms	Terms associated to the formula (only if formula-type call was used).

Author(s)

Robin Genuer, Jean-Michel Poggi and Christine Tuleau-Malot

References

Genuer, R. and Poggi, J.M. and Tuleau-Malot, C. (2010), *Variable selection using random forests*, Pattern Recognition Letters 31(14), 2225-2236

Genuer, R. and Poggi, J.M. and Tuleau-Malot, C. (2015), *VSURF: An R Package for Variable Selection Using Random Forests*, The R Journal 7(2):19-33

See Also

VSURF

Examples

```
data(iris)
iris.thres <- VSURF_thres(iris[,1:4], iris[,5])
iris.interp <- VSURF_interp(iris[,1:4], iris[,5],
  vars = iris.thres$varselect.thres)
iris.pred <- VSURF_pred(iris[,1:4], iris[,5],
  err.interp = iris.interp$err.interp,
  varselect.interp = iris.interp$varselect.interp)
iris.pred

## Not run:
# A more interesting example with toys data (see \code{\link{toys}})
# (a few minutes to execute)
data(toys)
toys.thres <- VSURF_thres(toys$x, toys$y)
toys.interp <- VSURF_interp(toys$x, toys$y,
  vars = toys.thres$varselect.thres)
toys.pred <- VSURF_pred(toys$x, toys$y, err.interp = toys.interp$err.interp,
  varselect.interp = toys.interp$varselect.interp)
toys.pred
## End(Not run)
```

VSURF_thres *Thresholding step of VSURF*

Description

Thresholding step is dedicated to roughly eliminate irrelevant variables a the dataset. This is the first step of the VSURF function. For refined variable selection, see VSURF other steps: VSURF_interp and VSURF_pred.

Usage

```
VSURF_thres(x, ...)

## Default S3 method:
VSURF_thres(
  x,
  y,
  mtry = max(floor(ncol(x)/3), 1),
  ntree.thres = 500,
  nfor.thres = 20,
  nmin = 1,
  RFimplem = "randomForest",
  parallel = FALSE,
  clusterType = "PSOCK",
  ncores = parallel::detectCores() - 1,
  verbose = TRUE,
  ntree = NULL,
  ...
)

## S3 method for class 'formula'
VSURF_thres(formula, data, ..., na.action = na.fail)
```

Arguments

x, formula	A data frame or a matrix of predictors, the columns represent the variables. Or a formula describing the model to be fitted.
...	others parameters to be passed on to the randomForest function (see ?randomForest for further information).
y	A response vector (must be a factor for classification problems and numeric for regression ones).
mtry	Number of variables randomly sampled as candidates at each split. Standard parameter of randomForest.
ntree.thres	Number of trees of each forest grown.
nfor.thres	Number of forests grown.

nmin	Number of times the "minimum value" is multiplied to set threshold value. See details below.
RFimplem	Choice of the random forests implementation to use : "randomForest" (default), "ranger" or "Rborist" (not that if "Rborist" is chosen, "randoForest" will still be used for the first step VSURF_thres). If a vector of length 3 is given, each coordinate is passed to each intermediate function: VSURF_thres, VSURF_interp, VSURF_pred, in this order.
parallel	A logical indicating if you want VSURF to run in parallel on multiple cores (default to FALSE). If a vector of length 3 is given, each coordinate is passed to each intermediate function: VSURF_thres, VSURF_interp, VSURF_pred, in this order.
clusterType	Type of the multiple cores cluster used to run VSURF in parallel. Must be chosen among "PSOCK" (default: SOCKET cluster available locally on all OS), "FORK" (local too, only available for Linux and Mac OS), "MPI" (can be used on a remote cluster, which needs snow and Rmpi packages installed), "ranger" and "Rborist" for internal parallelizations of those packages (not that if "Rborist" is chosen, "SOCKET" will still be used for the first step VSURF_thres). If a vector of length 2 is given, each coordinate is passed to each intermediate function: VSURF_thres, VSURF_interp, in this order.
ncores	Number of cores to use. Default is set to the number of cores detected by R minus 1.
verbose	A logical indicating if information about method's progress (included progress bars for each step) must be printed (default to TRUE). Adds a small extra overload.
ntree	(deprecated) Number of trees in each forest grown for "thresholding step".
data	a data frame containing the variables in the model.
na.action	A function to specify the action to be taken if NAs are found. (NOTE: If given, this argument must be named, and as randomForest it is only used with the formula-type call.)

Details

First, nfor.thres random forests are computed using the function randomForest with arguments importance=TRUE, and our choice of default values for ntree and mtry (which are higher than default in randomForest to get a more stable variable importance measure). Then variables are sorted according to their mean variable importance (VI), in decreasing order. This order is kept all along the procedure. Next, a threshold is computed: min.thres, the minimum predicted value of a pruned CART tree fitted to the curve of the standard deviations of VI. Finally, the actual thresholding is performed: only variables with a mean VI larger than nmin * min.thres are kept.

Value

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

varselect.thres	A vector of indices of selected variables, sorted according to their mean VI, in decreasing order.

`imp.varselect.thres`
: A vector of importance of the `varselect.thres` variables.

`min.thres`
: The minimum predicted value of a pruned CART tree fitted to the curve of the standard deviations of VI.

`num.varselect.thres`
: The number of selected variables.

`imp.mean.dec`
: A vector of the variables importance means (over `nfor.thres` runs), in decreasing order.

`imp.mean.dec.ind`
: The ordering index vector associated to the sorting of variables importance means.

`imp.sd.dec`
: A vector of standard deviations of all variables importance. The order is given by `imp.mean.dec.ind`.

`mean.perf`
: The mean OOB error rate, obtained by a random forests build with all variables.

`pred.pruned.tree`
: The predictions of the CART tree fitted to the curve of the standard deviations of VI.

`nmin`
: Value of the parameter in the call.

`comput.time`
: Computation time.

`RFimplem`
: The RF implementation used to run `VSURF_thres`.

`ncores`
: The number of cores used to run `VSURF_thres` in parallel (NULL if VSURF_thres did not run in parallel).

`clusterType`
: The type of the cluster used to run `VSURF_thres` in parallel (NULL if VSURF_thres did not run in parallel).

`call`
: The original call to `VSURF`.

`terms`
: Terms associated to the formula (only if formula-type call was used).

Author(s)

Robin Genuer, Jean-Michel Poggi and Christine Tuleau-Malot

References

Genuer, R. and Poggi, J.M. and Tuleau-Malot, C. (2010), *Variable selection using random forests*, Pattern Recognition Letters 31(14), 2225-2236

Genuer, R. and Poggi, J.M. and Tuleau-Malot, C. (2015), *VSURF: An R Package for Variable Selection Using Random Forests*, The R Journal 7(2):19-33

See Also

`VSURF`, `tune`

Examples

```
data(iris)
iris.thres <- VSURF_thres(iris[,1:4], iris[,5])
iris.thres

## Not run:
# A more interesting example with toys data (see \code{\link{toys}})
# (a few minutes to execute)
data(toys)
toys.thres <- VSURF_thres(toys$x, toys$y)
toys.thres
## End(Not run)
```

Index