VSURF: An R Package for Variable Selection Using Random Forests

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

Abstract This paper describes the R package **VSURF**. Based on random forests, and for both regression and classification problems, it returns two subsets of variables. The first is a subset of important variables including some redundancy which can be relevant for interpretation, and the second one is a smaller subset corresponding to a model trying to avoid redundancy focusing more closely on prediction objective. The two-stage strategy is based on a preliminary ranking of the explanatory variables using the random forests permutation-based score of importance and proceeds using a stepwise forward strategy for variable introduction. The two proposals can be obtained automatically using data-driven default values, good enough to provide interesting results, but can also be tuned by the user. The algorithm is illustrated on a simulated example and its applications to real datasets are presented.

Introduction

Variable selection is a crucial issue in many applied classification and regression problems (see e.g. Hastie et al., 2001). It is of interest for statistical analysis as well as for modelisation or prediction purposes to remove irrelevant variables, to select all important ones or to determine a sufficient subset for prediction. These main different objectives on a statistical learning perspective involve variable selection to simplify statistical problems, to help diagnosis and interpretation, and to speed up data processing.

The authors have proposed (Genuer et al., 2010b) a variable selection method based on random forests (Breiman, 2001), and the aim of this paper is to describe the associated R package called **VSURF** and to illustrate its use on real datasets. The stable version of the package is available on CRAN.[1] In order to make the paper self-contained, the description of the variable selection method is provided. A simulated toy example and a classical real dataset are processed using **VSURF**. In addition, two real examples in a high dimensional setting, not previously addressed by the authors, are used to illustrate the value of the strategy and the effectiveness of the R package.

Introduced by Breiman (2001), random forests (abbreviated RF in the sequel) are an attractive nonparametric statistical method to deal with such problems, since it requires only mild conditions on the model supposed to have generated the observed data. Indeed, since it is based on decision trees and it uses aggregation ideas, RF allow us to consider in an elegant and versatile framework different models and problems, namely regression, two-class and multiclass classifications.

Considering a learning set $L = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$ supposed to be independent observations of the random vector (X, Y), we distinguish as usual the predictors (or explanatory variables), collected in the vector $X = (X^1, ..., X^p)$ where $X \in \mathbb{R}^p$, from the explained variable $Y \in \mathcal{Y}$ where Y is either a class label for classification problems or a numerical response for regression ones. Let us recall that a classifier t is a mapping $t : \mathbb{R}^p \to \mathcal{Y}$ while the regression function appears naturally to be the function s when we suppose that $Y = s(X) + \varepsilon$ with $E[\varepsilon|X] = 0$. Then random forests provide estimators of either the Bayes classifier, which minimizes $P(Y \neq t(X))$ the classification error, or the regression function.

The CART (Classification and Regression Trees) method defined by Breiman et al. (1984) is a well-known way to design optimal single trees. It proceeds by performing first a growing step and then a pruning one. The principle of random forests is to aggregate many binary decision trees coming from two random perturbation mechanisms: the use of bootstrap samples of L instead of L and the random choice at each node of a subset of explanatory variables instead of all of them. There are two main differences with respect to CART trees: first, in the growing step, at each node, a fixed number of input variables are randomly chosen and the best split is calculated only among them and, second, no pruning step is performed so all the trees of the forest are maximal trees. RF algorithm is a very popular machine learning algorithm and appears to be powerful in a lot of different applications, see for example Verikas et al. (2011) and Boulesteix et al. (2012) for recent surveys.

Several implementations of these methods are available. Focusing on R packages we must mention **rpart** (Therneau et al., 2015) for CART, **randomForest** (Liaw and Wiener, 2002) for RF, **party** (Hothorn et al., 2006) for CART and RF (through the function `cforest`) and **ipred** (Peters et al., 2002) for bagging

[1] The current version under development of the package is also available at `https://github.com/robingenuer/VSURF`.

(Breiman, 1996), a closely related method cited here for the sake of completeness. In this paper, we use the **randomForest** procedure, which is based on the initial contribution of Breiman and Cutler (2004). We will concentrate on the prediction performance of RF focusing on out-of-bag (OOB) error (see Breiman, 2001) and on the quantification of the variable importance (VI in the sequel) which are key ingredients for our variable selection strategy. For a general discussion about variable importance, see Azen and Budescu (2003). In the random forests framework, one of the most widely used score of importance of a given variable is the increasing in mean of the error of a tree (mean square error (MSE) for regression and misclassification rate for classification) in the forest when the observed values of this variable are randomly permuted in the OOB samples (see Archer and Kimes, 2008).

Strobl et al. (2007) showed that VI scores are biased towards correlated variables, and Strobl et al. (2008) proposed an alternative permutation scheme as a solution, but it increases the computation cost. This seems to be especially critical for high-dimensional problems with strongly correlated predictors. Nevertheless, our previous experiments (Genuer et al., 2010b) on variable selection and, more recently, the theoretical study of Gregorutti et al. (2013) show that, in some situations, the VI scores are biased towards uncorrelated variables. Additional theoretical results and experiments are needed to more deeply understand these phenomenons, but they are out of the scope of the paper.

A lot of variable selection procedures are based on the cooperation of variable importance for ranking and model estimation to generate, evaluate and compare a family of models, in particular in the family of "wrapper" methods (Kohavi and John, 1997; Guyon and Elisseeff, 2003) which include the prediction performance in the score calculation, for which a lot of methods can be cited. We choose to highlight one of them since it is widely used and close to our procedure. Díaz-Uriarte and Alvarez De Andres (2006) propose a strategy based on recursive elimination of variables. At the beginning, they compute RF variable importance and then, at each step, eliminate iteratively the 20% of the variables having the smallest importance and build a new forest with the remaining variables. The final set of variables is selected by minimizing over the obtained forests, the OOB error rate defined by:

$$errOOB = \begin{cases} \frac{1}{n}\text{Card}\left\{i \in \{1,\ldots,n\} \mid y_i \neq \hat{y}_i\right\} & \text{in the classification framework} \\ \frac{1}{n}\sum_{i\in\{1,\ldots,n\}} \left(y_i - \hat{y}_i\right)^2 & \text{in the regression framework} \end{cases}$$

where $\hat{y}_i$ is the aggregation of the predicted values by trees t for which (x_i, y_i) belongs to the associated *OOB* sample (data not included in the bootstrap sample used to construct t). The proportion of variables to eliminate is an arbitrary parameter of their method and does not depend on the data. Let us remark that we propose an heuristic strategy which does not depend on specific model hypotheses, but based on data-driven thresholds to take decisions.

This topic of variable selection still continues to be interesting. Indeed recently Hapfelmeier and Ulm (2012) propose a new variable selection approach using random forests and, more generally, Cadenas et al. (2013) describe and compare such different approaches in a survey paper.

Some packages are available to cope with variable selection problems. Let us cite, for classification problems the R package **Boruta**, described in Kursa and Rudnicki (2010), finding all relevant variables using a random forest classification algorithm which removes iteratively the variables using a statistical test. An R package **varSelRF**, described in Diaz-Uriarte (2007), implements the previously described method for selecting very small sets of genes in the context of classification. The R package **ofw** (Lê Cao and Chabrier, 2008), also dedicated to the context of classification, selects relevant variables with the application of supervised multiclass classifiers such as CART or support vector machines. The R package **spikeSlabGAM** implements Bayesian variable selection via regularized estimation in additive mixed models (Scheipl, 2011). Dedicated to the biomarker identification in the life sciences, the R package **BioMark**, implements two meta-statistics for variable selection (Wehrens et al., 2012): the first sets a data-dependent selection threshold for significance, useful when two groups are compared and the second, more general, uses repeated subsampling and selects the model coefficients remaining consistently important.

The paper is organized as follows. After this introduction, we present the general variable selection strategy. We then describe how to use the package **VSURF** on a simple simulated dataset. Finally, we examine three real-life examples to illustrate the method in action on high dimensional situations as well as on a standard one.

The strategy

Objectives

In Genuer et al. (2010b) we have distinguished two variable selection objectives called interpretation and prediction.

Even if this distinction can be a little bit confusing since we use for both objectives the same criterion related to prediction performance, the idea is the following. The first objective, called interpretation, is to find important variables highly related to the response variable, even with some redundancy, possibly high. The second one, namely prediction, is to find a smaller number of variables with very low redundancy and sufficient for a good enough prediction of the response variable. The terminology used here can be misunderstood since usually for interpretation, one usually looks for parsimony but in many situations one may often want to identify all predictor variables associated with the response to interpret correctly the relation. We thank one anonymous reviewer for raising this issue and we detail an example in the next paragraph to clarify the difference.

A typical situation illustrates the distinction between the two kinds of variable selection. Let us consider a high dimensional ($n \ll p$) classification problem for which the predictor variables are associated to a pixel in an image or a voxel in a 3D-image like in fMRI brain activity classification problems (see e.g. Genuer et al., 2010a). In such situations, it is supposed that a lot of variables are useless and that there exists unknown groups of highly correlated predictors corresponding to brain regions involved in the response to a given stimulation. The two distinct objectives about variable selection can be of interest. Finding all the important variables highly related to the response variable is useful for interpretation, since it corresponds to the determination of entire regions in the brain or a full parcel in an image. And finding a small number of variables sufficient for good prediction, allows to get the most discriminant variables within the previously highlighted regions. For a more formal approach to this distinction, see also the interesting paper of Nilsson et al. (2007).

Principle

The principle of the two-steps algorithm is the following. First, we rank the variables according to a variable importance measure and the unimportant ones are eliminated. Second, we provide two different subsets obtained either by considering a collection of nested RF models and selecting the variables of the more accurate one, or by introducing sequentially the sorted variables.

Since the quantification of the variable importance is crucial for our procedure, let us recall the definition of RF variable importance. For each tree t of the forest, consider the associated OOB_t sample (data not included in the bootstrap sample used to construct t). Denote by $errOOB_t$ the error (MSE for regression and misclassification rate for classification) of a single tree t on this OOB_t sample. Now, randomly permute the values of X^j in OOB_t to get a perturbed sample denoted by $\widetilde{OOB_t}^j$ and compute $err\widetilde{OOB_t}^j$, the error of predictor t on the perturbed sample. Variable importance of X^j is then equal to:

$$VI(X^j) = \frac{1}{ntree} \sum_t \left(err\widetilde{OOB_t}^j - errOOB_t \right)$$

where the sum is over all trees t of the RF and $ntree$ denotes the number of trees of the RF. Notice that we use this definition of importance and not the normalized one. Indeed, instead of considering (as mentioned in Breiman and Cutler, 2004) that the raw VI are independent replicates, normalizing them and assuming normality of these scores, we prefer a fully data driven solution. This is a key point of our strategy: we prefer to estimate directly the variability of importance across repetitions of forests instead of using normality when $ntree$ is sufficiently large, which is only valid under some specific conditions. Those conditions are difficult to check since their validity depends heavily on tuning parameters and problem peculiarities, so data-driven normalization prevents some misspecified asymptotic behavior.

Another useful element to give the rationale of this kind of variable selection procedure based on a classical stepwise method combined with the use of VI measure, is that a variable not included in the underlying true model has a null "theoretical" importance. In a recent paper Gregorutti et al. (2013) theoretically state that, in the case of additive models, irrelevant variables have null "theoretical" VI. A similar result for the mean decrease impurity index (not used in this paper) has been proven by Louppe et al. (2013).

Detailed strategy

Let us now describe more precisely our two-steps procedure. Note that each RF is typically built using $ntree = 2000$ trees.

- Step 1. Preliminary elimination and ranking:
  - Rank the variables by sorting the VI (averaged from typically 50 RF runs) in descending order.

  - Eliminate the variables of small importance (let m denotes the number of remaining variables).

    More precisely, starting from this order, consider the ordered sequence of the corresponding standard deviations (sd) of VI and use it to estimate a threshold value for VI. Since variability of VI is larger for true variables with respect to useless ones, the threshold value is then given by an estimation of the VI standard deviation for useless variables. This threshold is set to the minimum prediction value given by a CART model where the Y are the sd of the VI and the X are the ranks.

    Then only the variables with an averaged VI exceeding this threshold are retained.

- Step 2. Variable selection:

  - For *interpretation*: construct the nested collection of RF models involving the k first variables, for $k = 1$ to m and select the variables involved in the model leading to the smallest OOB error. This leads to consider m' variables.

    More precisely, we compute OOB error rates of RF (averaged typically on 25 runs) of the nested models starting from the one with only the most important variable, and ending with the one involving all important variables kept previously. Ideally, the variables of the model leading to the smallest OOB error are selected. In fact, in order to deal with instability, we use a classical trick: we select the smallest model with an OOB error less than the minimal OOB error augmented by its standard deviation (based on the same 25 runs).

  - For *prediction*: starting from the ordered variables retained for interpretation, construct an ascending sequence of RF models, by invoking and testing the variables stepwise. The variables of the last model are selected.

    More precisely, the sequential variable introduction is based on the following test: a variable is added only if the error decrease is larger than a threshold. The idea is that the OOB error decrease must be significantly greater than the average variation obtained by adding noisy variables. The threshold is set to the mean of the absolute values of the first order differentiated OOB errors between the model with m' variables and the one with m variables:

    $$\frac{1}{m-m'} \sum_{j=m'}^{m-1} \mid errOOB(j+1) - errOOB(j) \mid \tag{1}$$

    where $errOOB(j)$ is the OOB error of the RF built using the j most important variables.

Comments

In addition to the detailed strategy, let us give some additional comments.

About the first step of our procedure, our previous simulation study in Genuer et al. (2010b) shows that both VI level and variability are larger for relevant variables.

We assume that the stabilized ranking performed in the first step of our procedure will not change after the elimination step.

The use of CART to find the threshold is of course not strictly necessary and can be replaced by many other strategies but it is interesting in our case because the idea is typically to find a constant on a large interval. Since CART fits a piece-wise constant function, this desired constant is the threshold for VI and is defined as the minimum prediction value given by a CART model fitting the curve of VI standard deviations. It is obtained automatically by the CART strategy whereas the user needs to select some window parameter if, for example, a simple smoothing method is used.

Note that the threshold value is based on VI standard deviations while the effective thresholding is performed on VI mean. In general, this rule is conservative and leads to keep more variables than necessary, postponing to the next step a more parsimonious choice.

In addition, we note that several implementation choices have been made: **randomForest** for RF fitting and VI calculation (repeated 50 times to quantify variability) and **rpart** for estimating the VI variability level associated with irrelevant variables leading to a convenient threshold value. Of course other choices would be possible, for example, using the function `cforest` of the package **party** to implement the same scheme of variable selection.

The next section describes the use of the package **VSURF** together with a complete illustration of the strategy which is not supported by specific model hypotheses but based on data-driven thresholds to take decisions.

But before entering in this description, let us emphasize that the objects of interest here are the subsets of important variables and not precisely the corresponding models. Thus OOB errors from the figures (or from the fitted objects) cannot be used or reported as the prediction error estimation of the final model which needs a proper cross-validation scheme.

Using the package VSURF

From data to final results

We illustrate the use of the **VSURF** package on a simple simulated moderately high dimensional dataset ($n = 100 < p = 200$) called toys data, introduced by Weston et al. (2003). It comes from an equiprobable two-class problem, $Y \in \{-1, 1\}$, with 6 influential variables, the others being some noise. Let us define the simulation model by giving the conditional distribution of the X^i for $Y = y$:

- For the six first variables: with probability 0.7, $X^i \sim \mathcal{N}(yi, 1)$ for $i = 1, 2, 3$ and $X^i \sim \mathcal{N}(0, 1)$ for $i = 4, 5, 6$; with probability 0.3, $X^i \sim \mathcal{N}(0, 1)$ for $i = 1, 2, 3$ and $X^i \sim \mathcal{N}\left(y(i-3), 1\right)$ for $i = 4, 5, 6$.
- Remaining variables are noise: $X^i \sim \mathcal{N}(0, 1)$ for $i = 7, \ldots, p$.

Actual variables obtained after the simulation are standardized.

First, we load **VSURF** package and toys data (and fix the random number generation seed for reproducibility):

```
> library("VSURF")
> data("toys")
> set.seed(3101318)
```

The standard way to use **VSURF** package is to use the VSURF function. This function executes the procedure all at once, and is just a wrapper for the three intermediate functions VSURF_thres, VSURF_interp, VSURF_pred described in the next section. Typical use of the VSURF function is as follows:

```
> toys.vsurf <- VSURF(x = toys$x, y = toys$y, mtry = 100)
```

The only mandatory inputs are x, an object containing input variables, and y, the output variable.

In this example, we also choose a specific value for mtry (default is $p/3$ for classification and regression problems in VSURF), which only affects RF runs in the first step of the procedure. In addition, we stress that the default value for ntree is 2000 in VSURF. Those values were considered as well adapted for VI calculations (see Genuer et al., 2010b, Section 2.2) and these two arguments are passed on the randomForest function (we kept the same name for consistency).

The function outputs a list containing all results.

```
> names(toys.vsurf)

 [1] "varselect.thres"     "varselect.interp"    "varselect.pred"
 [4] "nums.varselect"      "imp.varselect.thres" "min.thres"
 [7] "imp.mean.dec"        "imp.mean.dec.ind"    "imp.sd.dec"
[10] "mean.perf"           "pred.pruned.tree"    "err.interp"
[13] "sd.min"              "err.pred"            "mean.jump"
[16] "nmin"                "nsd"                 "nmj"
[19] "overall.time"        "comput.times"        "ncores"
[22] "clusterType"         "call"
```

The most important objects are varselect.thres, varselect.interp, varselect.pred, which contain the set of variables selected after the thresholding, interpretation and prediction step respectively.

The summary method gives a short summary of the results: numbers of selected variables and computation times.

```
> summary(toys.vsurf)

 VSURF computation time: 1.6 mins

 VSURF selected:
        36 variables at thresholding step (in 57.6 secs)
        4 variables at interpretation step (in 38.4 secs)
        3 variables at prediction step (in 2.2 secs)
```

The plot method gives a plot (see Figure 1) of the results using 4 graphs. To get selected graphs of Figure 1 one can either use parameters of this plot method, or use intermediate plot methods (associated to each procedure step) included in the package.

```
> plot(toys.vsurf)
```

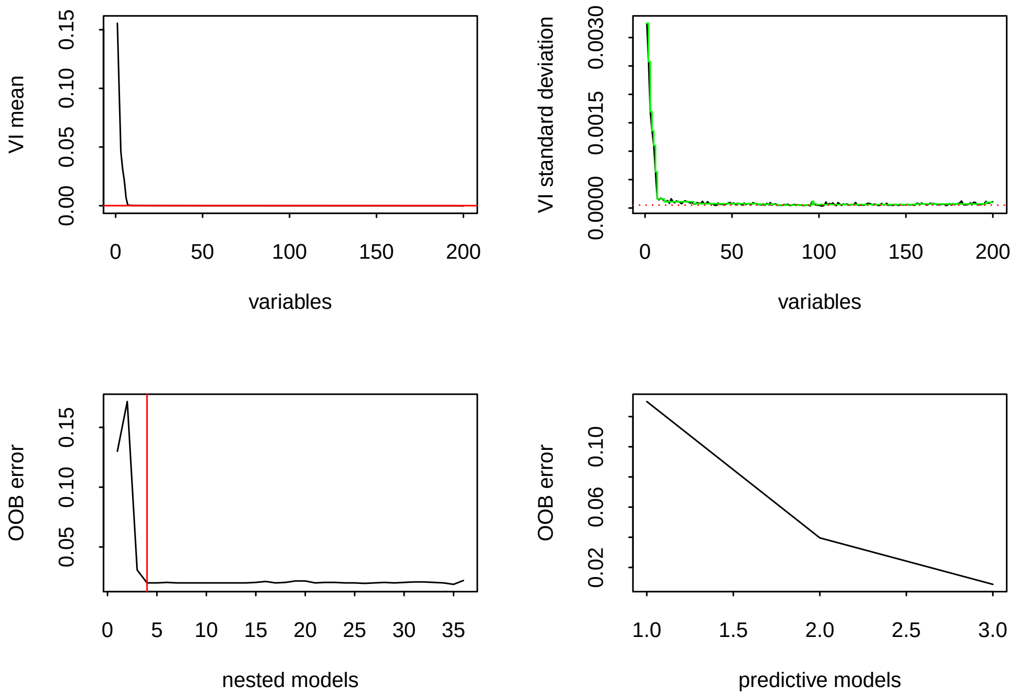

Figure 1: VSURF for toys data: Top graphs illustrate the thresholding step, bottom left and bottom right graphs are associated with interpretation and prediction steps respectively.

In addition, we include a predict method, which permits to predict outcomes of new data with RF only using variables selected by VSURF.

Finally, we point out that all computations of the package can be executed in parallel, whenever it is possible. For example, the following command runs VSURF in parallel on a Linux computing server using 40 cores:

```
> set.seed(2734, kind = "L'Ecuyer-CMRG")
> toys.vsurf.parallel <- VSURF(toys$x, toys$y, mtry = 100, parallel = TRUE,
+                              ncores = 40, clusterType = "FORK")

> summary(toys.vsurf.parallel)

 VSURF computation time: 8.3 secs

 VSURF selected:
        35 variables at thresholding step (in 3.8 secs)
        4 variables at interpretation step (in 2.3 secs)
        3 variables at prediction step (in 2.2 secs)

 VSURF ran in parallel on a FORK cluster and used 40 cores
```

Note that we use L'Ecuyer-CMRG kind in set.seed function to allow reproducibility (when the call is on a FORK cluster with the same number of cores). Even if one should have a computing server with 40 cores to benefit from this execution time reduction, the resulting VSURF object from this call will be the same on a computer with fewer cores available. Parallel call of VSURF will be much used to deal with the two high dimensional datasets at the end of the paper.

How to get intermediate results

Let us now detail the main stages of the procedure together with the results obtained on toys data. Note that unless explicitly stated otherwise, all graphs refer to Figure 1.

- Step 1.
  - Variable ranking.

    The result of variable ranking is drawn on the top left graph. True variables are significantly more important than the noisy ones.
  - Variable elimination.

    Starting from this order, the plot of the corresponding standard deviations of VI is used to estimate a threshold value for VI. This threshold (figured by the dotted horizontal red line in Figure 2, which is a zoom of the top right graph of Figure 1) is set to the minimum prediction value given by a CART model fitting this curve (see the green piece-wise constant function on the same graph).

    Then only the variables with an averaged VI exceeding this level (i.e. above the horizontal red line in the top left graph of Figures 1) are retained.

    The computation of the 50 forests, the ranking and elimination steps are obtained with the VSURF_thres function:

```
> set.seed(3101318)
> toys.thres <- VSURF_thres(toys$x, toys$y, mtry = 100)
```

The VSURF_thres function outputs a list containing all results of this step. Main outputs are: varselect.thres which contains the indices of variables selected by this step, imp.mean.dec and imp.sd.dec which hold the VI mean and standard deviation (the order according to decreasing VI mean can be found in imp.mean.dec.ind).

```
> toys.thres$varselect.thres
```

```
 [1]   3   2   6   5   1   4 184  37 138 159  81  17 180 131  52 191
[17]  96 192 165  94 198  25  21 109  64  12  29 188 107 157  70  46
[33]  54 143 186 111
```

Finally, Figure 2 can be obtained with the following command:

```
> plot(toys.vsurf, step = "thres", imp.mean = FALSE, ylim = c(0, 2e-4))
```

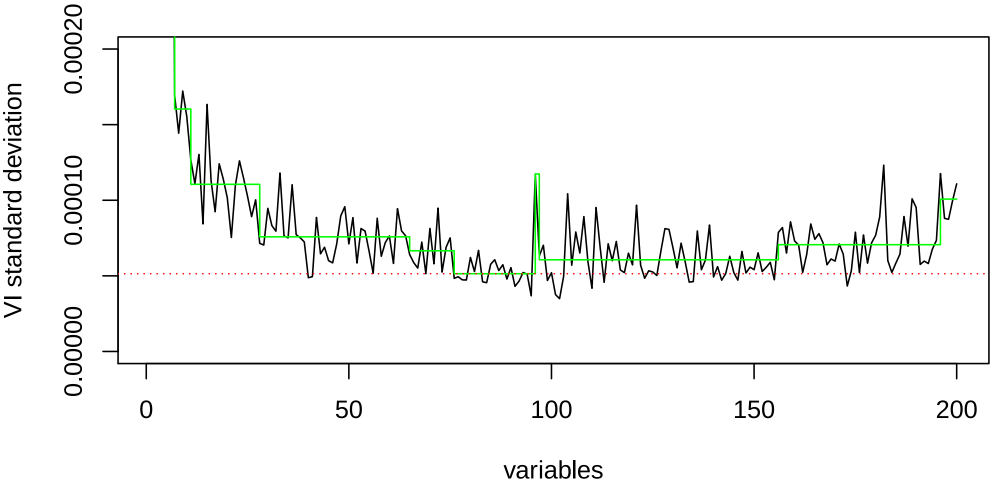

Figure 2: Zoom of the top right graph of Figure 1.

We can see on the plot of standard deviations of VI (top right graph of Figure 1) that the true variables' standard deviations are large compared to those of the noisy variables, which are close to zero.

- Step 2.
  - Variable selection procedure for interpretation.

    We use `VSURF_interp` for this step. Note that we have to specify the indices of variables selected at the previous step. So we set argument `vars` to `toys.thres$varselect.thres`:

    ```
    > toys.interp <- VSURF_interp(toys$x, toys$y,
    +                              vars = toys.thres$varselect.thres)
    ```

    The list resulting from `VSURF_interp` function mainly contains `varselect.interp`: the variables selected by this step, and `err.interp`: OOB error rates of RF nested models.

    ```
    > toys.interp$varselect.interp
    ```

    ```
    [1] 3 2 6 5
    ```

    In the bottom left graph, we see that the error decreases quickly. It reaches its (almost) minimum when the first four true variables are included in the model (see the vertical red line) and then it remains nearly constant. The selected model contains variables V3, V2, V6, V5, which are four of the six true variables, while the actual minimum is reached for 35 variables.

    Note that, to ensure quality of OOB error estimations (e.g. Genuer et al., 2008, Section 2) along embedded RF models, the `mtry` parameter of the `randomForest` function is here set to its default value if k (the number of variables involved in the current RF model) is not greater than n, while it is set to $k/3$ otherwise.
  - Variable selection procedure for prediction.

    We use the `VSURF_pred` function for this step. We need to specify the error rates and the variables selected in the interpretation step in the `err.interp` and `varselect.interp` arguments:

    ```
    > toys.pred <- VSURF_pred(toys$x, toys$y,
    +                         err.interp = toys.interp$err.interp,
    +                         varselect.interp = toys.interp$varselect.interp)
    ```

    Main outputs of the `VSURF_pred` function are the variables selected by this final step, `varselect.pred`, and the OOB error rates of RF models, `err.pred`.

    ```
    > toys.pred$varselect.pred
    ```

    ```
    [1] 3 6 5
    ```

    For the `toys` data, the final model for prediction purpose involves only variables V3, V6, V5 (see the bottom right graph). The threshold is set to the mean of the absolute values of the first order differentiated OOB errors between the model with $m' = 4$ variables and the one with $m = 36$ variables.

Finally, we mention that `VSURF_thres` and `VSURF_interp` can be executed in parallel with the same syntax as `VSURF` (setting `parallel = TRUE`), while `VSURF_pred` can not be parallelized.

Tuning the different steps of the procedure

We provide two additional functions for tuning the thresholding and interpretation steps without having to rerun all computations.

- First, a `tune` method which, applied to the result of `VSURF_thres`, can be used to tune the thresholding step. We can use the `nmin` parameter (which has default value 1) in order to set the threshold to the minimum prediction value given by the CART model times `nmin`.

  ```
  > toys.thres.tuned <- tune(toys.thres, nmin = 3)
  > toys.thres.tuned$varselect.thres
  ```

  ```
   [1]   3   2   6   5   1   4 184  37 138 159  81  17 180 131  52 191
  ```

  We get 16 selected variables instead of 36 previously.
- Secondly, a `tune` method which, applied to the result of `VSURF_interp`, is of the same kind and allows to tune the interpretation step. If we now want to be more restrictive in our selection in interpretation step, we can select the smallest model with an OOB error less than the minimal OOB error augmented by its empirical standard deviation times `nsd` (with `nsd` ≥ 1).

  ```
  > toys.interp.tuned <- tune(toys.interp, nsd = 5)
  > toys.interp.tuned$varselect.interp
  ```

```
[1] 3 2 6
```

We get 3 selected variables instead of 4 previously.

We did not write a tuning method for prediction step because it is a recursive step and needs to recompute the sequence. Hence, to adjust the parameter for this step, we have to rerun VSURF_pred function with a different value of nmj (which has default value 1). This multiplicative constant allows to modulate the threshold defined in (1).

For example, increasing the value of nmj leads to selection of fewer variables:

```
> toys.pred.tuned <- VSURF_pred(toys$x, toys$y, err.interp = toys.interp$err.interp,
+                         varselect.interp = toys.interp$varselect.interp, nmj = 70)
> toys.pred.tuned$varselect.pred

[1] 3 6
```

Remark

Thanks to one of the three anonymous reviewers, we formulate a warning following a remark made by Svetnik et al. (2004). Indeed, this work exhibits a situation where the link between X and Y was broken and for which they use OOB errors in a recursive strategy, different but not too far from our procedure, to select variables. Their results show that this kind of strategy can be seriously biased and can overfit the data. To illustrate directly this phenomenon, let us start with the original dataset toys for which our procedure performs quite well (see the paper Genuer et al., 2010b, containing an extensive study of the operating characteristics of the algorithm including no overfitting in presence of many additional dummy variables). Then, modify it by scrambling the Y values thus removing the link between X and Y. Applying VSURF on this modified dataset leads to an OOB error rate, in interpretation step, starting from 50% (which is correct) and exhibiting a minimum for 10 variables corresponding to 37%. So, in this situation for which there is no optimal solution and the desirable behavior is to find a constant OOB error rate along the sequence, the procedure still delivers a solution. So, the conclusion is that even when there is no link between X and Y the procedure can highlight a set of variables.

Of course, using an external 5-fold cross-validation of our entire procedure leads to the correct estimate of 51% error rate for both interpretation and prediction steps and the correct conclusion: there is nothing to find in the data. Alternatively, if we simulate a second sample coming from the same simulation model and use it only to rank the variables, the interpretation step exhibits an OOB error rate curve oscillating around 50%, leading to the good conclusion.

Three illustrative examples

In this section we apply the proposed procedure on three real-life examples: two high dimensional datasets (associated with a regression problem and a classification one respectively) and, before that, a standard one to illustrate the versatility of the procedure.

Let us mention that VSURF stability is a natural issue (see e.g. Meinshausen and Bühlmann, 2010) and is considered in Genuer et al. (2010b) Sections 3 and 4.

Ozone data

The Ozone dataset consists of $n = 366$ observations of 12 independent variables and 1 dependent variable. These variables are numbered as in the R package **mlbench** (Leisch and Dimitriadou, 2010): 1-Month, 2-Day of month, 3-Day of week, 5-Pressure height, 6-Wind speed, 7-Humidity, 8-Temperature (Sandburg), 9-Temperature (El Monte), 10-Inversion base height, 11-Pressure gradient, 12-Inversion base temperature, 13-Visibility, for independent variables and 4-Daily maximum one-hour-average ozone, for the dependent variable.

The interest of this dataset is that it has been extensively studied and that even if it is a real one, it is possible to a priori know the expected variables. Moreover, this dataset, which is not a high dimensional one, includes some missing data and allows us to give an example of how to handle such data using **VSURF**.

To begin, we load the data:

```
> data("Ozone", package = "mlbench")
> set.seed(221921186)
```

Then, we apply the complete procedure via VSURF. Note that the following formula-type call is necessary to handle missing values (as in randomForest).

```
> vozone <- VSURF(V4 ~ ., data = Ozone, na.action = na.omit)
> summary(vozone)

 VSURF computation time: 1.7 mins

 VSURF selected:
        9 variables at thresholding step (in 1.1 mins)
        5 variables at interpretation step (in 26.3 secs)
        5 variables at prediction step (in 12.4 secs)
```

As a first step, we look at the variable importance associated with each of the explanatory variables.

```
> plot(vozone, step = "thres", imp.sd = FALSE, var.names = TRUE)
```

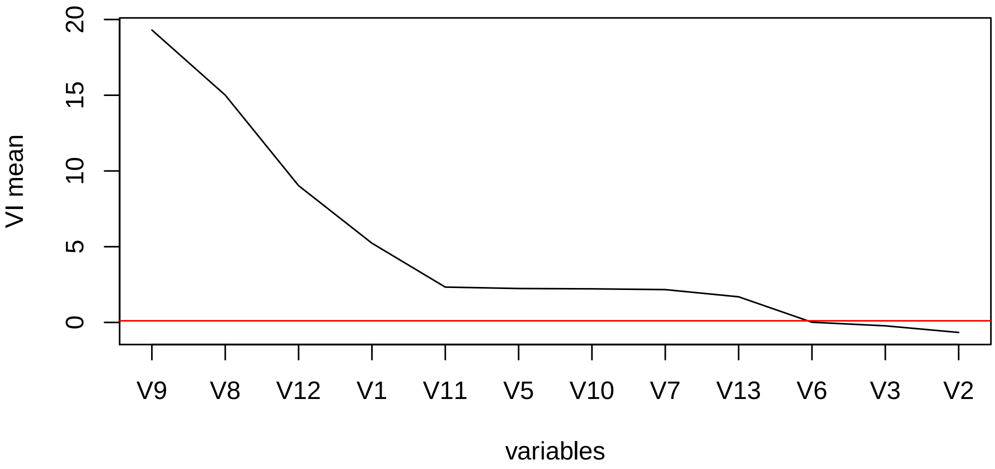

Figure 3: Sorted VI mean associated with the 12 explanatory variables of Ozone data, with variable names in x-axis.

On Figure 3, and as noticed in previous studies, three very sensible groups of variables appear from the most to the least important. First, the two temperatures (8 and 9), the inversion base temperature (12) known to be the best ozone predictors, and the month (1), which is an important predictor since ozone concentration exhibits an heavy seasonal component. A second group of clearly less important meteorological variables: pressure height (5), humidity (7), inversion base height (10), pressure gradient (11) and visibility (13). Finally three unimportant variables: day of month (2), day of week (3) of course and more surprisingly wind speed (6). This last fact is classical: wind enters in the model only when ozone pollution arises, otherwise wind and pollution are weakly correlated (see for example Chèze et al., 2003, who highlight this phenomenon using partial estimators).

Let us now examine the results of the selection procedures. To reflect the order used in the variables definition, we reorder the output variables of the procedure.

```
> number <- c(1:3,5:13)
> number[vozone$varselect.thres]

[1]  9  8 12  1 11  5 10  7 13
```

After the first elimination step, the 3 variables of negative importance (variables 6, 3 and 2) are eliminated, as expected.

```
> number[vozone$varselect.interp]

[1]  9  8 12  1 11
```

Then the interpretation procedure leads to select the model with 5 variables, which contains all the most important variables.

```
> number[vozone$varselect.pred]

[1]  9  8 12  1 11
```

With the default settings, the prediction step does not remove any additional variable.

Remark

Even though the comparison with other variable selection strategies is out of the scope of the paper, one of the three anonymous reviewers have kindly compared the results of the **VSURF** with the results of the R package **Boruta**, described in Kursa and Rudnicki (2010), on the two datasets toys and Ozone.

Let us recall that this module directly aims at selecting all-relevant features. Hence the comparison with the interpretation set delivered by **VSURF** is of interest. In the toys dataset, the number of truly relevant variables is 6. **VSURF** finds 4 out of 6 variables in the interpretation stage while **Boruta** finds all six truly relevant variables and one more false positive. In the case of Ozone data set **VSURF** finds 5 variables in the interpretation stage, while **Boruta** finds 9. So, this seems to confirm that the heuristic proposed by **VSURF** is prediction oriented (the price to pay is a risk of false negative) and suggests that the strategy proposed by **Boruta** is more accurate to recover weak redundant correlations between predictors and decision variable (the price to pay seems to be a risk of false positive).

In fact our strategy assumes more or less there are unnecessary variables in the set of all available variables initially, which is not really the case in the Ozone dataset. However, it is the case in the following two high-dimensional examples.

Toxicity data

This second dataset is also a regression framework, however, unlike before, this case is a high dimensional problem.

The liver.toxicity dataset, available in the R package **mixOmics** (Lê Cao et al., 2015), is a real dataset from a study by Heinloth et al. (2004). In this study, 4 male rats of the inbred strain Fisher 344 were exposed to different doses of acetaminophen (non toxic dose (50 or 150 mg/kg), moderate toxic dose (1500 mg/kg), severe toxic dose (2000 mg/kg)) in a controlled experiment. Necropsies were performed at different hours after exposure (6, 18, 24 and 48 hours) and the mRNA from the liver was extracted. In this original study, 10 clinical chemistry variables containing markers for the liver injury were measured. Those variables are numerical variables since they measure the serum enzymes level. For our study, the dataset extracted from this study contains:

- a data frame, called gene, with 64 rows representing the subjects and 3116 columns representing explanatory variables which are the gene expression levels after normalization and preprocessing due to Bushel et al. (2007),
- a vector, called clinic, with 64 rows and 1 column, one of the 10 clinical variables for the same 64 subjects: more precisely, the variable coded by ALB.g.dL. which corresponds to the albumin level, which is the one considered in González et al. (2012).

As in previous studies (Gidskehaug et al., 2007; Lê Cao et al., 2008), our aim is, using **VSURF**, to predict our clinical variable by the genes.

First, we load the data as follows:

```
> data("liver.toxicity", package = "mixOmics")
> clinic <- liver.toxicity$clinic$ALB.g.dL.
> set.seed(7162013, "L'Ecuyer-CMRG")
```

Now we apply our procedure and analyze the results.

```
> vtoxicity <- VSURF(liver.toxicity$gene, clinic, parallel = TRUE, ncores = 40,
+                      clusterType = "FORK")

> summary(vtoxicity)

 VSURF computation time: 5.9 mins

 VSURF selected:
       550 variables at thresholding step (in 1.1 mins)
       5 variables at interpretation step (in 4.8 mins)
```

```
         5 variables at prediction step (in 3.4 secs)

 VSURF ran in parallel on a FORK cluster and used 40 cores
```

We notice that after the elimination step, only 550 variables remain, thus the number of variables has been divided quite by six. This ratio is not surprising since we know that there exists extreme redundancy in the gene expression data together with a lot of irrelevant variables.

```
> plot(vtoxicity)
```

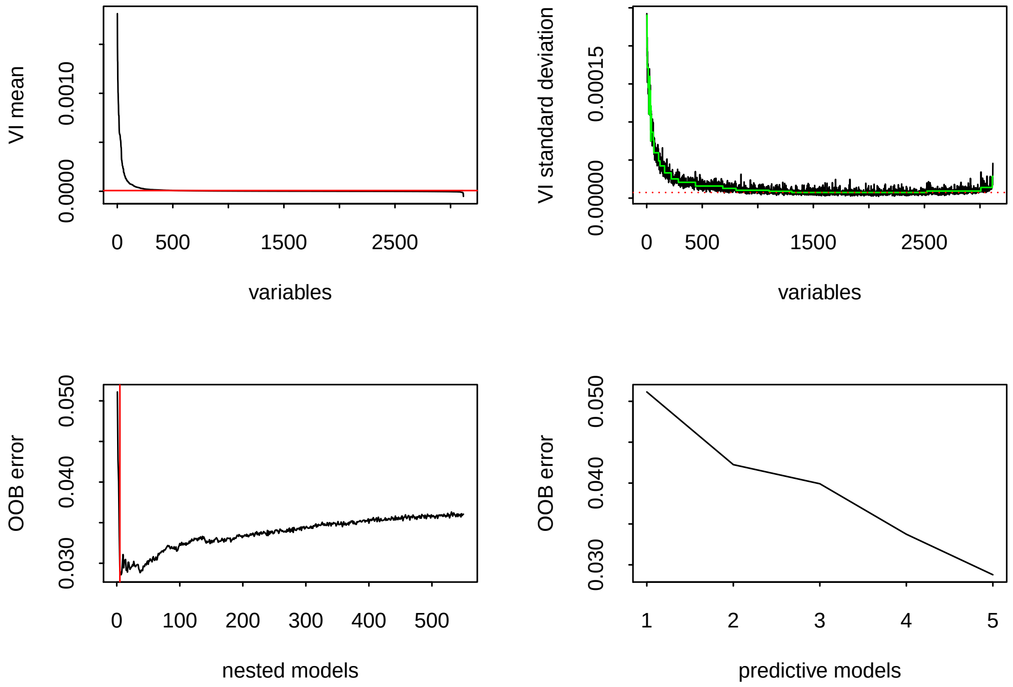

Figure 4: VSURF for toxicity data.

By considering the top left graph of Figure 4, it seems quite obvious to keep just a few variables. Indeed, the procedure leads to 5 variables, after both interpretation and prediction steps. Even if these numbers of selected variables are very small, they are not surprising if we refer to the study in González et al. (2012), where 12 variables were selected. It would be noted that, in general, our method failed to deliver all the variables related to the response variable in the case of numerous strongly correlated predictors. In addition, we do not select the same genes but our set of selected variables and the one in González et al. (2012) exhibit strong correlations.

Even if the results are quite similar, an advantage of using VSURF is that this procedure does not involve tuning parameters unlike the procedure developed in González et al. (2012). This difference is the main reason for the gap in computation time: several minutes with 40 cores for VSURF compared to several minutes with 1 core for González et al. (2012).

SRBCT data

The dataset we consider here will allow us to apply our procedure in a classification framework.

The real classification dataset is a small version of the small round blue cell tumors of childhood data and contains the expression measure of genes measured on 63 samples. This set is composed of:

- a data frame, called gene, of size 63 × 2308 which contains the 2308 gene expressions;
- a response factor of length 63, called class, indicating the class of each sample (4 classes in total).

These data, presented in details in (Khan et al., 2001), available in the R package **mixOmics**, have been widely studied but in most cases only 200 genes were considered and data have been transformed to reduce the problem to a regression problem (see e.g. Lê Cao and Chabrier, 2008). So as in Díaz-Uriarte and Alvarez De Andres (2006), we consider the 2308 genes and we deal directly with the classification problem, using **VSURF**.

```
> data("srbct", package = "mixOmics")
> set.seed(10131419, "L'Ecuyer-CMRG")
> vSRBCT <- VSURF(srbct$gene, srbct$class, parallel = TRUE, ncores = 40,
+                  clusterType = "FORK")

> summary(vSRBCT)

 VSURF computation time: 3.6 mins

 VSURF selected:
        676 variables at thresholding step (in 22.6 secs)
        25 variables at interpretation step (in 3 mins)
        13 variables at prediction step (in 13.6 secs)

 VSURF ran in parallel on a FORK cluster and used 40 cores
```

On this dataset, the procedure leads to 25 and 13 selected variables after the interpretation and prediction step respectively, and the selected variable sets are stable.

We can compare these results with those obtained in Díaz-Uriarte and Alvarez De Andres (2006) where the authors select respectively 22 genes on the original dataset and their number of selected variables is quite stable.

To get an idea of the performance of our procedure on the dataset, we perform an error rate estimation using an external 5-fold cross-validation scheme (meaning that we apply VSURF on each fold of the cross-validation). We obtain the following error rates[2] for interpretation and prediction sets respectively:

```
    interp       pred
0.01587302 0.07936508
```

The comparison with error rates evaluated using 200 bootstrap samples in Díaz-Uriarte and Alvarez De Andres (2006) suggests that our selections are reasonable.

Acknowledgements

We thank the editor and the three anonymous referees for their thorough comments and suggestions which really helped to improve the clarity of the paper.

Bibliography

K. J. Archer and R. V. Kimes. Empirical characterization of random forest variable importance measures. *Computational Statistics & Data Analysis*, 52(4):2249–2260, 2008. [p2]

R. Azen and D. V. Budescu. The dominance analysis approach for comparing predictors in multiple regression. *Psychological Methods*, 8(2):129–148, 2003. [p2]

A.-L. Boulesteix, S. Janitza, J. Kruppa, and I. R. König. Overview of random forest methodology and practical guidance with emphasis on computational biology and bioinformatics. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(6):493–507, 2012. [p1]

L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996. [p2]

L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. [p1, 2]

L. Breiman and A. Cutler. Random forest manual. 2004. URL http://www.stat.berkeley.edu/~breiman/RandomForests/cc_manual.htm. [p2, 3]

L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Chapman & Hall, New York, 1984. [p1]

[2] The script is available at https://github.com/robingenuer/VSURF/blob/master/Example/srbct_cv.R

P. Bushel, R. Wolfinger, and G. Gibson. Simultaneous clustering of gene expression data with clinical chemistry and pathological evaluations reveals phenotypic prototypes. *BMC Systems Biology*, 1(1): 15, 2007. [p11]

J. M. Cadenas, M. Carmen Garrido, and R. Martínez. Feature subset selection filter-wrapper based on low quality data. *Expert Systems with Applications*, 40(16):6241–6252, 2013. [p2]

N. Chèze, J.-M. Poggi, and B. Portier. Partial and recombined estimators for nonlinear additive models. *Statistical Inference for Stochastic Processes*, 6(2):155–197, 2003. [p10]

R. Diaz-Uriarte. GeneSrF and varSelRF: A web-based tool and R package for gene selection and classification using random forest. *BMC Bioinformatics*, 8(1):328, 2007. [p2]

R. Díaz-Uriarte and S. Alvarez De Andres. Gene selection and classification of microarray data using random forest. *BMC Bioinformatics*, 7(1):3, 2006. [p2, 13]

R. Genuer, J.-M. Poggi, and C. Tuleau. Random forests: Some methodological insights. 2008. arXiv:0811.3619. [p8]

R. Genuer, V. Michel, E. Eger, and B. Thirion. Random forests based feature selection for decoding fMRI data. In *Proceedings of COMPSTAT'2010 – 19th International Conference on Computational Statistics*, pages 1071–1078, 2010a. [p3]

R. Genuer, J.-M. Poggi, and C. Tuleau-Malot. Variable selection using random forests. *Pattern Recognition Letters*, 31(14):2225–2236, 2010b. [p1, 2, 4, 5, 9]

L. Gidskehaug, E. Anderssen, A. Flatberg, and B. K. Alsberg. A framework for significance analysis of gene expression data using dimension reduction methods. *BMC Bioinformatics*, 8(1):346, 2007. [p11]

I. González, K.-A. Lê Cao, M. J. Davis, and S. Déjean. Visualising associations between paired 'omics' data sets. *BioData Mining*, 5(1):1–23, 2012. [p11, 12]

B. Gregorutti, B. Michel, and P. Saint-Pierre. Correlation and variable importance in random forests. 2013. arXiv:1310.5726. [p2, 3]

I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3:1157–1182, 2003. [p2]

A. Hapfelmeier and K. Ulm. A new variable selection approach using random forests. *Computational Statistics & Data Analysis*, 60:50–69, 2012. [p2]

T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. Springer-Verlag, New York, 2001. [p1]

N. Heinloth, A, D. Irwin, R, A. Boorman, G, P. Nettesheim, D. Fannin, R, O. Sieber, S, L. Snell, M, J. Tucker, C, L. Li, S. Travlos, G, G. Vansant, P. E. Blackshear, R. W. Tennant, M. L. Cunningham, and R. S. Paules. Gene expression profiling of rat livers reveals indicators of potential adverse effects. *Toxicological Sciences*, 80(1):193–202, 2004. [p11]

T. Hothorn, K. Hornik, and A. Zeileis. Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical Statistics*, 15(3):651–674, 2006. [p1]

J. Khan, J. S. Wei, M. Ringner, L. H. Saal, M. Ladanyi, F. Westermann, F. Berthold, M. Schwab, C. R. Antonescu, C. Peterson, and P. S. Meltzer. Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. *Nature Medicine*, 7(6):673–679, 2001. [p12]

R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1):273–324, 1997. [p2]

M. B. Kursa and W. R. Rudnicki. Feature selection with the Boruta package. *Journal of Statistical Software*, 36(11):1–13, 2010. [p2, 11]

K.-A. Lê Cao and P. Chabrier. ofw: An R package to select continuous variables for multiclass classification with a stochastic wrapper method. *Journal of Statistical Software*, 28(9):1–16, 2008. [p2, 13]

K.-A. Lê Cao, D. Rossouw, C. Robert-Granié, and P. Besse. A sparse PLS for variable selection when integrating omics data. *Statistical Applications in Genetics and Molecular Biology*, 7(1), 2008. [p11]

K.-A. Lê Cao, I. Gonzalez, S. Dejean, with key contributors, F. Rohart, B. Gautier, contributions from, P. Monget, J. Coquery, F. Yao, and B. Liquet. *mixOmics: Omics Data Integration Project*, 2015. URL http://CRAN.R-project.org/package=mixOmics. R package version 5.0-4. [p11]

F. Leisch and E. Dimitriadou. *mlbench: Machine Learning Benchmark Problems*, 2010. URL http://cran.r-project.org/package=mlbench. R package version 2.1-1. [p9]

A. Liaw and M. Wiener. Classification and regression by randomForest. *R News*, 2(3):18–22, 2002. [p1]

G. Louppe, L. Wehenkel, A. Sutera, and P. Geurts. Understanding variable importances in forests of randomized trees. In *Advances in Neural Information Processing Systems*, pages 431–439, 2013. [p3]

N. Meinshausen and P. Bühlmann. Stability selection. *Journal of the Royal Statistical Society B*, 72(4): 417–473, 2010. [p9]

R. Nilsson, J. M. Peña, J. Björkegren, and J. Tegnér. Consistent feature selection for pattern recognition in polynomial time. *The Journal of Machine Learning Research*, 8:589–612, 2007. [p3]

A. Peters, T. Hothorn, and B. Lausen. ipred: Improved predictors. *R News*, 2(2):33–36, 2002. [p1]

F. Scheipl. spikeSlabGAM: Bayesian variable selection, model choice and regularization for generalized additive mixed models in R. *Journal of Statistical Software*, 43(14):1–24, 2011. [p2]

C. Strobl, A.-L. Boulesteix, A. Zeileis, and T. Hothorn. Bias in random forest variable importance measures: Illustrations, sources and a solution. *BMC Bioinformatics*, 8(1):25, 2007. [p2]

C. Strobl, A.-L. Boulesteix, T. Kneib, T. Augustin, and A. Zeileis. Conditional variable importance for random forests. *BMC Bioinformatics*, 9(1):307, 2008. [p2]

V. Svetnik, A. Liaw, C. Tong, and T. Wang. Application of Breiman's random forest to modeling structure-activity relationships of pharmaceutical molecules. In *Multiple Classifier Systems*, pages 334–343. Springer, 2004. [p9]

T. Therneau, B. Atkinson, and B. Ripley. *rpart: Recursive Partitioning and Regression Trees*, 2015. URL http://CRAN.R-project.org/package=rpart. R package version 4.1-9. [p1]

A. Verikas, A. Gelzinis, and M. Bacauskiene. Mining data with random forests: A survey and results of new tests. *Pattern Recognition*, 44(2):330–349, 2011. [p1]

H. R. Wehrens, M. Johan, and P. Franceschi. Meta-statistics for variable selection: The R package BioMark. *Journal of Statistical Software*, 51(10):1–18, 2012. [p2]

J. Weston, A. Elisseeff, B. Schölkopf, and M. Tipping. Use of the zero norm with linear models and kernel methods. *The Journal of Machine Learning Research*, 3:1439–1461, 2003. [p5]

Robin Genuer
University of Bordeaux
ISPED, Centre INSERM U-897-Epidemiologie-Biostatistique
33000 Bordeaux, France
and
INRIA Bordeaux Sud Ouest, SISTM team
33400 Talence, France
Robin.Genuer@isped.u-bordeaux2.fr

Jean-Michel Poggi
University of Orsay
Lab. Mathematics
bat 425
91405 Orsay, France
Jean-Michel.Poggi@math.u-psud.fr

Christine Tuleau-Malot
University of Nice Sophia Antipolis
CNRS, LJAD, UMR 7351
06100 Nice, France
Malot@unice.fr