Random Survival Forests

Hemant Ishwaran (https://ishwaran.org/ishwaran.html)

Michael S. Lauer

Eugene H. Blackstone

Min Lu

Udaya B. Kogalur

2022-03-03

Introduction

Early applications of random forests (RF) focused on regression and classification problems. Random survival forests (Ishwaran, Kogalur, Blackstone, & Lauer, 2008) (RSF) was introduced to extend RF to the setting of right-censored survival data. Implementation of RSF follows the same general principles as RF: (a) Survival trees are grown using bootstrapped data; (b) Random feature selection is used when splitting tree nodes; (c) Trees are generally grown deeply, and (d) The survival forest ensemble is calculated by averaging terminal node statistics (TNS).

The presence of censoring is a unique feature of survival data that complicates certain aspects of implementing RSF compared to RF for regression and classification. In right-censored survival data, the observed data is (T, δ) where T is time and δ is the censoring indicator. The observed time T is defined as the minimum of the true (potentially unobserved) survival event time T^o and the true (potentially unobserved) censoring time C^o; thus $T = \min(T^o, C^o)$ and the actual event time might not be observed. The censoring indicator is defined as $\delta = I\{T^o \leq C^o\}$. When $\delta = 1$, an event has occurred (i.e., death has occurred) and we observe the true event time, $T = T^o$. Otherwise when $\delta = 0$, the observation is censored and we only observe the censoring time $T = C^o$: thus we know that the subject has survived to time C^o, but not when the subject actually dies.

Hereafter we denote the data by $(T_1, \mathbf{X}_1, \delta_1), \ldots, (T_n, \mathbf{X}_n, \delta_n)$ where $\mathbf{X}_i$ is the feature vector (covariate) for individual i and T_i and δ_i are the observed time and censoring indicators for i. RSF trees just like RF trees are grown using resampling (for example by using the bootstrap; the default is to use .632 sampling without replacement). However for notational simplicity, we will sometimes ignore this distinction.

RSF splitting rules

The true event time being subject to censoring must be dealt with when growing a RSF tree. In particular, the splitting rule for growing the tree must specifically account for censoring. Thus, the goal is to split the tree node into left and right daughters with dissimilar event history (survival) behavior.

Log-rank splitting

The default splitting rule used by the package is the log-rank test statistic and is specified by `splitrule="logrank"` . The log-rank test has traditionally been used for two-sample testing with survival data, but it can be used for survival splitting as a means for maximizing between-node survival differences (Ciampi, Hogg, McKinney, & Thiffault, 1988; LeBlanc & Crowley, 1992, 1993; Mark Robert Segal, 1988; Mark R. Segal, 1995).

To explain log-rank splitting, consider a specific tree node to be split. Without loss of generality let us assume this is the root node (top of the tree). For simplicity assume the data is not bootstrapped, thus the root node data is $(T_1, \mathbf{X}_1, \delta_1), \ldots, (T_n, \mathbf{X}_n, \delta_n)$. Let X denote a specific variable (i.e., one of the coordinates of the feature vector). A proposed split using X is of the form $X \leq c$ and $X > c$ (for simplicity we assume X is nominal) and splits the node into left and right daughters, $L = \{X_i \leq c\}$ and $R = \{X_i > c\}$, respectively. Let

$$t_1 < t_2 < \cdots < t_m$$

be the distinct death times and let $d_{j,L}, d_{j,R}$ and $Y_{j,L}, Y_{j,R}$ equal the number of deaths and individuals at risk at time t_j in daughter nodes L, R. At risk means the number of individuals in a daughter who are alive at time t_j, or who have an event (death) at time t_j:

$$Y_{j,L} = \#\{T_i \geq t_j,\ X_i \leq c\}, \qquad Y_{j,R} = \#\{T_i \geq t_j,\ X_i > c\}.$$

Define

$$Y_j = Y_{j,L} + Y_{j,R}, \qquad d_j = d_{j,L} + d_{j,R}.$$

The log-rank split-statistic value for the split is

$$L(X,c) = \frac{\displaystyle\sum_{j=1}^{m} \left(d_{j,L} - Y_{j,L} \frac{d_j}{Y_j} \right)}{\sqrt{\displaystyle\sum_{j=1}^{m} \frac{Y_{j,L}}{Y_j} \left(1 - \frac{Y_{j,L}}{Y_j}\right) \left(\frac{Y_j - d_j}{Y_j - 1}\right) d_j}}.$$

The value $|L(X,c)|$ is a measure of node separation. The larger the value, the greater the survival difference between L and R, and the better the split is. The best split is determined by finding the feature X^* and split-value c^* such that $|L(X^*,c^*)| \geq |L(X,c)|$ for all X and c.

Log-rank score splitting

The package also implements splitting using the log-rank score test (Hothorn & Lausen, 2003). This is specified by the option `splitrule="logrankscore"` . To describe this rule, assume the variable X has been ordered so that $X_1 \leq X_2 \leq \cdots \leq X_n$ where for simplicity we assume there are n unique values for X (no ties). Now compute the ``ranks" for each survival time T_j,

$$a_j = \delta_j - \sum_{k=1}^{\Gamma_j} \frac{\delta_k}{n - \Gamma_k + 1}$$

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

$$S(x,c) = \frac{\sum_{X_j \leq c} a_j - n_L \bar{a}}{\sqrt{n_L \left(1 - \frac{n_L}{n}\right) s_a^2}}$$

where $\bar{a}$ and s_a^2 are the sample mean and sample variance of $\{a_j : l = 1, \ldots, n\}$ and $n = n_L + n_R$ where n_L is the sample size of the left daughter node. 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.

Randomized splitting

All models in the package including RSF allow the use of randomized splitting specified by the option `nsplit`. Rather than splitting the node by considering all possible split-values for a variable, instead a fixed number of randomly selected split-points $c_1, \ldots, c_{\texttt{nsplit}}$ are chosen (Ishwaran, 2015; Ishwaran et al., 2008; Ishwaran, Kogalur, Gorodeski, Minn, & Lauer, 2010). For example, the best randomized split using log-rank splitting is the maximal value of

$$|L(X,c_1)|, \ldots, |L(X,c_{\texttt{nsplit}})|.$$

For each variable X, this reduces n split-statistic evaluations (worst case scenario) to `nsplit` evaluations. Not only does randomized splitting greatly reduce computations, it also mitigates the well known tree bias of favoring splits on variables with a large number of split-points, such as continuous variables or factors with a large number of categorical labels (Loh & Shih, 1997). Related work includes Geurts, Ernst, & Wehenkel (2006) who investigated extremely randomized trees. Here a single random split-point is chosen for each variable (i.e., $\texttt{nsplit} = 1$). Traditional deterministic splitting (all split values considered) is specified by `nsplit = 0`.

Terminal node statistics (TNS)

RSF estimates the survival function, $S(t|\mathbf{X}) = \mathbb{P}\{T^o > t|\mathbf{X}\}$, and the cumulative hazard function (CHF),

$$H(t|\mathbf{X}) = \int_{(0,t]} \frac{F(du|\mathbf{X})}{S(u|\mathbf{X})}, \qquad F(u|\mathbf{X}) = \mathbb{P}\{T^o \leq u|\mathbf{X}\}.$$

Below we describe how these two quantities are estimated using a survival tree.

In-bag (IB) estimator

Once the survival tree is grown, the ends of the tree are called the terminal nodes. The survival tree predictor is defined in terms of the predictor within each terminal nodel. Let h be a terminal node of the tree and let

$$t_{1,h} < t_{2,h} < \cdots < t_{m(h),h}$$

be the unique death times in h and let $d_{j,h}$ and $Y_{j,h}$ equal the number of deaths and individuals at risk at time $t_{j,h}$. The CHF and survival functions for h are estimated using the bootstrapped Nelson-Aalen and Kaplan-Meier estimators

$$H_h(t) = \sum_{t_{j,h} \leq t} \frac{d_{j,h}}{Y_{j,h}}, \qquad S_h(t) = \prod_{t_{j,h} \leq t} \left(1 - \frac{d_{j,h}}{Y_{j,h}}\right).$$

The survival tree predictor is defined by assigning all cases within h the same CHF and survival estimate. This is because the purpose of the survival tree is to partition the data into homogeneous groups (i.e., terminal nodes) of individuals with similar survival behavior. To estimate $H(t|\mathbf{X})$ and $S(t|\mathbf{X})$ for a given feature $\mathbf{X}$, drop $\mathbf{X}$ down the tree. Because of the binary nature of a tree, $\mathbf{X}$ will fall into a unique terminal node h. The CHF and survival estimator for $\mathbf{X}$ equals the Nelson-Aalen and Kaplan-Meier estimator for $\mathbf{X}$'s terminal node:

$$H^{\text{IB}}(t|\mathbf{X}) = H_h(t), \quad S^{\text{IB}}(t|\mathbf{X}) = S_h(t), \text{ if } \mathbf{X} \in h.$$

Note that we use the notation IB because the above estimators are based on the training data, which is the IB data.

Out-of-bag (OOB) estimators

To define the OOB estimator, let $I_i \in \{0, 1\}$ indicate whether case i is IB or OOB. Let $I_i = 1$ if and only if i is OOB. Drop i down the tree and let h denote i's terminal node. The OOB tree estimators for i is

$$H^{\text{OOB}}(t|\mathbf{X}_i) = H_h(t), \quad S^{\text{OOB}}(t|\mathbf{X}_i) = S_h(t), \text{ if } \mathbf{X}_i \in h \text{ and } I_i = 1.$$

Observe these are NULL unless i is OOB for the tree.

Ensemble CHF and Survival Function

The ensemble CHF and survival function are determined by averaging the tree estimator. Let $H_b^{\text{IB}}(t|\mathbf{X})$ and $S_b^{\text{IB}}(t|\mathbf{X})$ be the IB CHF and survival estimator for the bth survival tree. The IB ensemble estimators are

$$\bar{H}^{\text{IB}}(t|\mathbf{X}) = \frac{1}{\texttt{ntree}} \sum_{b=1}^{\texttt{ntree}} H_b(t|\mathbf{X}), \quad \bar{S}^{\text{IB}}(t|\mathbf{X}) = \frac{1}{\texttt{ntree}} \sum_{b=1}^{\texttt{ntree}} S_b(t|\mathbf{X}).$$

Let O_i record trees where case i is OOB. The OOB ensemble estimators for individual i are

$$\bar{H}_i^{\text{OOB}}(t) = \frac{1}{|O_i|} \sum_{b \in O_i} H_b^{\text{IB}}(t|\mathbf{X}_i), \quad \bar{S}_i^{\text{OOB}}(t) = \frac{1}{|O_i|} \sum_{b \in O_i} S_b^{\text{IB}}(t|\mathbf{X}_i), \quad i = 1, \ldots, n.$$

An important distinction between the two sets of estimators is that OOB estimators are used for inference on the training data and for estimating prediction error. In-bag estimators on the other hand are used for prediction and can be used for any feature $\mathbf{X}$.

Why are two estimates provided?

Why does RSF estimate both the CHF and the survival function? Classically, we know the two are related by

$$H(t) = -\log(S(t)).$$

The problem is that even if it were true that $H_b = -\log(S_b)$ for every tree $b = 1, \ldots, \texttt{ntree}$, the above identity will not hold for the ensemble. Let $\bar{S}$ and $\bar{H}$ denote the ensemble survival and CHF and let $\mathbb{E}_b$ denote ensemble expectation (i.e., $\bar{S} = \mathbb{E}_b[S_b]$ and $\bar{H} = \mathbb{E}_b[H_b]$). Then by Jensen's inequality for convex functions,

$$-\log(\bar{S}) = -\log(\mathbb{E}_b[S_b]) \leq \mathbb{E}_b[-\log(S_b)] = \mathbb{E}_b[H_b] = \bar{H}.$$

In other words, $-\log$ of the survival ensemble does not necessarily equal the ensemble of $-\log$ of the tree survival function. The inequality above also shows that taking $-\log$ of the ensemble survival will generally be smaller than the true ensemble CHF. This is why RSF provides the two values separately.

Prediction error

Prediction error for survival models is measured by $1 - \mathrm{C}$, where C is Harrell's concordance index (Harrell Jr et al., 1982). Prediction error is between 0 and 1, and measures how well the predictor correctly ranks two random individuals in terms of survival. 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. The method is a popular means for assessing prediction performance in survival settings since it is easy to understand and interpret.

Mortality (the predicted value used by RSF)

To compute the concordance index we must define what constitutes a worse predicted outcome. For survival models this is defined by the concept of *mortality* which is the predicted value used by RSF. Let $t_1 < \cdots < t_m$ denote the entire set of unique event times for the learning data. The IB ensemble mortality for a feature $\mathbf{X}$ is defined as

$$\bar{M}^{\mathrm{IB}}(\mathbf{X}) = \sum_{j=1}^{m} \bar{H}^{\mathrm{IB}}(t_j|\mathbf{X}).$$

This estimates the number of deaths expected if all cases were similar to $\mathbf{X}$. OOB ensemble mortality which is used for the C-index calculation is defined by

$$\bar{M}_i^{\mathrm{OOB}} = \sum_{j=1}^{m} \bar{H}_i^{\mathrm{OOB}}(t_j), \quad i = 1, \ldots, n.$$

Individual i is said to have a worse outcome than individual j if

$$\bar{M}_i^{\mathrm{OOB}} > \bar{M}_j^{\mathrm{OOB}}.$$

The mortality value (Ishwaran et al., 2008) represents estimated risk for each individual calibrated to the scale of the number of events. Thus as a specific example, if i has a mortality value of 100, then if all individuals had the same covariate as i, which is $\mathbf{X} = \mathbf{x}_i$, we would expect an average of 100 events.

C-index calculation

The C-index is calculated using the following steps:

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, j) if $T_i = T_j$ unless $(\delta_i = 1, \delta_j = 0)$ or $(\delta_i = 0, \delta_j = 1)$ or $(\delta_i = 1, \delta_j = 1)$. The last restriction only allows ties in event times if at least one of the observations is a death. Let the resulting pairs be denoted by $\mathbb{S}$. Let *permissible* $= |\mathbb{S}|$.
3. If $T_i \neq T_j$, count 1 for each $s \in \mathbb{S}$ in which the shorter time had the worse predicted outcome.
4. If $T_i \neq T_j$, count 0.5 for each $s \in \mathbb{S}$ in which $\bar{M}_i^{\mathrm{OOB}} = \bar{M}_j^{\mathrm{OOB}}$.
5. If $T_i = T_j$, count 1 for each $s \in \mathbb{S}$ in which $\bar{M}_i^{\mathrm{OOB}} = \bar{M}_j^{\mathrm{OOB}}$.

6. If $T_i = T_j$, count 0.5 for each $s \in \mathbb{S}$ in which $\bar{M}_i^{\text{OOB}} \neq \bar{M}_j^{\text{OOB}}$.
7. Let concordance denote the resulting count over all permissible pairs. Define the concordance index C as

$$C = \frac{concordance}{permissible}.$$

8. The error rate is $\texttt{PE} = 1 - \texttt{C}$. Note that $0 \leq \texttt{PE} \leq 1$ and that $\texttt{PE} = 0.5$ corresponds to a procedure doing no better than random guessing, whereas $\texttt{PE} = 0$ indicates perfect prediction.

Brier score

The Brier score, $\text{BS}(t)$, is another popular measure used to assess prediction peformance. Let $\hat{S}(t|\mathbf{X})$ be some estimator of the survival function. To estimate the prediction performance of $\hat{S}$, let $\hat{G}(t|\mathbf{X})$ be a prechosen estimator of the censoring survival function, $G(t|\mathbf{X}) = \mathbb{P}\{C^o \geq t|\mathbf{X}\}$. The Brier score for $\hat{S}(t|\mathbf{X})$ can be estimated by using inverse probability of censoring weights (IPCW) using the method of Gerds & Schumacher (2006)

$$\hat{\text{BS}}(t) = \frac{1}{n}\sum_{i=1}^{n}\left\{\frac{\hat{S}^2(t|\mathbf{X}_i)I\{T_i \leq t\}\delta_i}{\hat{G}(T_i - |\mathbf{X}_i)} + \frac{\left(1 - \hat{S}(t|\mathbf{X}_i)\right)^2 I\{T_i > t\}}{\hat{G}(t|\mathbf{X}_i)}\right\}.$$

The integrated Brier score at time τ is defined as

$$\text{IBS}(\tau) = \frac{1}{\tau}\int_0^{\tau} \text{BS}(t)dt$$

which can be estimated by substituting $\hat{\text{BS}}(t)$ for $\text{BS}(t)$. Lower values for the Brier score indicate better prediction performance. Using the Brier score we can calculate the continuous rank probability score (CRPS), defined as the integrated Brier score divided by time.

Illustration

To illustrate, we use the survival data from Hsich, Gorodeski, Blackstone, Ishwaran, & Lauer (2011) consisting of 2231 adult patients with systolic heart failure. All patients underwent cardiopulmonary stress testing. During a mean follow-up of 5 years (maximum for survivors, 11 years), 742 patients died. The outcome is all-cause mortality and a total of $p = 39$ covariates were measured for each patient including demographic, cardiac and noncardiac comorbidity, and stress testing information.

```
library(randomForestSRC)
data(peakVO2, package = "randomForestSRC")
dta <- peakVO2
obj <- rfsrc(Surv(ttodead,died)~., dta,
             ntree = 1000, nodesize = 5, nsplit = 50, importance = TRUE)
print(obj)

>  1                          Sample size: 2231
>  2                     Number of deaths: 726
>  3                      Number of trees: 1000
>  4            Forest terminal node size: 5
>  5        Average no. of terminal nodes: 259.368
>  6 No. of variables tried at each split: 7
>  7               Total no. of variables: 39
>  8        Resampling used to grow trees: swor
>  9     Resample size used to grow trees: 1410
> 10                             Analysis: RSF
> 11                               Family: surv
> 12                       Splitting rule: logrank *random*
> 13        Number of random split points: 50
> 14                            (OOB) CRPS: 0.15476339
> 15    (OOB) Requested performance error: 0.29830498
```

From the above code, `obj$survival` `obj$survival.oob` and `obj$chf` `obj$chf.oob` contain the IB and OOB survival functions and CHFs respectively. All these four objects have a dimension of $n \times c$, where n is number of patients and c is number of unique time points from the inputted argument `ntime` , and all the unique death times are ordered and stored in `obj$time.interest` . The one-dimensional predicted values are stored in `obj$predicted` and `obj$predicted.oob` , which are IB and OOB mortality. The continuous rank probability score (CRPS) is shown in line 14 which is defined as the integrated Brier score divided by time. The Brier score is calculated using inverse probability of censoring weights (IPCW) using the method of Gerds et al. (2006) Gerds & Schumacher (2006). The error rate is stored in `obj$err.rate` which is calculated using the C-calculation as described above (survival.html#c-index-calculation-1) and is shown in line 15. For the printout, line 2 displays the number of deaths; line 3 and 4 displays the number of trees and the size of terminal node, which are specified by the input argument `ntree` and `nodesize` . Line 5 displays the number of terminal nodes per tree averaged across the forest and line 6 reflects the input argument `mtry` , which is the number of variables randomly selected as candidates for splitting a node; line 8 displays the type of bootstrap, where `swor` refers to sampling without replacement and `swr` refers to sampling with replacement; line 9 displays the sample size for line 8 where for `swor` , the number equals to about 63.2% observations, which matches the ratio of the original data in sampling with replacement; line 10 and 11 show the type of forest where `RSF` and `surv` refer to family survival (getstarted.html#growing-a-forest-1); line 12 displays splitting rule (getstarted.html#split-rules-1) which matches the inputted argument `splitrule` and line 13 shows the number of random splits to consider for each candidate splitting variable which matches the inputted argument `nsplit` .

The following illustrates how to get the C-index directly from a RSF analysis using the built in function get.cindex.

```
get.cindex(obj$yvar[,1], obj$yvar[,2], obj$predicted.oob)
>  [1] 0.299378
```

Plot the estimated survival functions

Figure 1 displays the (in-bag) predicted survival functions for two hypothetical individuals, where all p features of the two individuals are set to the median level except for the variable peak VO_2. These two individuals were defined in the following code in `newdata`, whose predictions were stored in `y.pred` which was plotted in the next block of code. For one of the individuals this is set at the 25th quantile for peak VO_2 (peak $VO_2 = 12.8$ mL/kg per min) and shown using a solid black line. For the other individual this was set to the 75th quantile (peak $VO_2 = 19.3$ mL/kg per min) and shown using a dashed red line.

```
newdata <- data.frame(lapply(1:ncol(obj$xvar),function(i){median(obj$xvar[,i])}))
colnames(newdata) <- obj$xvar.names
newdata1 <- newdata2 <- newdata
newdata1[,which(obj$xvar.names == "peak_vo2")] <- quantile(obj$xvar$peak_vo2, 0.25)
newdata2[,which(obj$xvar.names == "peak_vo2")] <- quantile(obj$xvar$peak_vo2, 0.75)
newdata <- rbind(newdata1,newdata2)
y.pred <- predict(obj,newdata = rbind(newdata,obj$xvar)[1:2,])
```

```
pdf("survival.pdf", width = 10, height = 8)
par(cex.axis = 2.0, cex.lab = 2.0, cex.main = 2.0, mar = c(6.0,6,1,1), mgp = c(4, 1, 0)
plot(round(y.pred$time.interest,2),y.pred$survival[1,], type="l", xlab="Time (Year)",
     ylab="Survival", col=1, lty=1, lwd=2)
lines(round(y1$time.interest,2), y1$survival[2,], col=2, lty=2, lwd=2)
legend("topright", legend=c("Peak VO2=12.8","Peak VO2=19.3"), col=c(1:2), lty=1:2, cex=
dev.off()
```

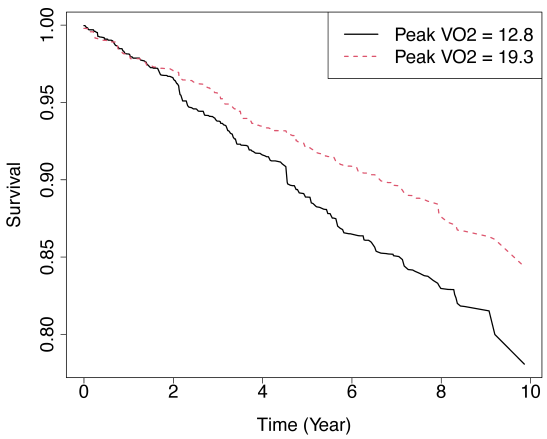


Figure 1

Predicted survival functions for two hypothetical individuals from RSF analysis of systolic heart failure data. Solid black line represents individual with peak $VO_2 = 12.8$ mL/kg per min. Red dash line represents individual with peak $VO_2 = 19.3$ mL/kg per min. All other variables for both individuals are set to the median value.

Plot the Brier Score and Calculate the CRPS

Figure 2 displays the Brier $\hat{\mathrm{BS}}(t)$ for the RSF analysis of the heart failure data given above. These values are obtained by using the function `get.brier.survival`. In this function, two methods are available for estimating the censoring distribution used for the IPCW. The first method uses the Kaplan-Meier censoring distribution estimator and the second using a RSF censoring distribution estimator. The results are given below:

```
## obtain Brier score using KM and RSF censoring distribution estimators
bs.km <- get.brier.survival(obj, cens.mode = "km")$brier.score
bs.rsf <- get.brier.survival(obj, cens.mode = "rfsrc")$brier.score

## plot the brier score
plot(bs.km, type = "s", col = 2)
lines(bs.rsf, type ="s", col = 4)
legend("bottomright", legend = c("cens.model = km", "cens.model = rfsrc"), fill = c(2,4
```

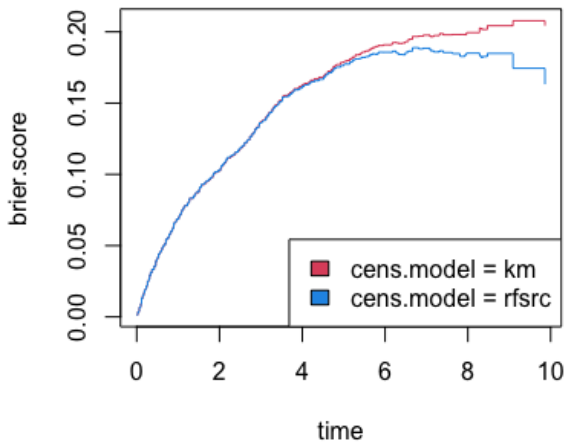
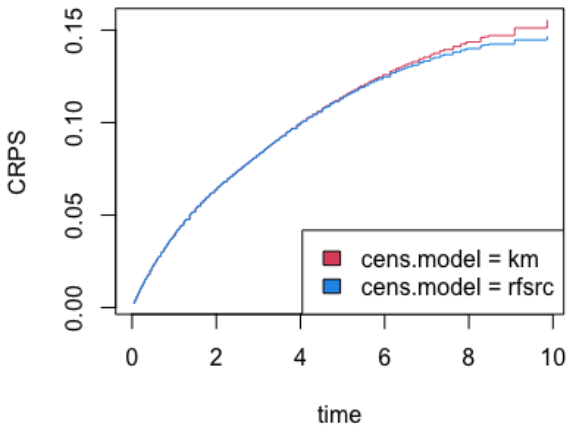


Figure 2

The Brier score can be used to obtain other information, such as the CRPS. The `get.brier.survival` function actually returns this value, but below we show how to extract CRPS for ever time point directly from the previously estimated Brier score.

```
## here's how to calculate the CRPS for every time point
trapz <- randomForestSRC:::trapz
time <- obj$time.interest
crps.km <- sapply(1:length(time), function(j) {
  trapz(time[1:j], bs.km[1:j, 2] / diff(range(time[1:j])))
})
crps.rsf <- sapply(1:length(time), function(j) {
  trapz(time[1:j], bs.rsf[1:j, 2] / diff(range(time[1:j])))
})

## plot CRPS as function of time
plot(time, crps.km, ylab = "CRPS", type = "s", col = 2)
lines(time, crps.rsf, type ="s", col = 4)
legend("bottomright", legend=c("cens.model = km", "cens.model = rfsrc"), fill=c(2,4))
```



Variable Importance

RSF provides a fully nonparametric measure of variable importance (VIMP). The most common measure is Breiman-Cutler VIMP (Breiman, 2002) and is called permutation importance. VIMP calculated using permutation importance adopts a prediction based approach by measuring prediction error attributable to the variable. A clever feature is that rather than using cross-validation, which can be computationally expensive, permutation importance makes use of OOB estimation. Specifically, to calculate the VIMP for a variable X, we randomly permute the OOB values of X in a tree (the remaining coordinates of $\mathbf{X}$ are not altered). The perturbed OOB data is dropped down the tree and the OOB error for the resulting tree predictor determined. The amount by which this new error exceeds the original OOB error for the tree equals the tree importance for X. Averaging over trees yields permutation importance for X.

Large positive VIMP indicates high predictive ability while zero or negative values identify noise variables. Subsampling (Ishwaran & Lu, 2019) can be used to estimate the standard error and to approximate the confidence intervals for VIMP. Figure 3 displays delete-d jackknife 99% asymptotic normal confidence intervals for the $p = 39$ variables from the systolic heart failure RSF analysis. Prediction error was calculated using the C-index.

```
jk.obj <- subsample(obj)
pdf("VIMPsur.pdf", width = 15, height = 20)
par(oma = c(0.5, 10, 0.5, 0.5))
par(cex.axis = 2.0, cex.lab = 2.0, cex.main = 2.0, mar = c(6.0,17,1,1), mgp = c(4, 1,
plot(jk.obj, xlab = "Variable Importance (x 100)", cex = 1.2)
dev.off()
```

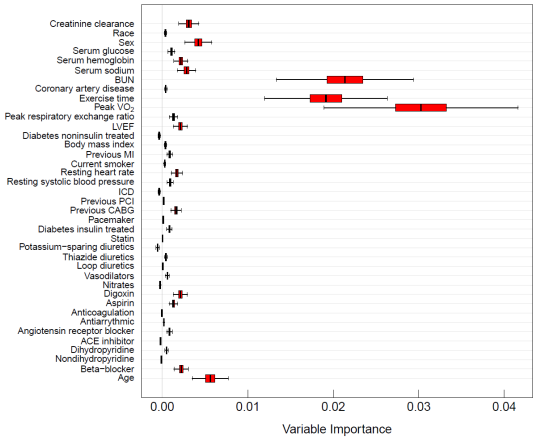


Figure 3

Delete-d jackknife 99% asymptotic normal confidence intervals of VIMP from RSF analysis of systolic heart failure data. Prediction error is defined using Harrell's concordance index.

Cite this vignette as

H. Ishwaran, M. S. Lauer, E. H. Blackstone, M. Lu, and U. B. Kogalur. 2021. "randomForestSRC: random survival forests vignette." https://luminwin.github.io/randomForestSRC/articles/survival.html (https://luminwin.github.io/randomForestSRC/articles/survival.html).

```
@misc{HemantRandomS,
    author = "Hemant Ishwaran and Michael S. Lauer and Eugene H. Blackstone and Min Lu
              and Udaya B. Kogalur",
    title = {{randomForestSRC}: random survival forests vignette},
    year = {2021},
    url = {https://luminwin.github.io/randomForestSRC/articles/survival.html}
}
```

References

Breiman, L. (2002). Manual on setting up, using, and understanding random forests v3. 1. *Statistics Department University of California Berkeley, CA, USA*, *1*.

Ciampi, A., Hogg, S. A., McKinney, S., & Thiffault, J. (1988). RECPAM: a computer program for recursive partition and amalgamation for censored survival data. *Comp. Methods Programs Biomed.*, *26*(3), 239–256. Elsevier.

Gerds, T. A., & Schumacher, M. (2006). Consistent estimation of the expected brier score in general survival models with right-censored event times. *Biometrical Journal*, *48*(6), 1029–1040. Wiley Online Library.

Geurts, P., Ernst, D., & Wehenkel, L. (2006). Extremely randomized trees. *Machine Learning*, *63*(1), 3–42. Springer.

Harrell Jr, F. E., Califf, R. M., Pryor, D. B., Lee, K. L., Rosati, R. A., et al. (1982). Evaluating the yield of medical tests. *JAMA*, *247*(18), 2543–2546.

Hothorn, T., & Lausen, B. (2003). On the exact distribution of maximally selected rank statistics. *Computational Statistics & Data Analysis*, *43*(2), 121–137. Elsevier.

Hsich, E., Gorodeski, E. Z., Blackstone, E. H., Ishwaran, H., & Lauer, M. S. (2011). Identifying important risk factors for survival in patient with systolic heart failure using random survival forests. *Circulation: Cardiovascular Quality and Outcomes*, *4*(1), 39–45. Lippincott Williams; Wilkins.

Ishwaran, H. (2015). The effect of splitting on random forests. *Machine Learning*, *99*(1), 75–118. Springer.

Ishwaran, H., Kogalur, U. B., Blackstone, E. H., & Lauer, M. S. (2008). Random survival forests. *The Annals of Applied Statistics*, *2*(3), 841–860. JSTOR.

Ishwaran, H., Kogalur, U. B., Gorodeski, E. Z., Minn, A. J., & Lauer, M. S. (2010). High-dimensional variable selection for survival data. *Journal of the American Statistical Association*, *105*(489), 205–217. Taylor & Francis.

Ishwaran, H., & Lu, M. (2019). Standard errors and confidence intervals for variable importance in random forest regression, classification, and survival. *Statistics in medicine*, *38*(4), 558–582. Wiley Online Library. Retrieved from http://web.ccs.miami.edu/~hishwaran/papers/IL.StatMed.2019.pdf (http://web.ccs.miami.edu/~hishwaran/papers/IL.StatMed.2019.pdf)

LeBlanc, M., & Crowley, J. (1992). Relative risk trees for censored survival data. *Biometrics*, 411–425. JSTOR.

LeBlanc, M., & Crowley, J. (1993). Survival trees by goodness of split. *Journal of the American Statistical Association*, *88*(422), 457–467. Taylor & Francis Group.

Loh, W.-Y., & Shih, Y.-S. (1997). Split selection methods for classification trees. *Statistica Sinica*, 815–840. JSTOR.

Segal, Mark Robert. (1988). Regression trees for censored data. *Biometrics*, 35–47. JSTOR.

Segal, Mark R. (1995). Extending the elements of tree-structured regression. *Statistical Methods in Medical Research*, *4*(3), 219–236. Sage Publications Sage CA: Thousand Oaks, CA.
