Random Survival Forests for R

Hemant Ishwaran and Udaya B. Kogalur

Introduction

In this article we introduce *Random Survival Forests*, an ensemble tree method for the analysis of right censored survival data. As is well known, constructing ensembles from base learners, such as trees, can significantly improve learning performance. Recently, Breiman showed that ensemble learning can be further improved by injecting randomization into the base learning process, a method called Random Forests (Breiman, 2001). Random Survival Forests is closely modeled after Breiman's approach. In Random Forests, randomization is introduced in two forms. First, a randomly drawn bootstrap sample of the data is used for growing the tree. Second, the tree learner is grown by splitting nodes on randomly selected predictors. While at first glance Random Forest might seem an unusual procedure, considerable empirical evidence has shown it to be highly effective. Extensive experimentation, for example, has shown it compares favorably to state of the art ensembles methods such as bagging (Breiman, 1996) and boosting (Schapire et al., 1998).

Random Survival Forests being closely patterned after Random Forests naturally inherits many of its good properties. Two features especially worth emphasizing are: (1) It is user-friendly in that only three, fairly robust, parameters need to be set (the number of randomly selected predictors, the number of trees grown in the forest, and the splitting rule to be used). (2) It is highly data adaptive and virtually model assumption free. This last property is especially helpful in survival analysis. Standard analyses often rely on restrictive assumptions such as proportional hazards. Also, with such methods there is always the concern whether associations between predictors and hazards have been modeled appropriately, and whether or not non-linear effects or higher order interactions for predictors should be included. In contrast, such problems are handled seamlessly and automatically within a Random Forests approach.

While R currently has a Random Forests package for classification and regression problems (the `randomForest()` package ported by Andy Liaw and Matthew Wiener), there is currently no version available for analyzing survival data[1]. The need for a Random Forests procedure separate from one that handles classification and regression problems is well motivated as survival data possesses unique features not handled within a CART (Classification and Regression Tree) paradigm. In particular, the notion of what constitutes a good node split for growing a tree, what prediction means, and how to measure prediction performance, pose unique problems in survival analysis.

Moreover, while a survival tree can in some instances be reformulated as a classification tree, thereby making it possible to use CART software for a Random Forests analysis, we believe such approaches are merely stop-gap measures that will be difficult for the average user to implement. For example, Ishwaran et al. (2004) show under a proportional hazards assumption that one can grow survival trees using the splitting rule of LeBlanc and Crowley (1992) using the `rpart()` algorithm (Therneau and Atkinson, 1997), hence making it possible to implement a relative risk forests analysis in R. However, this requires extensive coding on the users part, is limited to proportional hazard settings, and the splitting rule used is only approximate.

The algorithm

It is clear that a comprehensive method with accompanying software is needed. To fill this need we introduce `randomSurvivalForest`, an R software package for implementing Random Survival Forests. The algorithm used by `randomSurvivalForest` is broadly described as follows:

1. Draw `ntree` bootstrap samples from the original data.
2. Grow a tree for each bootstrapped data set. At each node of the tree randomly select `mtry` predictors (covariates) for splitting on. Split on a predictor using a survival splitting criterion. A node is split on that predictor which maximizes survival differences across daughter nodes.
3. Grow the tree to full size under the constraint that a terminal node should have no less than `nodesize` unique deaths.
4. Calculate an ensemble cumulative hazard estimate by combining information from the `ntree` trees. One estimate for each individual in the data is calculated.
5. Compute an out-of-bag (OOB) error rate for the ensemble derived using the first b trees, where $b = 1, \ldots, \texttt{ntree}$.

Splitting rules

Node splits are a crucial ingredient to the algorithm. The `randomSurvivalForest` package pro-

[1] We are careful to distinguish Random Forests procedures following Breiman's methodology from other approaches. Readers, for example, should be aware of the R party() package, which implements a random forests style analysis using conditional tree base learners (Hothorn et al., 2006).

vides four different survival splitting rules for the user. These are: (i) a log-rank splitting rule, the default splitting rule, invoked by the option `splitrule="logrank"`; (ii) a conservation of events splitting rule, `splitrule="conserve"`; (iii) a logrank score rule, `splitrule="logrankscore"`; (iv) and a fast approximation to the logrank splitting rule, `splitrule="logrankapprox"`.

Notation

Assume we are at node h of a tree during its growth and that we seek to split h into two daughter nodes. We introduce some notation to help discuss how the the various splitting rules work to determine the best split. Assume that within h are n individuals. Denote their survival times and 0-1 censoring information by $(T_1, \delta_1), \ldots, (T_n, \delta_n)$. An individual l will be said to be right censored at time T_l if $\delta_l = 0$, otherwise the individual is said to have died at T_l if $\delta_l = 1$. In the case of death, T_l will be refered to as an event time, and the death as an event. An individual l who is right censored at T_l simply means the individual is known to have been alive at T_l, but the exact time of death is unknown.

A proposed split at node h on a given predictor x is always of the form $x \leq c$ and $x > c$. Such a split forms two daughter nodes (a left and right daughter) and two new sets of survival data. A good split maximizes survival differences across the two sets of data. Let $t_1 < t_2 < \cdots < t_N$ be the distinct death times in the parent node h, and let $d_{i,j}$ and $Y_{i,j}$ equal the number of deaths and individuals at risk at time t_i in the daughter nodes $j = 1, 2$. Note that $Y_{i,j}$ is the number of individuals in daughter j who are alive at time t_i, or who have an event (death) at time t_i. More precisely,

$$Y_{i,1} = \#\{T_l \geq t_i,\ x_l \leq c\}, \quad Y_{i,2} = \#\{T_l \geq t_i,\ x_l > c\},$$

where x_l is the value of x for individual $l = 1, \ldots, n$. Finally, define $Y_i = Y_{i,1} + Y_{i,2}$ and $d_i = d_{i,1} + d_{i,2}$. Let n_j be the total number of observations in daughter j. Thus, $n = n_1 + n_2$. Note that $n_1 = \#\{l : x_l \leq c\}$ and $n_2 = \#\{l : x_l > c\}$.

Log-rank splitting

The log-rank test for a split at the value c for predictor x is

$$L(x,c) = \frac{\sum_{i=1}^{N} \left(d_{i,1} - Y_{i,1} \frac{d_i}{Y_i} \right)}{\sqrt{\sum_{i=1}^{N} \frac{Y_{i,1}}{Y_i} \left(1 - \frac{Y_{i,1}}{Y_i} \right) \left(\frac{Y_i - d_i}{Y_i - 1} \right) d_i}}.$$

The value $|L(x,c)|$ is the measure of node separation. The larger the value for $|L(x,c)|$, the greater the difference between the two groups, and the better the split is. In particular, the best split at node h is determined by finding the predictor x^* and split value c^* such that $|L(x^*, c^*)| \geq |L(x,c)|$ for all x and c.

Conservation of events splitting

The log-rank test for splitting survival trees is a well established concept (Segal, 1988), having been shown to be robust in both proportional and non-proportional hazard settings (LeBlanc and Crowley, 1993). However, one criticism often heard is that it tends to favor continuous predictors and often suffers from an end-cut preference (favoring uneven splits). However, in our experience with Random Survival Forests we have not found this to be a serious deficiency. Nevertheless, to address this potential problem we introduce another important class of test statistics for splitting that are related to conservation of events; a concept introduced in Naftel et al. (1985) (our simulations have indicated these tests may be much less susceptible to the aforementioned problems).

Under fairly general conditions, conservation of events asserts that the sum of the estimated cumulative hazard function over the observed time points (deaths and censored values) must equal the total number of deaths. This applies to a wide collection of estimates including the the Nelson-Aalen estimator. The Nelson-Aalen cumulative hazard estimator for daughter j is

$$\hat{H}_j(t) = \sum_{t_{i,j} \leq t} \frac{d_{i,j}}{Y_{i,j}}$$

where $t_{i,j}$ are the ordered death times for daughter j (note: we define $0/0 = 0$).

Let $(T_{l,j}, \delta_{l,j})$, for $l = 1, \ldots, n_j$, denote all survival times and censoring indicator pairs for daughter j. Conservation of events asserts that

$$\sum_{l=1}^{n_j} \hat{H}_j(T_{l,j}) = \sum_{l=1}^{n_j} \delta_{l,j}. \tag{1}$$

In other words, the total number of deaths is conserved in each daughter.

The conservation of events splitting rule is motivated by (1). First, order the time points within each daughter node such that

$$T_{(1),j} \leq T_{(2),j} \leq \cdots \leq T_{(n_j),j}.$$

Let $\delta_{(l),j}$ be the censoring indicator function for the ordered value $T_{(l),j}$. Define

$$\mathscr{M}_{k,j} = \sum_{l=1}^{k} \hat{H}_j(T_{(l),j}) - \sum_{l=1}^{k} \delta_{(l),j}, \quad k = 1, \ldots, n_j.$$

One can think of $\mathscr{M}_{k,j}$ as "residuals" that measure accuracy of conservation of events. The proposed test statistic takes the sum of the absolute values of $\mathscr{M}_{k,j}$ for $k = 1, \ldots, n_j$ for each daughter j, and weights these values by the number of individuals at risk

within each group. Observe that $\mathscr{M}_{n_j,j} = 0$, but nothing can be said about $\mathscr{M}_{k,j}$ for $k < n_j$. Thus, by considering $\mathscr{M}_{k,j}$ for each k, the proposed test measures how evenly distributed conservation of events is over all deaths. The measure of conservation of events for the split on x at the value c is

$$\text{Conserve}(x,c) = \frac{1}{Y_{1,1}+Y_{1,2}} \sum_{j=1}^{2} Y_{1,j} \sum_{k=1}^{n_j-1} |\mathscr{M}_{k,j}|.$$

This value is small if the two groups are well separated. Because we want to maximize survival differences due to a split, we use the transformed value $1/(1+\text{Conserve}(x,c))$ as our measure of node separation.

The preceding expression for $\text{Conserve}(x,c)$ can be quite expensive to compute as it involves summing over all survival times within the daughter nodes. However, we can greatly reduce the amount of work by compressing the sums to involve only event times. With some work, one can show that $\text{Conserve}(x,c)$ is equivalent to:

$$\frac{1}{Y_{1,1}+Y_{1,2}} \sum_{j=1}^{2} Y_{1,j} \sum_{k=1}^{N-1} \left\{ N_{k,j} Y_{k+1,j} \sum_{l=1}^{k} \frac{d_{l,j}}{Y_{l,j}} \right\},$$

where $N_{i,j} = Y_{i,j} - Y_{i+1,j}$ equals the number of observations in daughter j with observed time falling within the interval $[t_i, t_{i+1})$ for $i = 1, \ldots, N$ where $t_{N+1} = \infty$.

Log-rank score splitting

Another useful splitting rule available within the `randomSurvivalForest` package is the log-rank score test of Hothorn and Lausen (2003). To describe this rule, assume the predictor x has been ordered so that $x_1 \le x_2 \le \cdots \le x_n$. Now, compute the "ranks" for each survival time T_l,

$$a_l = \delta_l - \sum_{k=1}^{\Gamma_l} \frac{\delta_k}{n - \Gamma_k + 1}$$

where $\Gamma_k = \#\{t : T_t \le T_k\}$. The log-rank score test is defined as

$$S(x,c) = \frac{\sum_{x_l \le c} a_l - n_1 \bar{a}}{\sqrt{n_1 \left(1 - \frac{n_1}{n}\right) s_a^2}}$$

where $\bar{a}$ and s_a^2 are the sample mean and sample variance of $\{a_l : l = 1, \ldots, n\}$. Log-rank score splitting defines the measure of node separation by $|S(x,c)|$. Maximizing this value over x and c yields the best split.

Approximate logrank splitting

An approximate log-rank test can be used in place of $L(x,c)$ to greatly reduce computations. To derive the approximation, first rewrite the numerator of $L(x,c)$ in a form that uses the Nelson-Aalen estimator for the parent node. The Nelson-Aalen estimator is

$$\hat{H}(t) = \sum_{t_i \le t} \frac{d_i}{Y_i}.$$

As shown in LeBlanc and Crowley (1993) one can write

$$\sum_{i=1}^{N} \left(d_{i,1} - Y_{i,1} \frac{d_i}{Y_i} \right) = D_1 - \sum_{l=1}^{n} \text{I}\{x_l \le c\} \hat{H}(T_l),$$

where $D_j = \sum_{i=1}^{N} d_{i,j}$ for $j = 1, 2$. Because the Nelson-Aalen estimator is computed on the parent node, and not daughter nodes, this yields an efficient way to compute the numerator of $L(x,c)$.

Now to simplify the denominator, we approximate the variance of the numerator of $L(x,c)$ as in Section 7.7 of Cox and Oakes (1988) (this approximation was suggested to us by Michael LeBlanc in personal communication). Setting $D = \sum_{i=1}^{N} d_i$, we get the following approximation to the log-rank test $L(x,c)$:

$$\frac{D^{1/2} \left(D_1 - \sum_{l=1}^{n} \text{I}\{x_l \le c\} \hat{H}(T_l) \right)}{\sqrt{\left\{ \sum_{l=1}^{n} \text{I}\{x_l \le c\} \hat{H}(T_l) \right\} \left\{ D - \sum_{l=1}^{n} \text{I}\{x_l \le c\} \hat{H}(T_l) \right\}}}.$$

Ensemble estimation

The `randomSurvivalForest` package produces an ensemble estimate for the cumulative hazard function. This is our predictor and key deliverable. Error rate performance is calculated based on this value. The ensemble is derived as follows. First, for each tree grown from a bootstrap data set we estimate the cumulative hazard function for the tree. This is accomplished by grouping hazard estimates by terminal nodes. Consider a specific node h. Let $\{t_{l,h}\}$ be the distinct death times in h and let $d_{l,h}$ and $Y_{l,h}$ equal the number of deaths and individuals at risk at time $t_{l,h}$. The cumulative hazard estimate for node h is defined as

$$\hat{H}_h(t) = \sum_{t_{l,h} \le t} \frac{d_{l,h}}{Y_{l,h}}.$$

Each tree provides a sequence of such estimates, $\hat{H}_h(t)$. If there are M terminal nodes in the tree, then there are M such estimates. To compute $\hat{H}(t|\mathbf{x}_i)$ for an individual i with predictor $\mathbf{x}_i$, simply drop $\mathbf{x}_i$ down the tree. The terminal node for i yields the desired estimator. More precisely,

$$\hat{H}(t|\mathbf{x}_i) = \hat{H}_h(t), \text{ if } \mathbf{x}_i \in h. \qquad (2)$$

Note this value is computed *for all individuals* i in the data.

The estimate (2) is based on one tree. To produce our ensemble we average (2) over all `ntree` trees. Let $\hat{H}_b(t|\mathbf{x})$ denote the cumulative hazard estimate (2) for tree $b = 1,\ldots,$`ntree`. Define $I_{i,b} = 1$ if i is an OOB point for b, otherwise set $I_{i,b} = 0$. The OOB ensemble cumulative hazard estimator for i is

$$\hat{H}^*_e(t|\mathbf{x}_i) = \frac{\sum_{b=1}^{\texttt{ntree}} I_{i,b}\hat{H}_b(t|\mathbf{x}_i)}{\sum_{b=1}^{\texttt{ntree}} I_{i,b}}.$$

Observe that the estimator is obtained by averaging over only those bootstrap samples in which i is excluded (i.e., those datasets in which i is an OOB value). The OOB estimator is in contrast to the ensemble cumulative hazard estimator that uses all samples:

$$\hat{H}_e(t|\mathbf{x}_i) = \frac{1}{\texttt{ntree}} \sum_{b=1}^{\texttt{ntree}} \hat{H}_b(t|\mathbf{x}_i).$$

Concordance error rate

Given the OOB estimator $\hat{H}^*_e(t|\mathbf{x})$, it is a simple matter to compute the error rate. We measure error using Harrell's concordance index (Harrell et al., 1982). Unlike other measures of survival performance, Harrell's C-index does not depend on choosing a fixed time for evaluation of the model and specifically takes into account censoring of individuals (May et al., 2004). The method has quickly become quite popular in the literature as a means for assessing prediction performance in survival analysis settings. See Kattan et al. (1998) and references therein.

To compute the concordance index we must define what constitutes a worse predicted outcome. We take the following approach. Let $t^*_1,\ldots,t^*_N$ denote all unique event times in the data. Individual i is said to have a worse outcome than j if

$$\sum_{k=1}^{N} \hat{H}^*_e(t^*_k|\mathbf{x}_i) > \sum_{k=1}^{N} \hat{H}^*_e(t^*_k|\mathbf{x}_j).$$

The concordance error rate is computed as follows:

1. Form all possible pairs of observations over all the data.
2. Omit those pairs where the shorter event time is censored. Also, omit pairs i and j if $T_i = T_j$ unless $\delta_i = 1$ and $\delta_j = 0$ or $\delta_i = 0$ and $\delta_j = 1$. The last restriction only allows ties if one of the observations is a death and the other a censored observation. Let Permissible denote the total number of permissible pairs.
3. Count 1 for each permissible pair in which the shorter event time had the worse predicted outcome. Count 0.5 if the predicted outcomes are tied. Let Concordance denote the total sum over all permissible pairs.
4. Define the concordance index C as
$$C = \frac{\text{Concordance}}{\text{Permissible}}.$$
5. The error rate is Error $= 1 - C$. Note that $0 \leq$ Error ≤ 1 and that Error $= 0.5$ corresponds to a procedure doing no better than random guessing, whereas Error $= 0$ indicates perfect accuracy.

Usage in R

The user interface to `randomSurvivalforest` is similar in many aspects to `randomForest` and as the reader may have already noticed, many of the argument names are also the same. This was done deliberately in order to promote compatibility between the two packages. The primary R function call to the `randomSurvivalforest` package is `rsf()`. The online documentation describes `rsf()` in great detail and there is no reason to repeat this information here. Different R wrapper functions are provided with the `randomSurvivalforest` package to aid in interpreting the object produced by `rsf()`. The examples given below illustrate how some of these wrappers work, and also indicate how `rsf()` might be used in practice.

Lung-vet data

For our first example, we use the well known veteran's administration lung cancer data from Kalbfleisch and Prentice (Kalbfleisch and Prentice, 1980). This is an example data set available within the package. In total there are 6 predictors in the data. We first focus on analysis that includes only Karnofsky score as a predictor:

```
> library("randomSurvivalForest")
> data(veteran,package="randomSurvivalForest")
> ntree <- 1000
> v.out <- rsf(Survrsf(time,status) ~ karno,
        veteran, ntree=ntree, forest=T)
> print(v.out)

Call:
 rsf.default(formula = Survrsf(time, status)
      ~ karno, data = veteran, ntree = ntree)

                         Sample size: 137
                    Number of deaths: 128
                     Number of trees: 1000
          Minimum terminal node size: 3
       Average no. of terminal nodes: 8.437
No. of variables tried at each split: 1
              Total no. of variables: 1
                       Splitting rule: logrank
              Estimate of error rate: 36.28%
```

The error rate is significantly smaller than 0.5, the benchmark value associated with a procedure no better than flipping a coin. This is very strong evidence that Karnofsky score is predictive.

We can investigate the effect of Karnofsky score more closely by considering how the ensemble estimated mortality varies as a function of the predictor:

```
> plot.variable(v.out, partial=T)
```

Figure 1, produced by the above command, is a partial plot of Karnofsky score. The vertical axis represents expected number of deaths.

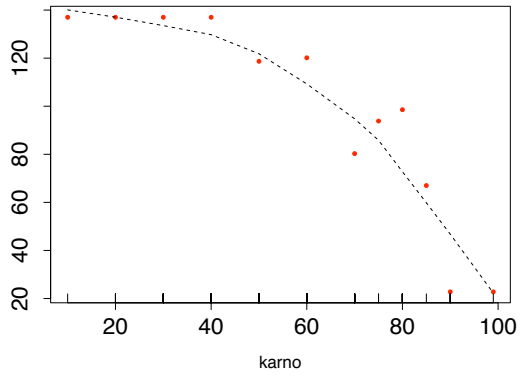

Figure 1: Partial plot of Karnofsky score. Vertical axis is mortality $\sum_{k=1}^{N} N_k \hat{H}_e(t_k^*|x)$ for a given Karnofsky value x and represents expected number of deaths.

Now we run an analysis with all six predictors under each of the four splitting rules. For each splitting rule we run 100 replications and record the mean and standard deviation of the concordance error rate (as before `ntree` equals 1000):

```
> splitrule <- c("logrank", "conserve",
          "logrankscore", "logrankapprox")
> nrep <- 100
> err.rate <- matrix(0, 4, nrep)
> names(err.rate) <- splitrule
> v.f <-
      as.formula("Survrsf(time,status) ~ .")
> for (j in 1:4) {
>  for (k in 1:nrep) {
>   err.rate[j,k] <- rsf(v.f,
      veteran,ntree=ntree,
      splitrule=splitrule[j])$err.rate[ntree]
>  }
> }
> err.rate <- rbind(
               mean=apply(err.rate, 1, mean),
               std=apply(err.rate, 1, sd))
> colnames(err.rate) <- splitrule
> print(round(err.rate,4))
```

```
     logrank conserve logrankscore logrankapx
mean  0.2982   0.3239       0.2951     0.3170
std   0.0027   0.0034       0.0027     0.0046
```

The analysis shows that `logrankscore` has the best predictive performance (`logrank` is a close second). Standard deviations in all cases are reasonably small. It is interesting to observe that the mean error rates are not substantially smaller than our previous analysis which used only Karnofsky score, thus indicating the predictor is highly influential. Our next example illustrates further techniques for studying the informativeness of a predictor.

Primary biliary cirrhosis (PBC) of the liver

Next we consider the PBC data set found in appendix D.1 of Fleming and Harrington (Fleming and Harrington, 1991). This is also an example data set available in the package. Similar to the previous analysis we analyzed the data by running a forest analysis for each of the four splitting rules, repeating the analysis 100 times independently (as before `ntree` was set to 1000). The R code is similar as before and suppressed:

```
     logrank conserve logrankscore logrankapx
mean  0.1703   0.1677       0.1719     0.1602
std   0.0014   0.0014       0.0015     0.0020
```

As can be seen, the error rates are between 16-17% with `logrankapprox` having the lowest value.

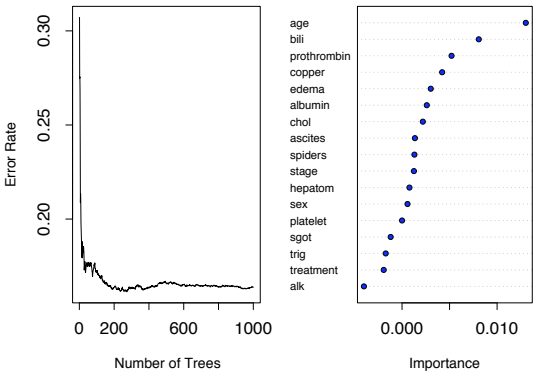

Figure 2: Error rate for PBC data as a function of trees (left-side) and out-of-bag importance values for predictors (right-side).

We now consider the informativeness of each predictor under the `logrankapprox` splitting rule:

```
> data("pbc",package="randomSurvivalForest")
> pbc.f <- as.formula("Survrsf(days,status)~.")
> pbc.out <- rsf(pbc.f, pbc, ntree=ntree,
    splitrule = "logrankapprox", forest=T)
> plot(pbc.out)
```

Figure 2 depicts the importance values for all 17 predictors. From the plot we see that "age" and "bili" are clearly predictive and have substantially larger importance values than all other predictors. The partial plots for the top six predictors are displayed in

Figure 3. The figure was produced using the command:

```
> plot.variable(pbc.out,3,partial=T,n.pred=6)
```

We now consider the incremental effect of each predictor using a nested analysis. We sort predictors by their importance values and consider the nested sequence of models starting with the top variable, followed by the model with the top 2 variables, then the model with the top three variables, and so on:

```
> imp <- pbc.out$importance
> pnames <- pbc.out$predictorNames
> pnames.order <- pnames[rev(order(imp))]
> n.pred <- length(pnames)
> pbc.err <- rep(0, n.pred)
> for (k in 1:n.pred){
>  rsf.f <- "Survrsf(days,status)~"
>  rsf.f <- as.formula(paste(rsf.f,
>    paste(pnames.order[1:k],collapse="+")))
>  pbc.err[k] <- rsf(rsf.f, pbc, ntree=ntree,
>   splitrule="logrankapprox")$err.rate[ntree]
> }
> pbc.imp.out <- as.data.frame(
>   cbind(round(rev(sort(imp)),4),
>         round(pbc.err,4),
>         round(-diff(c(0.5,pbc.err)),4)),
>         row.names=pnames.order)
>colnames(pbc.imp.out) <-
>               c("Imp","Err","Drop Err")
> print(pbc.imp.out)
```

	Imp	Err	Drop Err
age	0.0130	0.3961	0.1039
bili	0.0081	0.1996	0.1965
prothrombin	0.0052	0.1918	0.0078
copper	0.0042	0.1685	0.0233
edema	0.0030	0.1647	0.0038
albumin	0.0026	0.1569	0.0078
chol	0.0022	0.1606	-0.0037
ascites	0.0014	0.1570	0.0036
spiders	0.0013	0.1601	-0.0030
stage	0.0013	0.1557	0.0043
hepatom	0.0008	0.1570	-0.0013
sex	0.0006	0.1549	0.0021
platelet	0.0000	0.1565	-0.0016
sgot	-0.0012	0.1538	0.0027
trig	-0.0017	0.1545	-0.0007
treatment	-0.0019	0.1596	-0.0052
alk	-0.0040	0.1565	0.0032

The first column is the importance value of a predictor in the full model. The kth value in the second column is the error rate for the kth nested model, while the kth value in the third column is the difference between the error rate for the kth and $(k-1)$th nested model, where the error rate for the null model, $k = 0$, is 0.5. One can see not much is gained by using more than 6-7 predictors and that the top 3-4 predictors account for much of the predictive power.

Large scale problems

In terms of computationally challenging problems, we have applied `randomSurvivalForest` successfully to several large survival datasets. For example, we have considered data collected at the Cleveland Clinic involving over 20,000 records and well over 60 predictors. We have also analyzed a data set containing 1,000 records and with almost 250 predictors. Our success with these applications is consistent with that seen for Random Forests: namely, that the methodology has been shown to scale up very nicely, even in very large predictor spaces and with large sample sizes. In terms of computational speed, we have found that `logrankapprox` is almost always fastest. After that, `conserve` is second fastest. For very large datasets, discretizing continuous predictors and/or the observed survival times can greatly speed up computational times. Discretization does not have to be overly granular for substantial gains to be seen.

Acknowledgements

The authors are extremely grateful to Eugene H. Blackstone and Michael S. Lauer for their tremendous support, feedback, and generous time given to us throughout this project. Without their help this project would surely never have been completed. We also thank the referee of the paper and Paul Murrell for their constructive and helpful comments. This research was supported by the National Institutes of Health RO1 grant HL-072771.

Bibliography

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

L. Breiman. Bagging predictors. *Machine Learning*, 26:123–140, 1996.

D.R. Cox and D. Oakes. *Analysis of Survival Data*. Chapman and Hall, London, 1998.

T. Fleming and D. Harrington. *Counting Processes and Survival Analysis*. Wiley, New York, 1991.

F. Harrell, R. Califf, D. Pryor, K. Lee, and R. Rosati. Evaluating the yield of medical tests. *J. Amer. Med. Assoc.*, 247:2543–2546, 1982.

T. Hothorn, K. Hornik, and A. Zeileis. Unbiased recursive partitioning: A conditional inference framework. *J. Comp. Graph. Statist.*, 15:651–674, 2006.

T. Hothorn, and B. Lausen. On the exact distribution of maximally selected rank statistics. *Comput. Statist. Data Analysis*, 43:121–137, 2003.

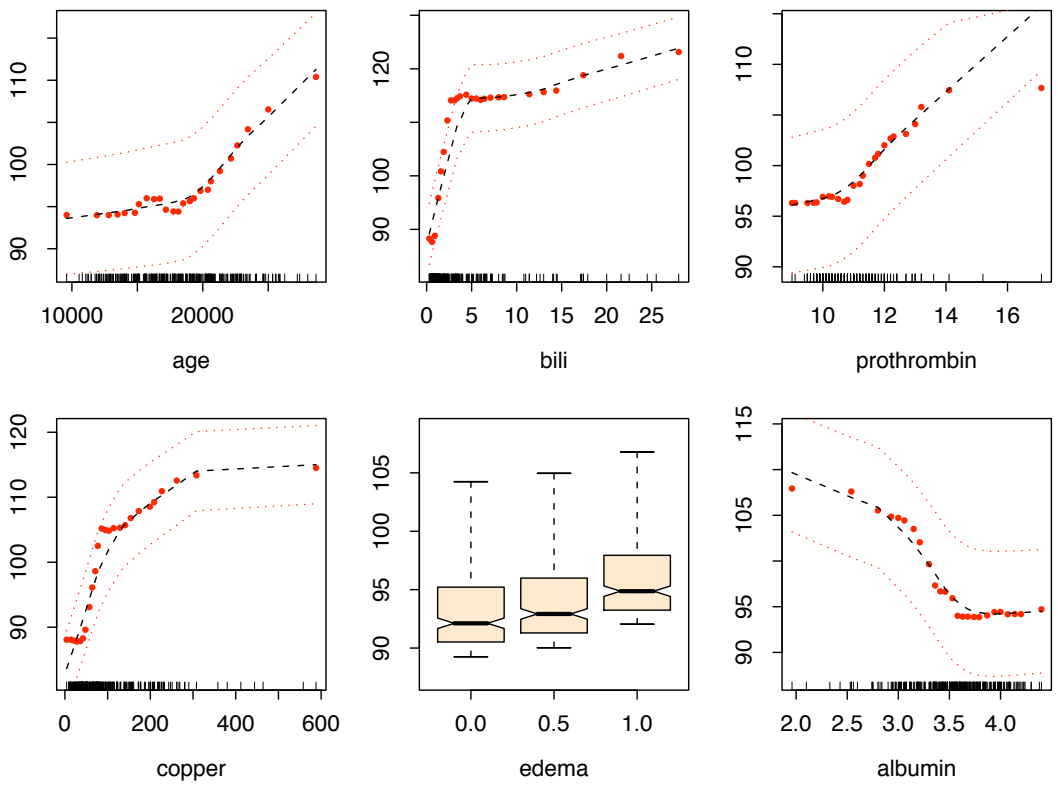

Figure 3: Partial plots for top six predictors from PBC data. Values on the vertical axis represent expected number of deaths for a given predictor, after adjusting for all other predictors. Dashed red lines for continuous predictors are ± 2 standard error bars.

H. Ishwaran, E. Blackstone, M. Lauer, and C. Pothier. Relative risk forests for exercise heart rate recovery as a predictor of mortality. *J. Amer. Stat. Assoc.*, 99: 591–600, 2004.

J. Kalbfleisch and R. Prentice. *The Statistical Analysis of Failure Time Data*. Wiley, New York, 1980.

M. Kattan, K. Hess, and J. Beck. Experiments to determine whether recursive partitioning (CART) or an artificial neural network overcomes theoretical limitations of Cox proportional hazards regression. *Computers and Biomedical Research*, 31:363–373, 1998.

M. LeBlanc and J. Crowley. Relative risk trees for censored survival data. *Biometrics*, 48:411–425, 1992.

M. LeBlanc and J. Crowley. Survival trees by goodness of split. *J. Amer. Stat. Assoc.*, 88:457–467, 1993.

M. May, P. Royston, M. Egger, A.C. Justice and J.A.C. Sterne. Development and validation of a prognostic model for survival time data: application to prognosis of HIV positive patients treated with antiretroviral therapy. *Statist. Medicine.*, 23: 2375–2398, 2004.

D. Naftel, E. Blackstone, and M. Turner. Conservation of events, 1985. Unpublished notes.

R. Schapire, Y. Freund, P. Bartlett, and W. Lee. Boosting the margin: a new explanation for the effectiveness of voting methods. *Ann. Statist.*, 26(5):1651–1686, 1998.

M.R. Segal. Regression trees for censored data. *Biometrics*, 44:35–47, 1988.

T. Therneau and E. Atkinson. An introduction to recursive partitioning using the rpart routine. Technical Report 61, 1997. Section of Biostatistics, Mayo Clinic, Rochester.

Hemant Ishwaran
Department of Quantitative Health Sciences
Cleveland Clinic, Cleveland, U.S.A.
`hemant.ishwaran@gmail.com`
Udaya B. Kogalur
Kogalur Shear Corporation
Clemmons, U.S.A.
`ubk@kogalur-shear.com`