Random Survival Forests for High-Dimensional Data

Hemant Ishwaran[1], **Udaya B. Kogalur**[1], **Xi Chen**[2] **and Andy J. Minn**[3]

[1]*Dept of Quantitative Health Sciences JJN3-01, Cleveland Clinic, Cleveland OH 44195*

[2]*Dept of Biostatistics, Vanderbilt University School of Medicine, Nashville, TN 37232-6848*

[3]*Dept of Radiation Oncology, University of Pennsylvania, Philadelphia, PA 19104*

Received 28 January 2010; revised 20 September 2010; accepted 22 November 2010
DOI:10.1002/sam.10103
Published online 11 January 2011 in Wiley Online Library (wileyonlinelibrary.com).

Abstract: Minimal depth is a dimensionless order statistic that measures the predictiveness of a variable in a survival tree. It can be used to select variables in high-dimensional problems using Random Survival Forests (RSF), a new extension of Breiman's Random Forests (RF) to survival settings. We review this methodology and demonstrate its use in high-dimensional survival problems using a public domain R-language package `randomSurvivalForest`. We discuss effective ways to regularize forests and discuss how to properly tune the RF parameters 'nodesize' and 'mtry'. We also introduce new graphical ways of using minimal depth for exploring variable relationships. © 2011 Wiley Periodicals, Inc. Statistical Analysis and Data Mining 4: 115–132, 2011

Keywords: forests; maximal subtree; minimal depth; trees; variable selection; VIMP

1. INTRODUCTION

Survival outcome, such as time until death, is often used as a powerful endpoint for studying disease processes. Traditionally, survival data have been used to predict outcome for patients based on clinical variables, but recently there has been tremendous interest in combining survival outcomes with data from high-throughput genomic technologies, especially DNA microarrays, as a way of studying disease processes at a molecular level.

Although the potential for discovery can be far greater when using genome-wide data, the high-dimensionality of this data poses challenging statistical issues. Various methods have been proposed to overcome these hurdles. These include partial least squares [1,2], semi-supervised principal components [3], and Cox-regression under lasso penalization [4,5]. As well, there has been considerable interest in applying machine learning methods. Boosting, in particular, has been considered in a variety of contexts; for example, Cox-gradient boosting [6,7], L_2-boosting [8], and componentwise Cox-likelihood based boosting [9]. See the study of Ridgeway [10] for the first instance of boosting for Cox models.

Another machine learning method that has attracted considerable interest is Random Forests (RF) [11]. RF is an ensemble learner formed by averaging base learners. The base learners are typically binary trees grown using CART methodology [12]. In traditional CART, a binary tree is grown to full size and then pruned back on the basis of a complexity measure. However, RF trees differ from classical CART as they are grown nondeterministically, without pruning, using a two-stage randomization procedure. First, an independent bootstrap sample of the data is drawn over which the tree is grown. Second, during the tree growing process, at each node of the tree, a subset of variables is randomly selected and these are used as candidates to split the node. The tree is grown to full, or near full size, so that each terminal node contains a minimum number of unique cases. Trees grown using CART methodology are well known to exhibit high variance [13,14]. However, by combining deep, random trees, RF is able to substantially reduce variance as well as bias.

Most applications of RF have focused on regression and classification settings. Many of these use the popular Breiman and Cutler software implemented in R [15]. See

Correspondence to: Hemant Ishwaran (hemant.ishwaran@gmail.com)

© 2011 Wiley Periodicals, Inc.

the studies of Lunetta *et al.* [16], Bureau *et al.* [17], and Diaz-Uriarte and Alvarez de Andres [18] for applications to Single Nucleotide Polymorphism (SNP) and microarray data. Recently, however, RF has been extended to right-censored survival settings. This new method called Random Survival Forests (RSF) [19] can be implemented using the R-software package `randomSurvivalForest` [20,21]. Similar to regression and classification settings, RSF is an ensemble learner formed by averaging a tree base-learner. In survival settings, the base-learner is a binary survival tree, and the ensemble is a cumulative hazard function formed by averaging each tree's Nelson–Aalen's cumulative hazard function.

More recently, Ishwaran *et al.* [22] described a new high-dimensional variable selection method based on a tree concept referred to as *minimal depth*. This differs from the traditional method of variable selection in RF which has been based on variable importance (VIMP) measures. VIMP measures the increase (or decrease) in prediction error for the forest ensemble when a variable is randomly 'noised up' [11]. While this informal idea first proposed by Breiman [11] has had good empirical success, the complex nature of the VIMP calculation has made it difficult to study [23], and the method has remained largely ad hoc. In contrast, rigorous theory for selecting variables as well as comprehensive methodology for regularizing forests is possible using minimal depth [22].

One of the main purposes of this paper is to familiarize the reader with this new methodology and to demonstrate how to effectively use it in high-dimensional survival problems, such as those frequently encountered in bioinformatics settings. Section 2 begins by reviewing the definition of minimal depth and minimal depth thresholding for variable selection. Sections 3 and 4 discuss regularization of forests through selection of the tuning parameters 'nodesize' and 'mtry'. Section 5 investigates the effectiveness of this regularization on minimal depth variable selection. In Section 6, we discuss an ultra-high dimensional regularization method, termed 'variable hunting'. Section 7 introduces new graphical methods using minimal depth and related tree structures for exploring relationships between variables. Although we focus on a low-dimensional problem for illustration we believe that these ideas could be utilized in high-dimensional problems. In Section 8, we summarize the principle findings of the paper. Finally, we note that all RSF computations in this paper have been implemented using the `randomSurvivalForest` package which has been extended to incorporate minimal depth methodology [21]. For the convenience of the readers, we have included small snippets of R-code to illustrate basic calls for implementing the methodology.

2. MINIMAL DEPTH

2.1. Definition

Minimal depth assesses the predictiveness of a variable by its depth relative to the root node of a tree. To make this idea more precise, we define the *maximal subtree* for a variable. A maximal subtree for a variable v is defined to be the largest subtree whose root node is split using v (i.e., no other parent node of the subtree is split using v). The shortest distance from the root of the tree to the root of the closest maximal subtree of v is the minimal depth of v. A smaller value corresponds to a more predictive variable [22].

Figure 1 provides an illustration. Shown is a single tree grown using the Primary Biliary Cirrhosis (PBC) data, an example data set available in the `randomSurvivalForest` package (hereafter abbreviated as `rsf`). Maximal subtrees for variables 'bili' and 'chol', two of the 17 variables in the data set, are highlighted using yellow and green, respectively. Tree depth is indicated by an integer value located in the center of a tree node. For example, on the right-hand side of the tree, chol is split at depth 2. In fact, this node and the subtree beneath it defines a maximal subtree for chol because no parent node above this node splits on chol. Although chol has other maximal subtrees (in the middle and left side), the shortest distance to the root node occurs for this particular maximal subtree. Thus, chol has a minimal depth of 2. Note that bili is also split for the first time on the right-hand side within the maximal subtree for chol (at depth 3). Thus, within this subtree is a maximal subtree for bili. There are other maximal subtrees for bili, in fact there is one with a depth of 3 near the middle of the tree. Scanning the tree, we see that the minimal depth for bili is 3.

2.2. Distribution Theory for Minimal Depth

The distribution for the minimal depth can be derived in closed form under certain conditions [22]. Let D_v denote the minimal depth for a variable v. Let $\pi_{v,j}(t)$ be the probability that v is selected as a candidate variable for splitting a node t of depth j, assuming no maximal v-subtree exists at depth less than j. Let $\theta_{v,j}(t)$ be the probability that v splits a node t of depth j given that v is a candidate variable for splitting t, and that no maximal v-subtree exists at depth less than j. Then for $0 \le d \le D(T)-1$, where $D(T)$ is the depth of the tree,

$$\mathbb{P}\Big\{D_v = d \big| \ell_0, \ldots, \ell_{D(T)-1}\Big\} = \left[\prod_{j=0}^{d-1}\big(1-\pi_{v,j}\theta_{v,j}\big)^{\ell_j}\right]\left[1-\big(1-\pi_{v,d}\theta_{v,d}\big)^{\ell_d}\right], \quad (1)$$

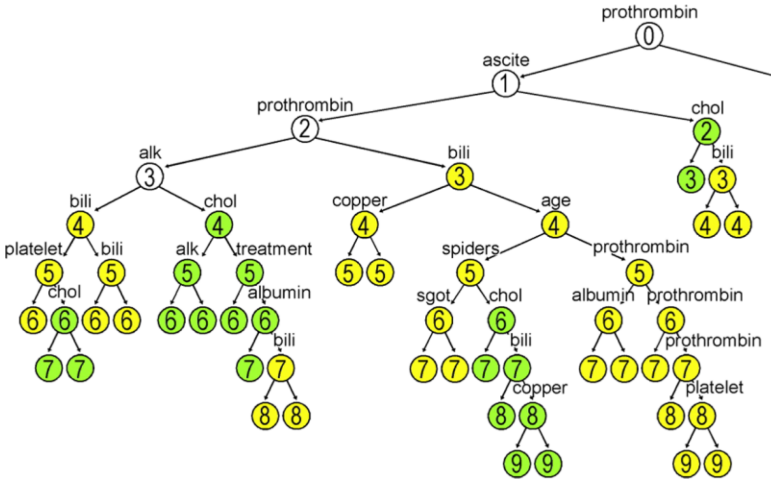

Fig. 1 Illustration of minimal depth (based on PBC data, an example data set available in the `randomSurvivalForest` package). Yellow and green colored points are maximal subtrees for variables ‘bili’ and ‘chol’, respectively. Depth of the tree is indicated by numbers $0, 1, \ldots, 9$ inside of each node. The minimal depth for bili is 3 and chol is 2.

where ℓ_d is the number of nodes at depth d. The distribution (1) is normalized by setting

$$\mathbb{P}\Big\{D_v = D(T)\Big|\ell_0, \ldots, \ell_{D(T)-1}\Big\}$$
$$= 1 - \sum_{d=0}^{D(T)-1} \mathbb{P}\Big\{D_v = d\Big|\ell_0, \ldots, \ell_{D(T)-1}\Big\}.$$

The above representation assumes that $\pi_{v,j}(t)$ and $\theta_{v,j}(t)$ are independent of the node t. As discussed by Ishwaran *et al.* [22] this assumption can hold in many scenarios. One especially important case is when v is a noisy variable in high-dimensional sparse settings.

DEFINITION 1: We call a variable noisy if it has no effect on the true survival distribution. A variable that effects the survival distribution is called strong.

Let m be the number of candidate variables randomly selected for splitting a node, and let p be the total number of variables. The distributional result (1) holds for a noisy variable if the following two conditions are met:

$$\pi_{v,j}(t) = \pi_{v,j} = \frac{m}{p}, \tag{2}$$

$$\theta_{v,j}(t) = \theta_{v,j} = \frac{1}{m}. \tag{3}$$

Condition (2) holds approximately if

$$\frac{m}{p} = o(1). \tag{4}$$

The value m is often referred to as *mtry* and is an important tuning parameter used in forests. Its value regulates the amount of correlation between trees. Smaller values encourage different sets of variables to be used when growing a tree which improves tree decorrelation. Typically m is selected so that $m = \sqrt{p}$, and while this satisfies Eq. (4), we will show (see Section 4) that much larger values are required for effective variable selection in high-dimensional settings. Larger values are needed to improve the chance of splitting a strong (non-noisy) variable.

Notice that Eq. (2) implies that each of the m candidate variables is randomly selected from the full set of p variables. Thus, among the m candidates, we would expect mp_0/p to be strong variables, where $1 \leq p_0 \leq p$ is the total number of strong variables. Now, because a strong variable will be $S_j > 0$ times more likely to be selected than a noisy variable, then if v is noisy, we have

$$\theta_{v,j} \approx \left[m\left(1 - \frac{p_0}{p}\right) + mS_j\,\frac{p_0}{p}\right]^{-1} \approx \frac{1}{m}$$

because $p_0/p \ll 1$ in sparse settings. Thus, condition (3) naturally holds in sparse settings.

Hence, in sparse high-dimensional settings, conditions (2) and (3) will hold if v is a noisy variable and if Eq. (4) is satisfied. Assuming that this is the case, the distribution (1) can be approximated in closed form for a noisy variable as follows:

$$\mathbb{P}\{D_v = d|v \text{ is a noisy variable}, \ell_0, \ldots, \ell_{D(T)-1}\}$$
$$= \left(1 - \frac{1}{p}\right)^{L_d}\left[1 - \left(1 - \frac{1}{p}\right)^{\ell_d}\right], \quad 0 \leq d \leq D(T) - 1,$$

$$\mathbb{P}\Big\{D_v = D(T) \big| v \text{ is a noisy variable}, \ell_0, \ldots, \ell_{D(T)-1}\Big\},$$
$$= 1 - \sum_{d=0}^{D(T)-1} \left(1 - \frac{1}{p}\right)^{L_d} \left[1 - \left(1 - \frac{1}{p}\right)^{\ell_d}\right], \quad (5)$$

where $L_d = \ell_0 + \cdots + \ell_{d-1}$. See the study of Ishwaran *et al.* [22] for details.

2.3. Minimal Depth Thresholding

Having a simple expression (5) for the minimal depth distribution is highly advantageous. The mean of this distribution, the *mean minimal depth*, easily calculated from Eq. (5), represents a threshold value for selecting variables in sparse high-dimensional settings. Those variables with forest averaged minimal depth exceeding the mean minimal depth threshold are classified as noisy and are removed from the final model. Definition 2 gives a formal description of this selection method.

DEFINITION 2: Choose a variable v if its forest averaged minimal depth, $D_v = d_v$, is less than or equal to the mean minimal depth of D_v under the null (5). We refer to this variable selection method as (mean) minimal depth thresholding.

Minimal depth thresholding is a simple and rigorous way to select variables in high dimensions. We list two of its key properties.

1. Because minimal depth is independent of prediction error, variables selected using mean minimal depth are not tied to any specific measure of error. This is important as prediction error can vary greatly in survival settings depending on the method used. For example, the amount of censoring can play a prominent role.

2. Because minimal depth utilizes only generic tree concepts, it applies to all forests regardless of the outcome. For example, it can be used in regression as well as classification settings.

Being able to select variables independently of prediction error is important. In some scenarios (such as survival settings), there may be debate about what constitutes an appropriate measure of prediction error, and as we remarked, different measures could yield different estimates of prediction error which ultimately could yield different selected variables. Equally importantly, there are scenarios where it may not even be clear *how* to measure prediction error. For example, in competing risks where the outcome is time to the event of interest, but where the event can be precluded by other types of events, assessing prediction error and defining VIMP is a challenging research problem by itself [24].

Furthermore, variable selection based on prediction error implicitly relies on having an accurate ensemble. While forests are known to be excellent predictors in high dimensions, without additional regularization, the number of noisy variables eventually overwhelms a tree as p increases, and prediction performance subsequently breaks down. When this happens it becomes impossible to effectively select variables. This is true regardless of the method used for variable selection, but methods that are based on prediction error, such as VIMP, may be more susceptible to these effects than minimal depth. As long as a variable v repeatedly splits across the forest, minimal depth has a good chance of identifying v even in the presence of many noisy variables.

Even when a forest provides reasonable VIMP, there is still the issue of thresholding these values. Ad hoc methods that use arbitrary cut-off values will not work in high dimensions as VIMP varies considerably depending upon the data; for example, the size of p, the correlation in the data, and the underlying sparsity p_0/p of the true model play crucial roles. It is important, therefore, to have a method for reliably thresholding variables. Furthermore, this applies even to methods such as stepwise regression, a type of regularization that has become popular as a way to compensate for the inability to directly threshold VIMP in high dimensions. In these methods, VIMP is used to order variables. Models are then fit based on this ordering in either a forward or backward stepwise manner, with the final model being selected on the basis of prediction error. See, for example, the studies of Diaz-Uriarte and Alvarez de Andres [18] and Genuer *et al.* [25]. Our contention is that such methods could be used far more effectively for selecting variables if we could reliably threshold noise variables. Indeed, in Section 6, we discuss an ultra-high dimensional stepwise selection mechanism (variable hunting) that combines the regularization of stepwise with the utility of minimal depth thresholding.

Figure 2 illustrates some of these ideas. There we have plotted VIMP based on a `rsf` analysis of simulated data, where $n = 200$, $p = 250, 500, 1000$, and $p_0 = 25$. In each simulation, the signal strength for the strong variables was set to the same value. Noisy variables were drawn independently from a uniform $[0, 1]$ distribution, while strong variables were sampled jointly so that they were pairwise correlated (see Section 5 for details). This simulation is intended to represent the type of correlated data one often encounters in high-throughput genomic problems. For example, gene expression values assayed

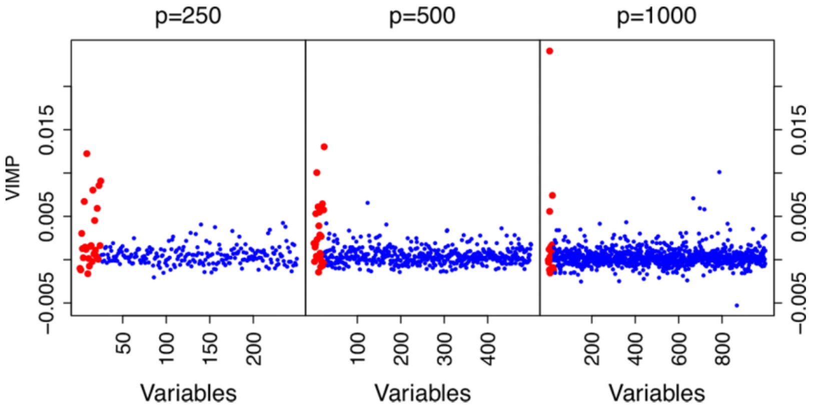

Fig. 2 VIMP from RSF applied to simulated data with increasing p where $n = 200$ and $p_0 = 25$. Strong variables (indicated by red) are pairwise correlated.

using microarrays may represent pathways and gene networks which naturally exhibit a high level of correlation.

Two things should be noticed about Fig. 2. The first is that the separation between noisy and strong variables degrades noticeably as p increases. This is because prediction error decreases rapidly with p (for $p = 1000$ the out-of-bag error rate was 39%, while for $p = 250$ it was 34%). Second, VIMP decreases in magnitude as p increases. Even going from $p = 500$ to $p = 1000$ leads to a noticeable change in the range of VIMP. A good thresholding cut-off rule for $p = 500$ would not necessarily work well for $p = 1000$. It is clear that arbitrary cut-off values would work poorly in this problem.

It would be extremely beneficial to be able to threshold variables in scenarios like Fig. 2. In the next few sections, we discuss how to do this effectively by using minimal depth thresholding.

3. SELECTING NODESIZE

3.1. Mean Minimal Depth under Balanced Trees

To use minimal depth we must know the number of nodes, ℓ_d, at each depth d of a random tree to be able to calculate (5). Because these values are unknown, a strategy that we first tried was to assume a balanced tree in which $\ell_d = 2^d$ for each d. Figure 3 shows how the mean minimal depth under the null (5) varies as a function of p assuming a sparse setting and a balanced tree.

As one can see from Fig. 3, the mean minimal depth converges to the maximal depth of a tree, $D(T)$ as p increases. The vertical lines and superimposed integers in the figure indicate the number of variables at which point the mean plus one-half of the standard deviation of D_v exceeds $D(T)$. As can be seen, this can occur quite rapidly.

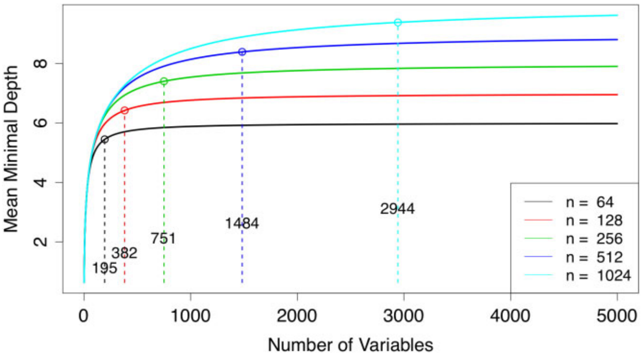

Fig. 3 Mean minimal depth under the null that a variable is noisy assuming sparsity and a balanced tree. Dashed vertical lines and superimposed integers indicate the number of variables at which point the mean plus one-half of the standard deviation of minimal depth exceeds the maximum depth of a balanced tree.

For example, with a sample size of $n = 256$, it only takes $p = 751$ variables to reach this point.

3.2. Random Trees Are Unbalanced in High Dimensions

What Fig. 3 demonstrates is that D_v will be nearly equal to $D(T)$ for a noisy variable if p is substantially larger than n under a balanced tree assumption. However, we now discuss how realistic this latter assumption is.

In fact, we found through experimentation that balancedness often failed to hold in high-dimensional sparse settings and that minimal depth often saturated at values far higher than $D(T) = \log_2(n)$, the asymptote for a balanced tree in such settings. Thus, while Fig. 3 shows that the effectiveness of minimal depth decreases rapidly with p for a balanced tree, the story may actually be much better when a tree is unbalanced.

What we found was that the balancedness of a tree was heavily influenced by the forest parameter *nodesize* defined as the minimum number of unique deaths within a terminal node. We found that trees became highly unbalanced when nodesize was set to a very small value. Intuitively, this can be understood as follows. In high-dimensional sparse settings, each node of a tree can be split by a large number of variables, many of these being noisy. Due to their overwhelming majority, a noisy variable will often split a node, and when this happens there is a non-negligible probability that a split will occur near an edge. Furthermore, this effect can become exaggerated due to the tendency of some splitting rules to split near their edge when the variable is noisy [12]. This can result in a daughter with close to but greater than or equal to nodesize unique deaths rather than, say, a daughter with approximately half the population of the parent node. When an edge split such as this occurs, any further splitting is impossible because it would result in less than nodesize unique deaths in the second-generation daughters of the node. Thus, the daughter node is terminated.

Clearly nodesize plays a prominent role in all of this. Because a noisy variable can split near its edge, nodes are terminated very early in the tree growing process when nodesize is large. On the other hand, if nodesize is small, the tree is able to grow deeply, but again because noisy variables often split nodes, the tree becomes unbalanced since there is always a non-negligible probability of splitting near an edge.

The following result makes these assertions rigorous and suggests a way to regularize forests.

THEOREM 1: Grow a random survival tree such that each terminal node contains a minimum of $1 \leq N < n$ unique deaths. Let t be a node with $1 \leq n_t \leq n$ observations. Let $\pi_t(d)$ be the probability that no daughter node under t of relative depth d is a terminal node if t is split repeatedly using ordered noisy variables. Then

$$\pi_t(d) \leq \begin{cases} \displaystyle\prod_{j=1}^{d}\left(1-\frac{4N-2}{n_t-2(j-1)N}\right)^{2^{j-1}} & \text{if } n_t \geq 2^{d+1}N \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

Nodesize is represented in Theorem 1 by N. Typically nodesize is used as a parameter for tuning prediction error, with smaller values of nodesize yielding deeper trees, lower bias, and in some instances, a lower mean-squared error. However, Theorem 1 shows that nodesize also plays a pivotal role in the balancedness of a tree. As an example, if t is the root node, $n_t = 256$, and $d = 4$, then $\pi_t(d) = 0.88$ if $N = 1$, whereas if $N = 5$, $\pi_t(d) = 0.29$. Thus when $N = 5$, terminal nodes occur with high probability (71%) after only a few noisy splits. This probability is much lower (12%) when $N = 1$, yet the probability is still non-negligible. This is important, because while we would not expect to find terminal nodes so close to the root node, under noisy splitting this is clearly not the case.

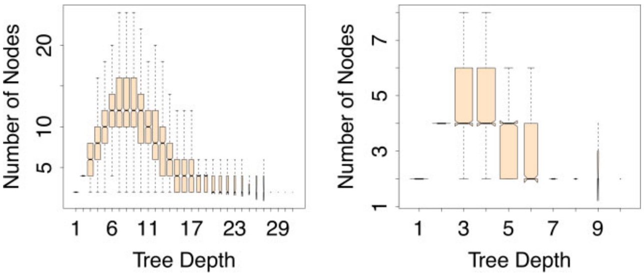

Fig. 4 Forest distribution of number of nodes at a given tree depth for PBC data with 1000 noise variables (boxplot widths are proportional to the square-root of the number of observations). Left- and right-hand sides were fit using a nodesize of $N = 1$ and $N = 5$, respectively.

Thus, Theorem 1 shows that large N will lead to unbalanced premature trees, while decreasing N will encourage deep unbalanced trees. This suggests that N should be chosen as small as possible. This will encourage deep trees and ensure that minimal depth does not saturate rapidly.

As evidence of this, we added 1000 uniform $[0, 1]$ variables to the PBC data and then fit this data using the `rsf` package. Default settings for the software were used except that nodesize was set to the value 1. Additionally, each tree was grown using log-rank splitting [26] with randomization. For each of the randomly selected variables, 'nsplit' randomly selected split points were chosen, and log-rank splitting was applied to these random split points. The tree node was split on that variable and random split point maximizing the log-rank statistic. We set nsplit $= 10$. Random splitting is a useful tool for mitigating tree bias favoring splitting on continuous variables and it can improve prediction error [19,27].

The left-hand side of Fig. 4 presents boxplots depicting the forest distribution for the number of nodes at a given tree depth when $N = 1$. In this data set, there are $n = 276$ observations, and if trees were balanced, we would expect tree depths to rarely exceed $\log_2(276) = 8.1$. Yet, the figure shows that trees can often have depths in excess of 20. Furthermore, depth ranges widely from about 3 to over 20 with the number of nodes at any given depth being generally much smaller than 2^d; the value expected for a balanced tree. This should be contrasted with the figure on the right-hand side based on a similar analysis, but using a nodesize of $N = 5$. Note how the distribution concentrates at much smaller depth values, with a range of values from about 3 to 6.

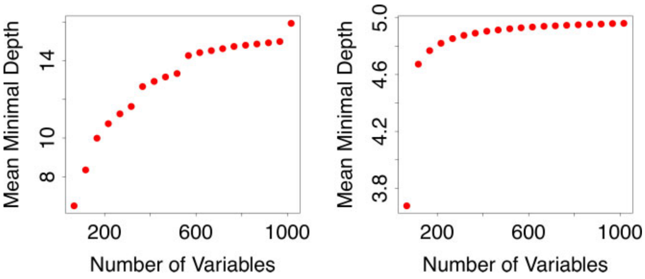

Fig. 5 Mean minimal depth for a noisy variable from Eq. (5) using forest estimates for ℓ_d and $D(T)$. Left- and right-hand sides based on nodesize values of $N = 1$ and $N = 5$, respectively.

3.3. Forest Estimates for the Number of Nodes and Tree Depth

While it is convenient to assume a balanced tree so that ℓ_d is a known value, our results clearly show that such an assumption is unrealistic in high dimensions when nodesize is small. Even when nodesize is large, it is unlikely that a balanced tree assumption will be appropriate.

Thus, to apply Eq. (5), we estimate ℓ_d as well as tree depth, $D(T)$, using forest estimated averaged values. These calculations can be easily programed, but for convenience, they have been incorporated into the R-wrapper `max.subtree` available in the `rsf` package. To extract these estimates, we first must make an initial forest call. Our call looked something like:

```
rsf.out <- rsf(rsf.f, pbc.noise,
 ntree=1000, nsplit=10, nodesize=1,
 forest=TRUE, big.data=TRUE)
```

The following commands extract minimal depth data from the forest object, rsf.out:

```
max.out <- max.subtree(rsf.out)
nodes.at.depth <- max.out$nodesAtDepth
threshold <- max.out$threshold
min.depth <- max.out$order[, 1]
```

The first command invokes the `max.subtree` wrapper which recursively parses the forest and gathers topological information from each tree. The object obtained from this call contains many useful items. One of these is the number of nodes per depth per tree (line 2) that was used to produce Fig. 4. Another is the mean minimal depth under the null (line 3). This value is estimated by substituting forest estimates for ℓ_d and $D(T)$ in Eq. (5). Line 4 extracts the forest averaged minimal depth for each variable. Figure 5 shows how mean minimal depth varies as the number of noise variables is increased for the PBC data. Observe how mean minimal depth asymptotes at much higher values under a nodesize of $N = 1$ (left-hand figure) than that for a nodesize of $N = 5$ (right-hand figure).

4. SELECTING MTRY

In this section, we look at the issue of selecting mtry. Similar to nodesize, mtry is a RF tuning parameter that plays a crucial role in accurate variable selection.

4.1. Mean Minimal Depth for Strong Variables

The default setting for mtry in the `rsf` package is $m = \sqrt{p}$. While this satisfies the condition (4) needed for Eq. (5) to hold, we now show that this value may be too small for effective variable selection in sparse settings.

To do so, we study the effect mtry has on the distribution of minimal depth for a strong variable. To derive this distribution we make the following assumptions:

1. $\pi_{v,j} = m/p$ for each v.
2. If v is a strong variable, $m\theta_{v,j} = \min(S\sqrt{n/\ell_j}, m)$.

Condition 1 assumes that all variables are equally likely to be selected as candidates for splitting a node. This is reasonable when p is large. Condition 2 states that the probability of splitting a node using a strong variable equals S/m times the square-root of the sample size of the node, $n_j = n/\ell_j$ (here S is the signal strength). This is realistic because we would expect any good splitting rule to have a $\sqrt{n_j}$-asymptotic property. Note that condition 2 implicitly assumes that the sample size for each node at depth j is equal; that is, that all daughters of the same depth are equally balanced in size. Of course this will not always hold in light of our previous discussions on the unbalanced nature of trees in high dimensions. But the assumption is stronger than needed. As long as the size of a node is appreciable whenever a strong variable splits a node, our distributional result will hold approximately.

Under these assumptions, the distribution for minimal depth can be derived in closed form for a strong variable analogous to Eq. (1):

$$\begin{aligned}\mathbb{P}\{D_v = d \mid v \text{ is a strong variable}, \ell_0, \dots, \ell_{D(T)-1}\} \\ = \left(1 - \frac{\min(S\sqrt{n/\ell_d}, m)}{p}\right)^{L_d} \\ \left[1 - \left(1 - \frac{\min(S\sqrt{n/\ell_d}, m)}{p}\right)^{\ell_d}\right], \qquad (7)\end{aligned}$$

where $0 \le d \le D(T) - 1$. The distribution is normalized at $D(T)$ as before.

Figure 6 shows the mean minimal depth under Eq. (7) for the PBC data using different values of mtry: $\sqrt{p}$, $p^{2/3}$, $p^{3/4}$, and $p^{4/5}$ (square points connected by dashed-lines). For nodesize we used $N = 1$. All other parameter settings were the same as before. For the analysis, we used $S = 1, 20$

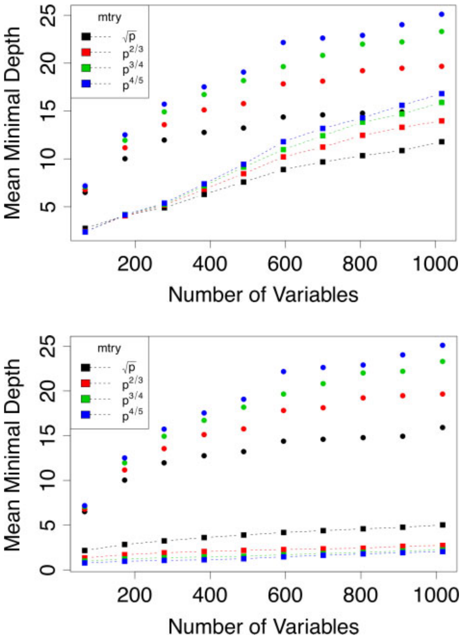

Fig. 6 Mean minimal depth for a noisy variable (circles) and a strong variable (square points connected by lines) for PBC data using forest estimates for ℓ_d and $D(T)$ as number of noise variables and mtry values are varied. Top figure based on a signal strength of $S = 1$; bottom figure $S = 20$.

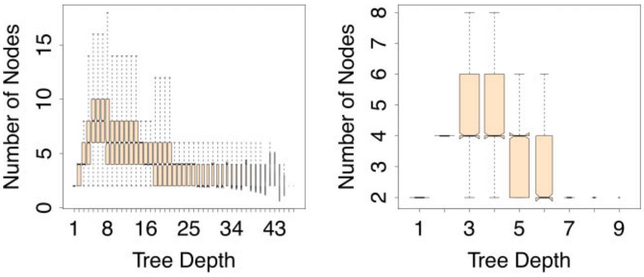

Fig. 7 Same as Fig. 4, but using an mtry value of $m = p^{4/5}$. Left- and right-hand sides correspond to nodesize values of $N = 1$ and $N = 5$, respectively.

(top and bottom figures, respectively) and as in Fig. 5, we have substituted forest estimates for ℓ_d and $D(T)$. For convenience, we have also superimposed the mean minimal depth under the null (circles).

The top plot of Fig. 6 shows that when the signal strength is weak, increasing mtry increases the mean minimal depth under the alternative. With a weak signal a large mtry value allows too many noisy variables to compete to split a node. Even though the mean minimal depth under the null increases, any benefit of increasing mtry appears to be washed out. Indeed, one can show that for very small signal values, mean minimal depth under the alternative can exceed that under the null if mtry is too large. On the other hand, the bottom figure shows that increasing mtry under a strong signal improves *both* the alternative and null distribution.

It is interesting that the mean minimal depth under the null increases with increasing mtry. One explanation for this is as follows. As mtry increases, the chance of splitting a noisy variable increases. Because splits on noisy variables yield unbalanced daughter nodes, the distribution of ℓ_d becomes more spread out and subsequently the mean value for minimal depth under the null increases with mtry.

To demonstrate this effect, we redid the analysis of Fig. 4 using an mtry value of $m = p^{4/5}$ in place of the default value $m = \sqrt{p}$ used previously. Figure 7 displays the results. Compared with Fig. 4, one can see that when $N = 1$ (left-hand side figures) tree depths are far larger and the distribution of tree depths are far more evenly spread. As the PBC data are known to have strong signal variables, and we know from Fig. 6 that the mean minimal depth will be relatively small assuming a strong signal under the alternative, we can conclude that Fig. 7 (left side) is at the very least understating the skewness of the null distribution and that the skewness seen in the figure is primarily driven by the effect of unbalanced splitting of noisy variables which is exaggerated by the large mtry value. This effect is also dependent on a small nodesize value. This can be seen from the right-hand side of Fig. 7, which is based on a node size of $N = 5$. This looks nearly the same as the right side of Fig. 4.

These results suggest that mtry should be set as high as possible in strong signal scenarios as this will encourage the best possible separation between the null and alternative minimal depth distributions. In moderate signal settings, there may be no benefit to using a very large mtry value. In very weak signal scenarios, a large mtry may even be detrimental.

Thus if the goal is to discover signal, then the best overall strategy is to choose a large mtry value, while maintaining the condition $m/p = o(1)$. The latter condition is designed to avoid poor performance in weak signal scenarios. Choosing a large mtry value may seem like a curious strategy, however, because one of the core principles of RF is that the tree growing process should encourage decorrelated trees. Decorrelated trees are desirable because they ensure low variance. With increasing mtry, trees are grown under less randomization and one might expect that correlation between trees would increase because of this.

However, we do not believe that this is what happens. In high dimensions, trees can maintain low correlation due to the sheer volume of noise variables. In fact, given any two trees, there will be little overlap in the noise variables used to split their nodes. Thus in high-dimensional sparse settings, we conjecture there is a built in decorrelation effect at work. Furthermore, one should keep in mind that the goal of using minimal depth is to achieve accurate variable selection and not necessarily to minimize prediction error, so that even if correlation degrades by increasing mtry, this may be necessary in order to ensure accurate selection of variables.

4.2. Encouraging Strong Variables

In addition to increasing mtry, another way to increase the number of nodes split by strong variables is to actively encourage their selection. Rather than selecting mtry variables randomly, it is better to select variables using predetermined weights indicating their potential importance. This idea has been explored in a related approach referred to as 'enriched random forests' [28].

It is not necessary to use complicated methods when selecting the weights. The only constraint is that conditions (2) and (3) must continue to hold in order for Eq. (5) to be valid. Consider condition (2). Let $W_j \geq 0$ be the weight for selecting variable v_j, for $j = 1, \ldots, p$. Suppose that v_k is noisy and let Π_k be the conditional probability that v_k is selected from $\{v_1, \ldots, v_p\}$ in a weighted-random draw, conditional on the values $(W_j)_{j=1}^p$. Then

$$\Pi_k = \frac{W_k}{\sum_{j \text{ is noisy}} W_j + \sum_{j \text{ is strong}} W_j}.$$

We will assume that $1 \leq W_j \leq \lambda_p$, for $j = 1, \ldots, p$. Then,

$$\frac{W_k}{\sum_{j \text{ is noisy}} W_j + p_0\lambda_p} \leq \Pi_k \leq \frac{W_k}{\sum_{j \text{ is noisy}} W_j + p_0}.$$

If $p_0\lambda_p/p \to 0$ and W_j are i.i.d. when j is noisy, then multiplying throughout by p, and using the strong law of large numbers, we have

$$p\Pi_k = \frac{W_k}{\mu} + o_p(1), \quad \text{as } p \to \infty,$$

where $\mu = \mathbb{E}(W_k)$ is the mean for a noisy variable. Let π_k be the unconditional probability that v_k is selected from $\{v_1, \ldots, v_p\}$ in a weighted-random draw. From the above, and noting that Π_k is bounded, it follows that

$$\pi_k = \mathbb{E}\left(\frac{W_k}{p\mu}\right) + o_p(1/p) = 1/p + o_p(1/p).$$

Therefore the probability that v_k is chosen as a candidate variable for splitting a node is approximately

$$\begin{aligned} 1 - (1 - \pi_k)^m &= 1 - (1 - 1/p + o_p(1/p))^m \\ &= m/p + o_p(1). \end{aligned}$$

For condition (3), notice that the preceding calculation shows that each noisy candidate variable can be assumed to be randomly selected from the p variables. In particular, this implies that a strong variable will also have a $1/p$ chance of being selected as a candidate. Thus, condition (3) holds by the same argument given earlier.

It may seem surprising that in trying to improve the chance of selecting a strong variable that the probability is still no better than a random draw. But keep in mind that this is only an asymptotic result. The key condition $p_0\lambda_p/p \to 0$ used in our asymptotic argument can be satisfied by many sequences. If λ_p is allowed to increase rapidly enough, then the effect can be substantial in finite sample settings. We demonstrate this in the following section.

5. THRESHOLDING USING MEAN MINIMAL DEPTH

In this section, we study the effectiveness of minimal depth thresholding under regularization. For proper regularization, we have shown that nodesize should be set as close to $N = 1$ as possible and mtry should be chosen as large as possible so that $m/p \to 0$. We have also discussed regularization by encouraging strong variables to be selected by using predetermined weights. These weights should be selected so that $p_0\lambda_p/p \to 0$. In the following simulation examples, we study the effectiveness of combining these three regularization methods.

5.1. Selection of Weights

We used the following strategy for defining $(W_k)_{k=1}^p$. Prior to fitting a forest, we fit a univariate Cox regression analysis for each variable v_k. Using the p-value, P_k, for the coefficient estimate for v_k, we defined

$$W_k = \frac{1}{\max(P_k, \lambda_p^{-1})}, \quad k = 1, \ldots, p, \tag{8}$$

where $\lambda_p \geq 1$. Notice that W_k are identically distributed for noisy variables since P_k is uniformly distributed under the assumption of a zero coefficient. Also:

$$1 \leq W_k \leq \lambda_p, \quad k = 1, \ldots, p.$$

To ensure $p_0\lambda_p/p \to 0$, we set $\lambda_p = p^{4/5}$.

5.2. Example 1: Correlated Strong Variables

For our first example, we return to the simulation presented earlier in Fig. 2. The data in this simulation were generated as follows. For each of the $n = 200$ observations, we randomly sampled the survival time T_i using

$$T_i = \log\left[1 + \xi_i \exp\left(\sum_{k=1}^{p} x_{i,k}\beta_k\right)\right], \quad i = 1, \ldots, n,$$

where $(\xi)_{i=1}^{n}$ were i.i.d. variables with a standard exponential distribution. Coefficients were set to zero except for the first $p_0 = 25$ variables (the strong variables); these were assigned the value $\beta_k = 1/\sqrt{p_0}$. Covariates $x_{i,k}$ for the noise variables were independently sampled from a uniform [0, 1] distribution. For the strong variables, we set $x_{i,k} = U_{i,k}U_i$, where $(U_{i,k}, U_i)$ were i.i.d. uniform [0, 1] variables. By using the same U_i, this ensured that the strong variables were pairwise correlated (pairwise correlation of $3/7 = 0.43$). The data were not censored.

We ran two sets of forests: one using the standard approach in which mtry variables were sampled randomly at each node, and the other using weighted variable sampling (see Eq. (8)). Weights were assigned within the `rsf` package by making use of the 'predictorWt' option. All forest calculations were implemented using `rsf` similar to the PBC analysis described earlier. In both cases, nodesize was set to $N = 2$ and for mtry we used $m = p^{4/5}$. For each of these forests, we extracted VIMP as well as the forest averaged minimal depth for each variable.

Figure 8 displays the results. Rows 1 and 2 display VIMP, where row 1 was obtained from the standard forest and row 2 from the forest using weighted variable sampling. VIMP values were determined using the C-index measure of prediction error under random daughter node assignment [19]. This type of VIMP calculation is similar to the Breiman permutation based VIMP commonly used [11]. Rows 3 and 4 display the results from the minimal depth analysis. Row 4 was obtained from the forest using weighted variable sampling (the same as that used for row 2). The red horizontal dashed line superimposed on the figure is the mean minimal threshold value under the null (5).

Figure 8 shows, not surprisingly, that variable selection performance degrades as p increases regardless of the method. We also find that minimal depth thresholding (row 3) is conservative, but accurate. The number of falsely identified variables (blue points below the red dashed line) is no more than one or two variables in all scenarios. The use of weighted variable sampling had no noticeable affect on VIMP (row 2), whereas for minimal depth (row 4) the effect is substantial. Now in place of only a handful of strong variables, almost all strong variables (red points) are below the minimal depth threshold, even in the $p = 1000$ simulation. The number of noisy variables that fall below this threshold has increased, but the end effect is that weighted variable sampling has allowed minimal depth to discover more true signal.

The reason for the success of weighted variable sampling with minimal depth but not VIMP goes back to our earlier discussion in Section 2 regarding prediction error. Weighted sampling improves the number of times a strong variable splits a tree, which improves its chance of splitting near the root, which ultimately improves variable selection under minimal depth thresholding. However, weighted sampling has a marginal effect on prediction error. The out-of-bag prediction error without and with weighting was 39% and 35%, respectively, for the $p = 1000$ simulation. Even with this improvement the prediction error is still too high for VIMP to be effective.

To study the effect of mtry, we repeated the minimal depth analyses for the $p = 1000$ simulation under different mtry values: $p^{1/3}$, $\sqrt{p}$, $p^{2/3}$, $p^{3/4}$, and $p^{4/5}$. All other parameters were kept the same. For each mtry value, we recorded the size of the estimated model $\hat{p}$, the false positive (FP) rate, and false negative (FN) rate based on the variables identified by minimal depth. Variables were selected using minimal depth thresholding as described in Definition 2. The prediction error (PE) for the selected model was estimated by using a hold out test data set (the same n and p were used). The simulation was repeated 100 times. Performance values were averaged over the 100 independent runs. We then repeated the same analysis, but where the data were randomly censored to have a censoring rate of 25%.

The results are displayed in Table 1. For minimal depth (abbreviated as MD), regardless of censoring, the FP rate and PE decrease with increasing mtry up until roughly $p^{3/4}$ at which point they start to increase. With increasing mtry, estimated model size increases, but the FN rate remains nearly zero. The best overall performance is when mtry equals $p^{3/4}$. For minimal depth combined with weighted variable sampling (abbreviated as MD-W), regardless of censoring, the FP rate and PE are near constant across mtry, but estimated model size increases and is accompanied by an increasing FN rate. The best overall performance is when mtry equals $p^{1/3}$ or $\sqrt{p}$.

For comparison, Table 1 also includes Cox-likelihood based boosting [9]. Computations were implemented using the `CoxBoost` R-software package [29]. The optimal number of boosting iterations was estimated using tenfold cross-validation. A boosting penalty of 100 was used. The row entry CoxBoost in Table 1 records the results from this procedure. As can be seen, CoxBoost is highly conservative with a high FP rate. Its performance is roughly on par with MD (although its PE is better under censoring), but its performance is noticeably worse than MD-W.

Table 1. High-dimensional simulation ($n = 200$, $p = 1000$, and $p_0 = 25$) where strong variables are pairwise correlated.

		No censoring				25% Censoring			
	mtry	FP	FN	$\hat{p}$	PE	FP	FN	$\hat{p}$	PE
MD	$p^{1/3}$	0.840	0.000	4	0.369	0.928	0.000	2	0.393
	$\sqrt{p}$	0.821	0.000	5	0.367	0.916	0.000	2	0.389
	$p^{2/3}$	0.714	0.001	8	0.345	0.810	0.001	5	0.365
	$p^{3/4}$	0.702	0.002	9	0.344	0.784	0.002	7	0.361
	$p^{4/5}$	0.706	0.003	10	0.345	0.787	0.004	9	0.363
MD-W	$p^{1/3}$	0.005	0.025	49	0.330	0.036	0.020	44	0.342
	$\sqrt{p}$	0.006	0.025	49	0.330	0.038	0.020	44	0.342
	$p^{2/3}$	0.005	0.035	59	0.330	0.043	0.027	51	0.342
	$p^{3/4}$	0.005	0.041	64	0.331	0.059	0.032	55	0.342
	$p^{4/5}$	0.006	0.040	64	0.331	0.075	0.028	50	0.342
CoxBoost		0.879	0.001	4	0.348	0.884	0.001	4	0.354

Results are averaged over 100 runs. Rows labeled MD, MD-W are minimal depth without and with weighting.

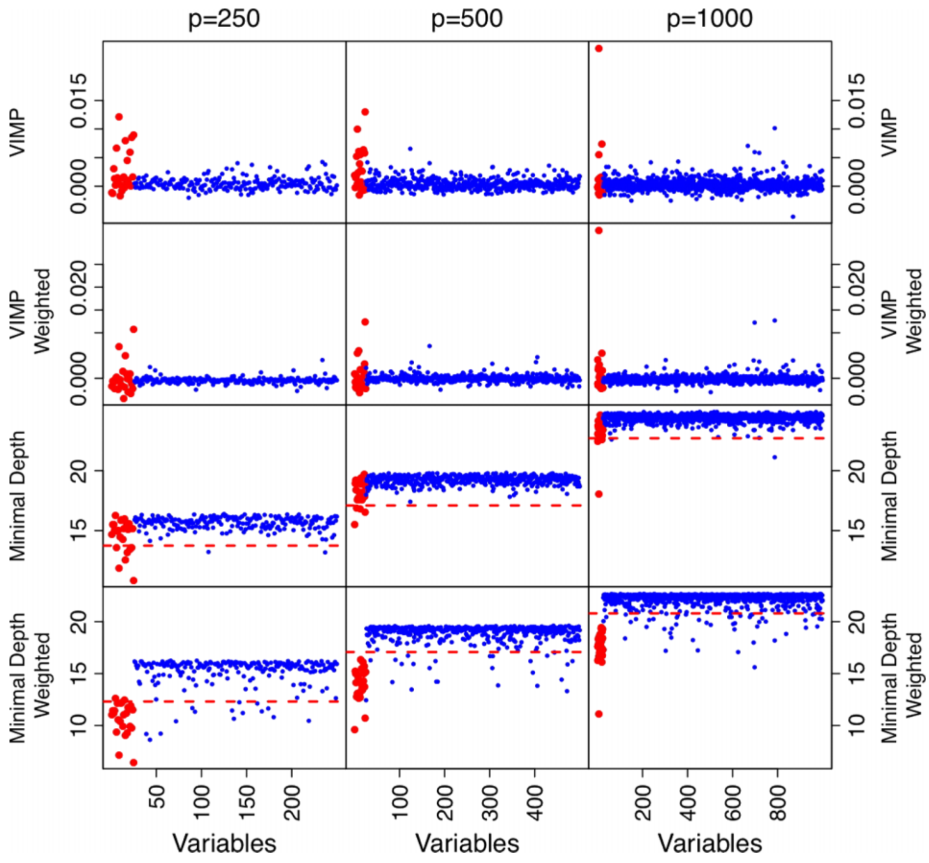

Fig. 8 High-dimensional simulation ($n = 200$, $p = 250, 500, 1000$, and $p_0 = 25$) where strong variables (red) are pairwise correlated. Rows 1 and 2 are VIMP analyses, rows 3 and 4 are minimal depth analyses. Rows 2 and 4 use weighted variable sampling where the likelihood of choosing a variable as a candidate for splitting a node was specified by Eq. (8).

5.3. Example 2: Interactions Without Main Effects

Our second simulation investigates minimal depth's ability to identify interactions. One of the strengths of VIMP is that it measures the importance of a variable in combination with all other variables. This allows VIMP to be successful in problems involving interactions. Here we study whether minimal depth has this same property.

For our simulation, all covariates were drawn independently from a uniform $[0, 1]$ distribution. We randomly selected q_0 distinct indices from $\{1, \ldots, p-1\}$ and took the interaction of each variable j in this random set with its right adjacent variable $j+1$. These $p_0 = 2q_0$ variables corresponded to the strong variables. Let $\mathscr{J}$ denotes the indices for the q_0 randomly selected values. For each i, the event time was simulated using

$$T_i = \log\left[1 + \xi_i \exp\left(\sum_{j \in \mathscr{J}} x_{i,j} x_{i,j+1} \beta_j\right)\right], \quad i = 1, \ldots, n,$$

where as before, $(\xi)_{i=1}^n$ were i.i.d. variables with a standard exponential distribution. Observe that the model includes pairwise interactions between strong variables but without any main effects. We set $\beta_j = 5/\sqrt{p_0}$ for $j \in \mathscr{J}$ where $q_0 = 5$ (so that $p_0 = 10$). The data were randomly censored to have a censoring rate of 25%.

Figure 9 displays the results. The analysis was implemented exactly as was done for Fig. 8. Delineation between strong and noisy variables is not as good as in Fig. 8, but the overall trends are similar. In particular, minimal depth combined with weighted variable sampling identifies the most signal, whereas weighted variable sampling has very little effect on VIMP.

We repeated the $p = 1000$ simulation 100 times independently under different mtry values. We then repeated the same experiment, but with a censoring rate of 50%. Table 2 displays the results. One trend that is immediately apparent is that the FP rate and PE are noticeably larger than Table 1. These set of simulations are challenging and all procedures have difficulty in selecting true signal variables, especially under heavy 50% censoring. However, although performance measures are noticeably degraded, overall trends are not that different from Table 1. In particular, weighted variable sampling continues to promote larger models which in turn has allowed for greater discovery of signal. For MD selection, PE decreases with mtry and then begin to increase at around $p^{3/4}$. FP rates decrease with mtry. For MD-W selection, FP rates decrease with mtry until $p^{2/3}$. PE increases with mtry but is generally flat. For MD, the optimal mtry appears to be $p^{3/4}$ while for MD-W it is roughly $p^{2/3}$. Finally, as before, CoxBoost selects very small models. Its FP rate and PE are better than MD but worse than MD-W.

Combining the results of Tables 1 and 2, we conclude that in sparse high-dimensional problems, minimal depth with weighted variable sampling outperforms minimal depth thresholding in terms of PE and in terms of finding more signal (lower FP rate). Weighted variable sampling appears robust to the choice of mtry, whereas minimal depth alone performs best when mtry is moderately large. These trends appear to hold independently of the amount of censoring. However, if censoring is very high, and complex interactions are at play between variables, then performance eventually degrades regardless of the method used.

6. VARIABLE HUNTING WITH MINIMAL DEPTH FOR ULTRA-HIGH DIMENSIONAL PROBLEMS

In ultra-high dimensional problems, where p may be many times larger than n, mean minimal depth thresholding becomes ineffective even with regularization to mtry, nodesize, and weighted variable sampling. For these challenging problems, further regularization is needed.

One promising method is called variable hunting. In this approach, stepwise regularization is combined with minimal depth thresholding. Briefly, the procedure works as follows. First, the data are randomly subsetted and a number of variables, *mvars*, are randomly selected. A forest is fit to these data and variables selected using minimal depth thresholding (recall Definition 2). These selected variables are used as an initial model. Variables are then added to the initial model in order by minimal depth until the joint VIMP for the nested models stabilizes. This defines the final model. This whole process is then repeated several times. Those variables appearing the most frequently up to the average estimated model size from the repetitions are selected for the final model.

Several points should be noted. First, the variable hunting algorithm described by Ishwaran *et al.* [22] did not make use of weighted variable sampling, but here we show that this can be easily and effectively incorporated into the procedure. Second, the use of joint VIMP for model selection goes back to our earlier comments about stepwise VIMP-based variable selection. We had remarked that these methods could be improved by making use of minimal depth and in fact a crucial ingredient in implementing variable hunting is that the initial model used to prime the search is based on minimal depth thresholding. Starting with variables meeting a minimal depth criterion initializes the algorithm at a good candidate model and is crucial to it success. Furthermore, models are ordered by minimal depth, as opposed to VIMP. Again, this is done to reduce the dependence on prediction error. Finally, we note that computing joint VIMP (the VIMP for a collection of variables) can be done efficiently without having to refit the forest [21].

The variable hunting algorithm can be implemented using the `varSel` R-wrapper available in the `rsf` package. A general call to `varSel` has following form:

```
vh.out <- varSel(rsf.f, data,
  method = ``vh'',
  mvars = 1000,
  predictorWt = cox.wts,
  nodesize = 2, nrep = 100,
  nstep = 5)
```

The wrapper provides several methods for selecting variables. Setting method to ‘vh’ (as above) implements the

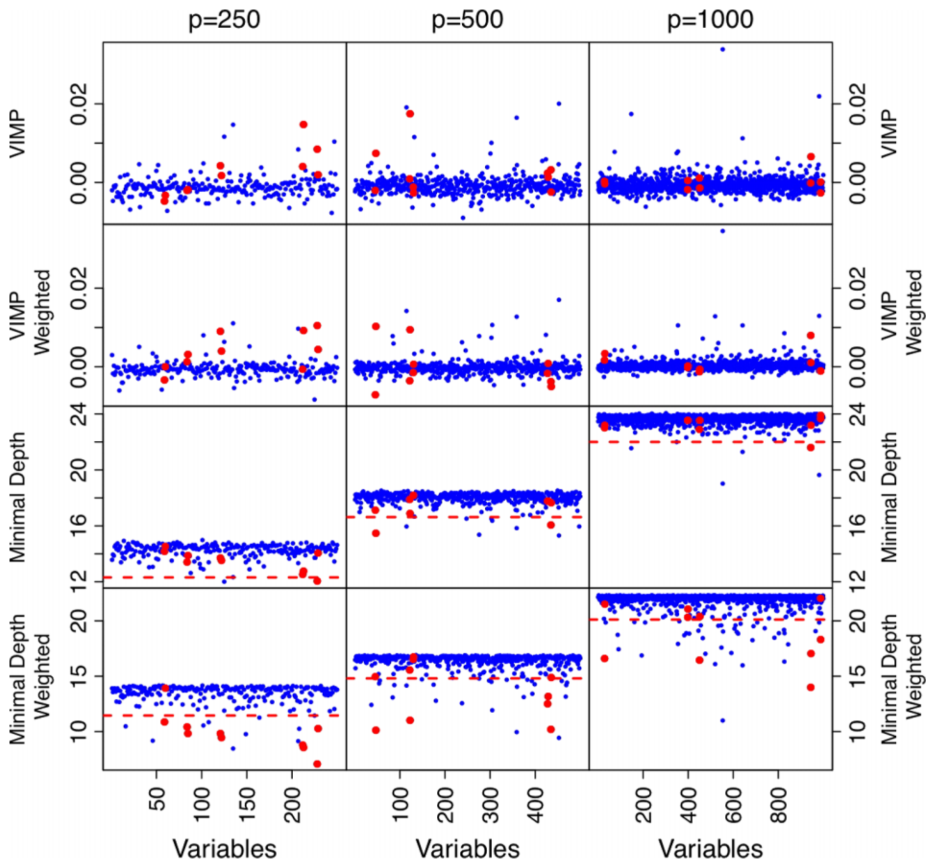

Fig. 9 High-dimensional simulation with 25% censoring ($n = 200$, $p = 250, 500, 1000$, and $p_0 = 10$). Model includes all pairwise interactions but no main effects. Rows defined as in Fig. 8.

Table 2. High-dimensional simulation ($n = 200$, $p = 1000$, and $p_0 = 10$) for model with pairwise interactions but no main effects.

		25% Censoring				50% Censoring			
	mtry	FP	FN	$\hat{p}$	PE	FP	FN	$\hat{p}$	PE
MD	$p^{1/3}$	0.949	0.001	1	0.478	0.963	0.001	1	0.487
	$\sqrt{p}$	0.949	0.001	1	0.477	0.962	0.001	1	0.488
	$p^{2/3}$	0.922	0.002	2	0.477	0.963	0.001	1	0.484
	$p^{3/4}$	0.879	0.005	6	0.468	0.946	0.003	3	0.486
	$p^{4/5}$	0.871	0.008	9	0.468	0.945	0.004	5	0.487
MD-W	$p^{1/3}$	0.392	0.027	32	0.421	0.622	0.018	22	0.450
	$\sqrt{p}$	0.393	0.027	33	0.423	0.616	0.020	23	0.451
	$p^{2/3}$	0.339	0.037	43	0.426	0.610	0.023	27	0.451
	$p^{3/4}$	0.339	0.040	47	0.427	0.644	0.023	26	0.457
	$p^{4/5}$	0.342	0.041	47	0.427	0.648	0.022	26	0.457
CoxBoost		0.719	0.002	5	0.423	0.871	0.002	3	0.461

Results are averaged over 100 runs.

variable hunting algorithm. Preweighted selection for variables is implemented using the 'predictorWt' option. This option ensures that each variable is selected with probability proportional to its preassigned weight. Another useful option is 'nstep'. This controls the step size used in the stepwise procedure. Instead of adding one variable at a time, this adds nstep variables when incrementing the initial model. Larger values of nstep encourage more model space exploration.

As illustration, we analyzed the 78 patient breast cancer microarray data set of van't Veer *et al.* [30], which we refer to as NKI-78. The data comprise $p = 24136$

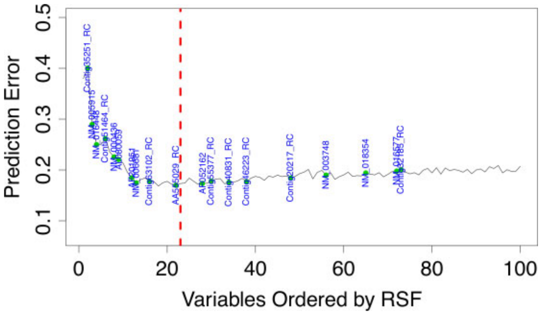

Fig. 10 Results from fitting the NKI-78 microarray breast cancer data set [30]. Shown are the error rates for nested RSF models for the top 100 variables, with variables ordered on the basis of frequency selected using variable hunting. Dashed vertical line indicates the forest estimated model size. Text in blue indicates MammaPrint genes found in the original analysis.

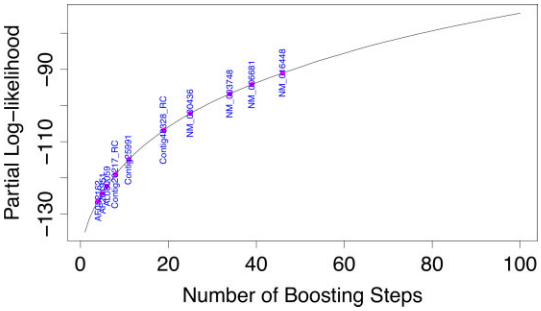

Fig. 11 Similar to Fig. 10, but with variables ordered using Cox-likelihood boosting. Vertical axis records partial log-likelihood values.

expression values. Event time was defined as the time in years for metastasis to occur. Of the 78 patients, 34 experienced metastasis, the remaining 44 were right-censored. We included only those genes whose expression values varied substantially and filtered genes similar to Bair and Tibshirani [3]. This left $p = 4707$ genes. The variable hunting algorithm was applied to these data with parameter settings of nodesize = 2, nrep = 100, and nstep = 5. In setting predictorWt, we used weights in Eq. (8) as before. All other `rsf` parameters were set to default values.

Variables were ordered by the frequency of their occurrences. These were then fit in a nested fashion with the first model consisting of the top variable hunting variable, the second model consisting of the top two variable hunting variables, and so on, up to the top 100 variables. Figure 10 displays the out-of-bag error rate for each of these models. The blue text superimposed on the figure highlights the genes found in the original van't Veer *et al*. [30] analysis (which we refer to as the MammaPrint genes). Of the top 100 RSF genes, 19 overlap with the MammaPrint set. The estimated model size determined by the variable hunting algorithm was 23 (see the dashed vertical line in Fig. 10). Of these, ten overlap with the MammaPrint signature. As comparison, Fig. 11 displays results from Cox-likelihood based boosting. Computations were implemented as in Section 5. As can be seen, CoxBoost identifies ten of the original MammaPrint genes in the first 100 boosting steps. These results are consistent with what we found in our simulations of Section 5 in that CoxBoost tends to select small parsimonious models.

For validation, we used the 295 patient breast cancer microarray data set of van de Vijver *et al*. [31], which we refer to as NKI-295. Of the 295 patients in this cohort, 61 overlapped with the original NKI-78 data. We removed these patients and used the resulting data ($n = 234$ and 63 events) as a test data set. Five different RSF models were fit to the NKI-78 training data. One used the top 23 genes from the variable hunting algorithm. Another used only those 70 genes from MammaPrint. A third model used those ten genes that were common to both models 1 and 2. The final two models used those genes common to only the first and second models, respectively (13 and 60 genes, respectively). All models were fit using a nodesize of 2. All other parameters were set to default values.

The results given in Table 3 show that all methods have lower training error rates than test set error rates. This is not unexpected, since all models used genes selected using the NKI-78 training data. The RSF model for the top 23 genes had the lowest training error rate. Over the test data, the model with genes common only to MammaPrint was best, followed closely by the RSF model based on the genes common only to RSF. The MammaPrint and RSF models had nearly similar test set error. Thus, RSF maintains prediction error on par with a well established gene signature, yet it does so with far fewer genes.

7. OTHER WAYS OF EXPLORING VARIABLE RELATIONSHIPS

So far we have focused on minimal depth as a means for thresholding variables by using mean minimal depth, but there are other ways of using minimal depth as well as maximal subtrees for exploring relationships between variables which we now briefly illustrate. One useful technique is to consider how minimal depth varies by an individual. We define $D_{v,i}$ to be the minimal depth for individual i for variable v. This value is estimated by averaging D_v over the forest for i. Variable selection is often thought of as a population-based approach but one can also consider how a variable impacts survival for a given individual. The value $D_{v,i}$ captures this concept.

To illustrate, we reanalyzed data from a multi-institutional study of esophageal cancer patients [32]. Variables measured on each patient included TNM classifications,

Table 3. Training and test set error using the NKI-78 and NKI-295 microarray breast cancer data sets.

	RSF	MammaPrint	Shared	RSF only	MammaPrint only
NKI78	0.159	0.234	0.211	0.175	0.254
NKI295	0.388	0.372	0.421	0.371	0.361
$\hat{p}$	23	70	10	13	60

Overlapping patients have been removed from NKI-295.

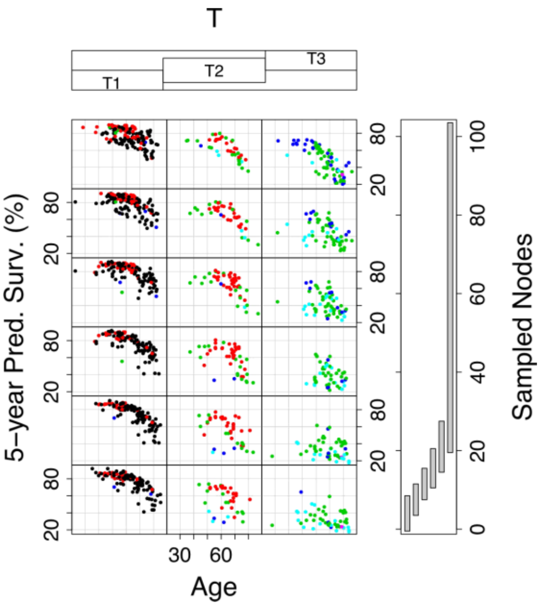

Fig. 12 Five-year predicted survival versus age, T classification, and number of sampled nodes for N0M0 adenocarcinoma esophageal patients. Colored points indicate minimal depth for each patient for histologic grade. Minimal depth increases in size going from black to red, to green, to blue, to cyan.

number of lymph nodes removed at surgery, number of cancer-positive lymph nodes, other non-TNM cancer characteristics, patient demographics, and additional variables to adjust for institution. The primary outcome used in the analysis was time to death, measured from date of surgery. For our analysis we focused on metastasis-free adenocarcinoma patients. This gave a total of $n = 3110$ observations and $p = 31$ variables. The data were analyzed using `rsf` under all default settings.

In Fig. 12, we have plotted the forest 5-year predicted survival against age, T classification, and number of sampled nodes for patients free of nodal disease and metastasis (N0M0 patients). The colored points indicate the minimal depth $D_{v,i}$ for each individual i for histologic tumor grade, a crude indicator of tumor biology. Depths $D_{v,i}$ increase in size going from black to red, to green, to blue, to cyan. For individuals with black or red points, tumor grade is especially influential on survival. We see that these patients are primarily those with T1 classification and in some cases T2 classification. This is consistent with the findings of Ishwaran *et al.* [32], and is also consistent with the biology of esophageal cancer. The unique lymphatic anatomy of the esophagus allows spread of cancer to regional lymph nodes for minimally invasive tumors such as T1 and T2 tumors, and tumor grade is an early warning sign of this. It is also interesting that tumor grade plays less of a role in younger T3 patients with more extensive lymphadenectomies (top right plot with blue points). For these deeply invasive tumors, a more extensive lymphadenectomy (more sampled lymph nodes) decreases the likelihood of being understaged so that tumor grade becomes less relevant.

As we have mentioned, maximal subtrees are a powerful tool for exploring relationships between variables. We have so far considered first-order maximal subtrees, but second-order maximal subtrees are another type of maximal subtree useful for identifying associations. A second-order maximal (w, v)-subtree is a maximal w-subtree within a maximal v-subtree for a variable v. A variable w having a maximal (w, v)-subtree close to the root of v will be highly associated with v because w splits closely after v. By considering the minimal depth of w for its maximal (w, v)-subtrees we can quantify the association of w with v. This is done for each variable w and each v so that a matrix D of dimension $p \times p$ can be constructed with each entry (i, j) measuring the pairwise association between variables v_i and v_j. These values are estimated by using forest averaged values.

Figure 13 illustrates this idea. We calculated D using the forest from our esophageal analysis and converted D to a distance matrix which we then clustered using hierarchical clustering. Interestingly, the blue cluster in the middle of the dendrogram corresponds to the top six variables in terms of their first-order minimal depth (given in the bottom row of the figure). Importantly note that D does not measure first-order minimal depth. Therefore, it is interesting that these variables are grouped together in this manner.

8. DISCUSSION

Minimal depth is a simple and powerful method for selecting variables in high-dimensional survival settings. It is easily implemented using the open source `rsf` package

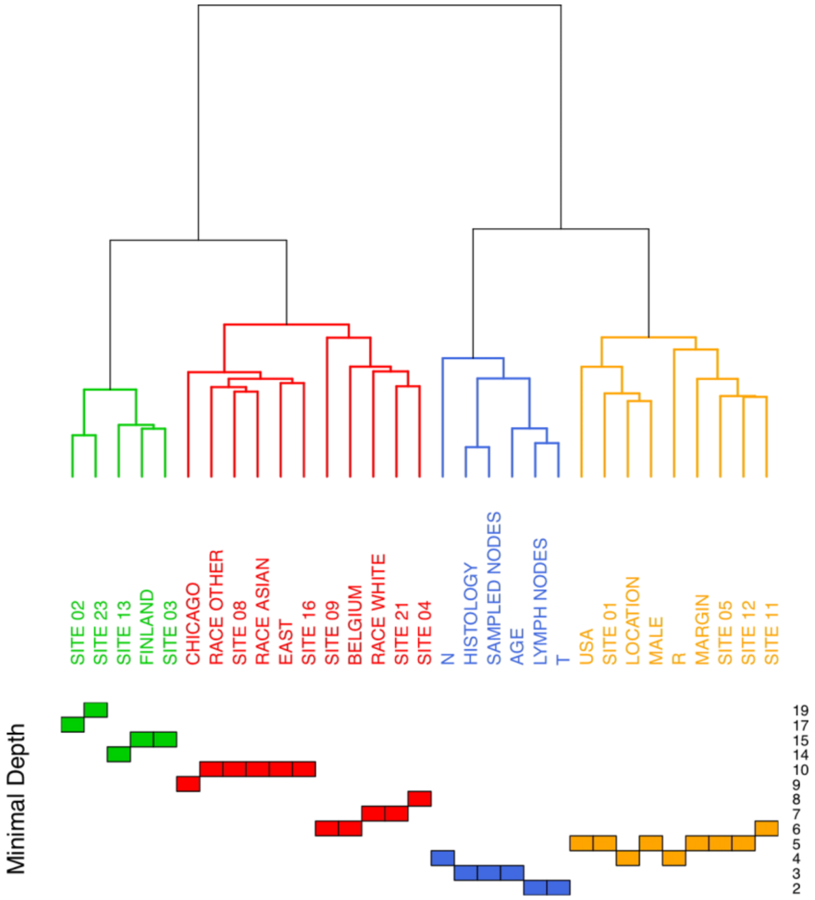

Fig. 13 Variables clustered by second-order minimal depth using second-order maximal subtrees. Bottom panel displays first-order minimal depth.

(http://cran.r-project.org). If p is on the order of n, variable selection can be based directly on mean minimal depth thresholding. In this case, mtry or nodesize must be regularized, but this is easy to do. We recommend setting nodesize to a small value, such as 1 or 2, and setting mtry to as large a value as computationally feasible while maintaining the condition $m/p = o(1)$. In our examples, we found that using larger mtry values such as $p^{2/3}$ and $p^{3/4}$ yielded far better results than using $\sqrt{p}$, the default value used by `rsf`.

We also recommend using random splitting rules. Not only does this substantially improve computational speeds, but it may also help to decorrelate trees and to improve variable selection. In our examples, randomly selected split points were chosen, and log-rank splitting was applied to these random split points. The tree node was split on that variable and random split point maximizing the log-rank statistic. The number of split points is controlled using the nsplit option in `rsf`. With continuous high-throughput data, setting nsplit to a value such as 10 should be adequate.

As well as regularizing mtry and nodesize, we discussed weighted variable sampling to encourage selection of strong variables. The weight $1 \leq W_k \leq \lambda_p$ for selecting a variable v_k should satisfy $p_0\lambda_p/p = o(1)$. In our examples, we used univariate p-values from Cox-regression to define W_k. We found the method to be quite effective. It encouraged larger models and discovered more strong variables. The effect of mtry has a minimal impact on performance of minimal depth thresholding when using weighted variable sampling. The default mtry choice of $\sqrt{p}$ appears to work best.

With very large p relative to n, different regularization is needed. One such method is the variable hunting algorithm. In this approach, minimal depth thresholding is applied to randomly selected subsets of the data and to randomly selected subsets of variables. Weighted variable sampling can also be used to select candidate variables.

Regularization for mtry and nodesize becomes less crucial with variable hunting and their values can be selected less extremely to ensure faster computations without compromising accuracy. Coupled with the fact that the algorithm uses only a subset of the data in each iteration, overall computational times can be reasonably fast. Usually about 100 iterations of the algorithm are enough to discover useful variables. The R-wrapper `varSel` implements this procedure. It can also be used as a front-end to the `max.subtree` core function for minimal depth thresholding. Note that in terms of computational burden, most of the computations are spent in growing the forest. Minimal depth calculations are relatively fast. The number of operations required to parse a tree to extract maximal subtree information is on order of the number of nodes in the tree, which is bounded by the sample size n. Thus maximal subtree parsing calculations are independent of p.

While much of the manuscript focused on minimal depth as a means for thresholding variables, in Section 7, we described alternate ways of using minimal depth as well as maximal subtrees for exploring relationships between variables. In particular, Figs. 12 and 13 indicate the potential of patient-specific minimal depth and second-order maximal subtrees for understanding long term survival of esophageal cancer patients. Although we focused on a large data set with a moderate number of variables for illustration, we believe that these methods can be extended to high-dimensional scenarios. In future work, we plan to study this more closely.

ACKNOWLEDGMENTS

We are indebted to the Associate Editor and referee for their careful and diligent refereeing. This work was partially funded by the National Heart, Lung, and Blood Institute (CAN 8324207), NIH/NCI (3P30CA016520-34S5) and by Department of Defense Era of Hope Scholar Award (W81XWH-09-1-0339).

APPENDIX: PROOF OF THEOREM 1

First consider the case when the relative depth is $d = 1$. To bound $\pi_t(d)$, we consider the complementary event. Let π_t^* be the conditional probability that t has a terminal node daughter, given that t is split using a noisy ordered variable. Suppose that v is the noisy variable used to split t and let $v_{(1)} \le \cdots \le v_{(n_t)}$ be the ordered values of v in t. If the split for v is v^* and $v_{(1)} \le v^* \le v_{(2N-1)}$ or $v_{(n_t-2N+1)} \le v^* \le v_{(n_t)}$ then one of t's daughters, say t', will have less than $2N$ observations. But if t' has less than $2N$ observations then it must be a terminal node, since any split on t' will yield a daughter with less than N unique deaths. Because each split of v is equally likely, it follows that

$$\pi_t^* \ge \frac{4N-2}{n_t}.$$

Because this is the complementary event, we have for $d = 1$ that

$$\pi_t(d) \le 1 - \frac{4N-2}{n_t}.$$

Now consider the case when $d = 2$. Let t', t'' be nonterminal daughter nodes of t such that $n_{t'}, n_{t''} \ge 2N$ (if this were not the case, they would be terminal nodes). Let $\pi_{t'}^*$ be the conditional probability that one of t''s daughters is a terminal node, given that t' is split using a noisy ordered variable. Using our previous argument we have

$$\pi_{t'}^* \ge \frac{4N-2}{n_{t'}} \ge \frac{4N-2}{n_t - 2N}.$$

As the same inequality applies to t'', it follows that

$$(1-\pi_{t'}^*)(1-\pi_{t''}^*) \le \left(1 - \frac{4N-2}{n_t-2N}\right)^2.$$

Therefore, by the product rule of probability, we have for $d = 2$ that

$$\pi_t(d) \le \left(1 - \frac{4N-2}{n_t}\right)\left(1 - \frac{4N-2}{n_t-2N}\right)^2.$$

By repeating this argument we obtain the general result (6) for arbitrary $d \ge 1$. Finally, note that the size of each daughter node at depth d must be no smaller than $2N$ as otherwise they would be classified as being terminal nodes. As there are 2^d such daughters, $n_t \ge 2^d(2N)$.

REFERENCES

[1] D. Nguyen and D. M. Rocke, Partial least squares proportional hazard regression for application to DNA microarray data, Bioinformatics 18 (2002), 1625–1632.

[2] H. Z. Li and J. Gui, Partial Cox regression analysis for high-dimensional microarray gene expression data, Bioinformatics 20 (2004), 208–215.

[3] E. Bair and R. Tibshirani, Semi-supervised methods to predict patient survival from gene expression data, PLoS Biol 2 (2004), 0511–0522.

[4] M.-Y. Park and T. Hastie, L_1-regularization path algorithm for generalized linear models, JRSSB 69 (2007), 659–677.

[5] H. H. Zhang and W. Lu, Adaptive Lasso for Cox's proportional hazards model, Biometrika 94 (2007), 691–703.

[6] H. Li and Y. Luan, Boosting proportional hazards models using smoothing splines, with applications to high-dimensional microarray data, Bioinformatics 21 (2006), 2403–2409.

[7] S. Ma and J. Huang, Clustering threshold gradient descent regularization: with applications to microarray studies, Bioinformatics 23 (2006), 466–472.

[8] T. Hothorn and P. Buhlmann, Model-based boosting in high-dimensions, Bioinformatics 22 (2006), 2828–2829.

[9] B. Binder and M. Schumacher, Allowing for mandatory covariates in boosting estimation of sparse high-dimensional survival models, BMC Bioinform 9 (2008), 14.

[10] G. Ridgeway, The state of boosting, Comput Sci Stat 31 (1999), 172–181.

[11] L. Breiman, Random forests, Mach Learn 45 (2001), 5–32.

[12] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, Classification and Regression Trees, California, Belmont, 1984.

[13] L. Breiman, Bagging predictors, Mach Learn 26 (1996), 123–140.

[14] L. Breiman, Heuristics of instability and stabilization in model selection, Ann Stat 24 (1996), 2350–2383.

[15] A. Liaw and M. Wiener. randomForest 4.5-36. R package, 2010, http://cran.r-project.org.

[16] K. L. Lunetta, L. B. Hayward, J. Segal, and P. V. Eerdewegh, Screening large-scale association study data: exploiting interactions using random forests, BMC Genet 5 (2004), 32.

[17] A. Bureau, J. Dupuis, K. Falls, K. L. Lunetta, B. Hayward, T. P. Keith, and P. V. Eerdewegh, Identifying SNPs predictive of phenotype using random forests, Genet Epidemiol 28 (2005), 171–182.

[18] R. Diaz-Uriarte and S. Alvarez de Andres, Gene selection and classification of microarray data using random forest, BMC Bioinform 7 (2006), 3.

[19] H. Ishwaran, U. B. Kogalur, E. H. Blackstone, and M. S. Lauer, Random survival forests, Ann Appl Stat 2 (2008), 841–860.

[20] H. Ishwaran and U. B. Kogalur, Random survival forests for R, Rnews 7/2 (2007), 25–31.

[21] H. Ishwaran and U. B. Kogalur, RandomSurvivalForest: Random Survival Forests. R package version 3.6.3, 2010, http://cran.r-project.org.

[22] H. Ishwaran, U. B. Kogalur, E. Z. Gorodeski, A. J. Minn, and M. S. Lauer, High-dimensional variable selection for survival data, J Am Stat Assoc 105 (2010), 205–217.

[23] H. Ishwaran, Variable importance in binary regression trees and forests, Electron J Stat 1 (2007), 519–537.

[24] H. Ishwaran, U. B. Kogalur, R. D. Moore, S. J. Gange, and B. M. Lau, Random survival forests for competing risks (submitted), 2010.

[25] R. Genuer, J.-M. Poggi, and C. Tuleau, Random Forests: some methodological insights, *ArXiv e-prints*, 0811.3619, 2008.

[26] M. R. Segal, Regression trees for censored data, Biometrics 44 (1988), 35–47.

[27] P. Geurts, D. Ernst, and L. Wehenkel, Extremely randomized trees, Mach Learn 63 (2006), 3–42.

[28] D. Amaratunga, J. Cabrera, and Y.-S. Lee, Enriched random forests, Bioinformatics 24(18) (2008), 2010–2014.

[29] H. Binder. Cox models by likelihood based boosting for a single survival endpoint or competing risks. R package version 1.2–1, 2010, http://cran.r-project.org.

[30] L. J. van't Veer, H. Dai, M. J. van de Vijver, D. Yudong, A. A. M. Hart, M. Mao, H. L. Peterse, K. van der Kooy, M. J. Marton, A. T. Witteveen, G. J. Schreiber, R. M. Kerkhoven, C. Roberts, P. S. Linsley, R. Bernards, and S. H. Friend, Gene expression profiling predicts clinical outcome of breast cancer, Nature 415 (2002), 530–536.

[31] M. J. van de Vijver, Y. D. He, L. J. van't Veer, H. Dai, A. A. M. Hart, D. W. Voskuil, G. J. Schreiber, J. L. Peterse, C. Roberts, M. J. Marton, M. Parrish, D. Atsma, A. Witteveen, A. Glas, L. Delahaye, T. van der Velde, H. Bartelink, S. Rodenhuis, E. T. Rutgers, S. H. Friend, and R. Bernards, A gene-expression signature as a predictor of survival in breast cancer, N Engl J Med 347 (2002), 1999–2009.

[32] H. Ishwaran, E. H. Blackstone, C. A. Hansen, and T. W. Rice, A novel approach to cancer staging: application to esophageal cancer, Biostatistics 10 (2009), 603–620.