Journal of Statistical Software

February 2018, Volume 83, Issue 12. *doi: 10.18637/jss.v083.i12*

Constructing Multivariate Survival Trees: The MST Package for R

Peter Calhoun
San Diego State

Xiaogang Su
University of Texas

Martha Nunn
Creighton University

Juanjuan Fan
San Diego State

Abstract

Multivariate survival trees require few statistical assumptions, are easy to interpret, and provide meaningful diagnosis and prediction rules. Trees can handle a large number of predictors with mixed types and do not require predictor variable transformation or selection. These are useful features in many application fields and are often required in the current era of big data. The aim of this article is to introduce the R package **MST**. This package constructs multivariate survival trees using marginal model and frailty model based approaches. It allows the user to control and see how the trees are constructed. The package can also simulate high-dimensional, multivariate survival data from marginal and frailty models.

Keywords: multivariate survival trees, CART, marginal model, frailty model, R.

1. Introduction

Decision trees have gained popularity in many application fields because they can handle a variety of data structures, require few statistical assumptions, and yield classification and prediction rules that are easy to interpret. Generalizations of classification and regression trees (CARTs) applied to survival analysis can provide meaningful prognosis rules in medical research. Many authors have proposed tree-based methods to handle univariate (or uncorrelated) survival data (see, e.g., Gordon and Olshen 1985; Davis and Anderson 1989; Therneau, Grambsch, and Fleming 1990; LeBlanc and Crowley 1992). We have extended the research to handle multivariate survival data (Su and Fan 2004; Fan, Su, Levine, Nunn, and LeBlanc 2006; Fan, Nunn, and Su 2009). The goal of this paper is to discuss constructing multivariate survival trees in R (R Core Team 2017) using the **MST** package (Su, Calhoun, and Fan 2018) available at the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/package=MST`.

The paper is organized as follows. In Section 2 we introduce multivariate survival trees and their construction. This will help users decide which model is most appropriate for their dataset. We discuss the **MST** package including its features in Section 3 and analyze a simulated and a real multivariate survival dataset in Section 4. We conclude with a discussion on possible future developments.

2. Background

Multivariate failure time arises when individuals or objects under study are naturally clustered or when individuals experience multiple events of the same type (namely, recurrent times). Examples include disease incidence times of family members, component failure times in medical devices, and tooth loss times in dental studies. When not all subjects experience the event of interest and failure times are not all independent, multivariate survival methods must be used to account for the censoring and dependence. The details to analyze correlated failure times and construct multivariate survival trees are discussed in our papers Su and Fan (2004) and Fan *et al.* (2006, 2009); key aspects are summarized herein.

The construction of multivariate survival trees adopts a modified CART procedure to model the correlated failure times. The procedure consists of three steps: (1) growing a large initial tree, (2) pruning it back to obtain a sequence of subtrees, and (3) selecting the best tree size. For the tree growing and pruning purposes, one needs a splitting statistic that handles the dependence of failure times and a split-complexity measure to evaluate the performance of the tree. There are two main approaches to analyzing correlated failure times and developing the splitting statistic. One is the marginal approach where the correlation is modeled implicitly using generalized estimating equations on the marginal distribution formulated by a Cox (1972) proportional hazards model. The other approach is the frailty model approach where the correlation is modeled explicitly by a multiplicative random effect called frailty. The frailty term corresponds to some common unobserved characteristics shared by all correlated failure times in a cluster. We have developed two frailty model based methods: one based on the semiparametric gamma frailty model (Su and Fan 2004) and the other based on the parametric exponential frailty model (Fan *et al.* 2009) to improve the computation speed.

The marginal method uses a robust log-rank statistic given in Appendix A in Equation S1. Su and Fan (2004) used an integrated log-likelihood shown in Equation S2 in Appendix A for the gamma frailty model; however, the **MST** package uses a Wald test statistic to improve stability and computation time. The exponential frailty model uses a score test statistic given in Equation S3 in Appendix A. The splitting statistic is calculated for each possible split and the largest value corresponds to the best split.

The **MST** package uses the **survival** package (Therneau 2017) to fit marginal and frailty models and the **MASS** package (Venables and Ripley 2002) to simulate from a multivariate normal distribution. In R, we can calculate the splitting statistic with the following logic where x is a numeric predictor variable and 0.5 is a candidate split:

The robust log-rank statistic is calculated by fitting the model

```
R> coxph(Surv(time, status) ~ I(x > 0.5) + cluster(id))$rscore
```

The Wald test statistic is calculated by fitting the model

```
R> coxph(Surv(time, status) ~ I(x > 0.5) + frailty.gamma(id, method = "em"),
+    control = coxph.control(eps = 1e-04, iter.max = 10,
+      toler.inf = sqrt(1e-09), outer.max = 5))$wald.test
```

The gamma frailty model can be computationally intensive, so the `coxph.control` function is utilized as shown above to speed up the convergence. The Wald test statistic should yield similar splits to the integrated log-likelihood studied by Su and Fan (2004).

The score test statistic for the exponential frailty model is calculated by utilizing matrix multiplication, simulated annealing with the `optim(..., method = "SANN")` function, and a quasi-Newton method to estimate the parameters in the model. The default values are used when optimizing parameters in the score test statistic, except that the maximum number of iterations is increased from 100 to 800 and the initial value estimates are set to 1.

The best choice for the splitting statistic depends on many factors including the true hazard function, the types of covariates in the model, the sample size, the cluster size, and the censoring rates. While the true hazard function is typically unknown, prior simulation studies and data analyses (Nunn, Fan, Su, Levine, Lee, and McGuire 2012) showed that the frailty model based approaches often performed better than the marginal model based approach in dental applications. However, the marginal and exponential frailty models are much less computationally intensive and often select the correct cutoff points even under model misspecification. All three splitting statistics outperformed the naive log-rank statistic that ignores the dependence. The **MST** package also allows users to grow a tree using a stratified or naive log-rank statistic, but we do not suggest using these splitting statistics except in rare circumstances. Interested readers are referred to our simulation studies (Su and Fan 2004; Fan *et al.* 2009).

Growing the initial tree is done by splitting nodes iteratively until either a small number of observations or uncensored times remain at the terminal node. The final tree model can be, theoretically, any subtree of the initial tree, and the number of subtrees can become massive. Following the CART approach, the initial tree is pruned to reduce the number of subtrees considered. Let T denote a binary tree, T^i denote the set of internal nodes, and $|T|$ denote the number of nodes of a tree. Defining $G(h)$ to represent the maximum splitting statistic on a particular node h, the split-complexity measure is:

$$G_\alpha(T) = G(T) - \alpha|T^i|, \tag{1}$$

where $G(T) = \sum_{h \in T} G(h)$ measures the overall goodness of splits of T, $|T^i|$ measures the complexity of the tree, and $\alpha \geq 0$ acts as a penalty for each additional split. The goodness of split measured at each split h is the robust log-rank statistic, the squared Wald test statistic, and the score test statistic for the marginal, gamma frailty, and exponential frailty models, respectively. Note that for a given split, each of these measures has an approximate $\chi^2(1)$ distribution when the sample size at h is large.

An efficient pruning algorithm is determined by comparing each branch T_h to a possible trim at node h and solving for α. Since the split complexity measure is 0 when T_h is trimmed and h becomes a terminal node, this equates to:

$$0 = G(T_h) - \alpha|T_h^i|,$$

which is equivalent to

$$\alpha = G(T_h)/|T_h^i|.$$

Thus, for any internal node h of the initial tree, the value $G(T_h)/|T_h^i|$ is calculated where T_h denotes the branch with h as its root. The weakest link of the tree is the node such that $G(T_h)/|T_h^i|$ is minimal. The subtree that prunes off the branch with the weakest link is taken and this process is repeated until we have pruned back to the root node. This procedure reduces the number of possible subtrees (which can be very massive) to a manageable value.

After pruning back the tree, we have to select the best-sized subtree. The best-sized subtree is taken as the tree with the largest splitting complexity measure in Equation 1 with a pre-specified α. LeBlanc and Crowley (1993) suggested α be fixed within the range of $2 \leq \alpha \leq 4$. Our previous simulation study (Fan *et al.* 2009) showed $\alpha = 4$ typically yielded the better performance compared with $\alpha = 2$ and $\alpha = 3$ for most models. The number of uncensored events, $\alpha = \log(n_0)$, can also be used to select the best-sized subtree. Since we have already used the sample to grow and prune the tree, we need to use a validation method for tree size selection. Two validation methods are implemented: the test sample approach and the bootstrap method. When the sample size is large, one may use a subset of the data to grow and prune the initial tree and the other subset to select the best-sized tree. When the sample size is small or moderate, one may use bootstrapping techniques as described previously in our papers.

There are alternative approaches to constructing survival trees, the most similar approach to the one implemented in package **MST** is available in package **rpart** (Therneau, Atkinson, and Ripley 2017). The two main differences are that **rpart** implements the CART procedure assuming independence and prunes the tree using cross-validation. It should also be noted that the **rpart** package uses an exponential scaling statistic to grow the tree, which differs from the naive log-rank statistic provided by **MST**. The **MST** package is slightly more automated where the user simply inputs the data and candidate variables and the `MST` function outputs the final tree; whereas, the **rpart** package allows a bit more control when pruning the tree. Both packages allow users to control how the tree is grown and automatically sort trees based on survival rate.

Other methods for constructing survival trees include the `ctree` function in the **partykit** package which constructs conditional inference trees with stopping rules (Hothorn, Hornik, and Zeileis 2006) and the **DStree** package which builds a tree for discrete-time survival data (Mayer, Larocque, and Schmid 2016). Nice features with the alternative approaches are the faster computation time compared with package **MST**. Splitting statistics that handle the dependence are typically more computationally intensive. Simulation studies specifically comparing trees constructed using the `rpart`, `ctree`, and `MST` functions are warranted.

3. The MST package

3.1. Simulating multivariate survival data

While the main purpose of the **MST** package is to construct multivariate survival trees, the package also includes the `rmultime` function to generate multivariate survival data. The details of generating survival data are described in Su, Fan, Wang, and Johnson (2006); the possible models that can be simulated are given in Table 1. The user specifies the coefficients (β_0 and $\vec{\beta_1}$), the cutoff values (c), the censoring rate, and the model with the respective parameters. The arguments of the function are described in the R package documentation.

`gamma.frailty`:

$$\lambda_{ij}(t) = \exp\left(\beta_0 + \vec{\beta_1} \cdot I(x_{ij} \leq c)\right) w_i \text{ with } w_i \sim \Gamma(1/\nu, 1/\nu).$$

`log.normal.frailty`:

$$\lambda_{ij}(t) = \exp\left(\beta_0 + \vec{\beta_1} \cdot I(x_{ij} \leq c) + w_i\right) \text{ with } w_i \sim N(0, \nu).$$

`marginal.multivariate.exponential`:

$$\lambda_{ij}(t) = \exp\left(\beta_0 + \vec{\beta_1} \cdot I(x_{ij} \leq c)\right) \text{ absolutely continuous.}$$

`marginal.nonabsolutely.continuous`:

$$\lambda_{ij}(t) = \exp\left(\beta_0 + \vec{\beta_1} \cdot I(x_{ij} \leq c)\right) \text{ not absolutely continuous.}$$

`nonPH.weibull`:

$$\lambda_{ij}(t) = \lambda_0(t) \exp\left(\beta_0 + \vec{\beta_1} \cdot I(x_{ij} \leq c)\right) w_i \text{ with } w_i \sim \Gamma(1/\nu, 1/\nu),\ \lambda_0(t) = \alpha\lambda t^{\alpha-1}.$$

Table 1: Models that can be simulated in the **MST** R package.

An example where multivariate failure times are simulated using this functionality and a tree is constructed is presented in Section 4.

There are several packages that can simulate survival data; those include packages **genSurv** (includes one time-dependent covariate; Araújo, Meira-Machado, and Faria 2015), **PermAlgo** (generates independent survival data conditional on covariates; Sylvestre, Evans, MacKenzie, and Abrahamowicz 2015), **simMSM** (simulates multistate models with possibly nonlinear baseline hazards; Reulen 2015), **survsim** (simulates survival data with recurrent events or competing risks; Moriña and Navarro 2014), and many others. However, to our knowledge, the `rmultime` function is the only function available in R that can simulate high-dimensional, multivariate survival data from marginal or frailty models.

3.2. Constructing multivariate survival trees

The `MST` function in the R package constructs the multivariate survival tree. The wrapper function grows, prunes, and selects the best tree. The main arguments are described below:

- `formula`: A linear survival model with the survival response, the predictors, and the cluster (or id) variable (e.g., `Surv(time, status) ~ x1 + x2 | id`).
- `data`: Data to grow and prune the tree.
- `method`: Indicates the method to handle correlation. Options include `"marginal"`, `"gamma.frailty"`, `"exp.frailty"`, `"stratified"`, and `"independence"` models as described in Section 2.
- `selection.method`: Indicates the method of selecting the best-sized tree. Options are `"test.sample"` (for large datasets) or `"bootstrap"` (for small/moderate datasets).
- `test`: Test sample to select the best-sized tree if `selection.method = "test.sample"`.
- `B`: Number of bootstrap samples if `selection.method = "bootstrap"`. A rough guideline for the number of bootstrap samples is $25 \leq$ `B` ≤ 100 following LeBlanc and Crowley (1993).

The MST function has some additional features to improve computational efficiency or allow users to see more details about the construction of the multivariate survival tree for a given dataset; these features are briefly described below:

- minsplit: Indicates minimum number of observations to continue splitting the branch.
- minevents: Indicates minimum number of uncensored events to continue splitting the branch.
- minbucket: Indicates minimum number of observations in any terminal node.
- maxdepth: Indicates maximum depth of the tree if user wants to stop growing the tree at a certain depth.
- mtry: Indicates number of variables randomly considered at each split.
- distinct: Indicates if all distinct cutpoints considered. If distinct = FALSE, then only cutpoints from delta to 1 - delta with nCutPoints are considered.
- LeBlanc: Indicates if the entire sample or the out-of-bag sample is used while bootstrapping. The interested reader can find more details in Fan *et al.* (2006).
- sortTrees: Indicates if trees should be sorted such that splits to the left have lower risk of failure.
- plot.Ga: Indicates if a plot of the goodness of split vs. tree size is produced.
- details: Indicates if detailed information should be given about how the final tree was constructed.

The plot.Ga and details parameters provide additional information regarding the construction of the tree for the given dataset, while the other additional features above can improve computational efficiency. The default sortTrees = TRUE will sort the trees such that splits to the left have higher rates of survival. However, it is possible that a terminal node on the left has lower rate of survival than a terminal node on the right if they do not share the same parent node.

The output of the MST function returns the initial tree, the trees pruned and considered in the best tree selection, and the best-sized tree. The final multivariate survival tree depends on the penalty (α) used, so four possible final trees are returned corresponding to $\alpha = 2$, 3, 4, and $\log(n_0)$. All trees are objects of class 'constparty' with methods defined in package **partykit** (Hothorn and Zeileis 2015). A 'constparty' tree requires observations that fall in the same terminal node to have the same prediction (or constant fit). While this requirement may not be met if two observations come from two different clusters, there are many benefits with treating the tree as a 'constparty' object. One is the added functionality for print/plot/predict methods. The plot method for 'party' objects produces Kaplan-Meier curves in each terminal node, which gives the survival rate for an observation with an average cluster effect. The predict method for 'party' objects returns the median survival time, but can easily be modified to return any prediction and handle the dependence (see examples in Section 4). Additionally, the 'constparty' object allows more compatibility with other R packages and provides additional functions to extract elements from the tree.

4. Illustrating examples

4.1. Simulation example

We illustrate the ability of the **MST** package to correctly construct multivariate survival trees with the following simulation. Suppose we have 200 clusters each with 4 patients per cluster. Suppose the failure times (e.g., time until illness) follow a marginal multivariate exponential distribution described below:

$$\lambda_{ij}(t) = \exp\left(\beta_0 + \vec{\beta_1} \cdot I(x_{ij} \leq c)\right) \quad \text{with } i = 1, \ldots, 200,\ j = 1, \ldots, 4, \qquad \beta_0 = -1,\ \vec{\beta_1} = (0.8, 0.8, 0, 0)^\top. \tag{2}$$

Suppose failure times have a 0.65 correlation within the cluster and approximately half the failure times are censored. Each patient has four covariates (x_1, x_2, x_3, x_4) with the first two covariates having cutpoints at 0.5 and 0.3, respectively, with the same β coefficients at 0.8, while the last two covariates have no effect on failure. The failure times can be simulated using the following commands:

```
R> set.seed(186117)
R> data <- rmultime(N = 200, K = 4, beta = c(-1, 0.8, 0.8, 0, 0),
+    cutoff = c(0.5, 0.3, 0, 0), model = "marginal.multivariate.exponential",
+    rho = 0.65)$dat
R> test <- rmultime(N = 100, K = 4, beta = c(-1, 0.8, 0.8, 0, 0),
+    cutoff = c(0.5, 0.3, 0, 0), model = "marginal.multivariate.exponential",
+    rho = 0.65)$dat
```

The dataset can be analyzed with the following commands:

```
R> fit <- MST(formula = Surv(time, status) ~ x1 + x2 + x3 + x4 | id,
+    data, test, method = "marginal", minsplit = 100, minevents = 20,
+    selection.method = "test.sample")
```

The output of the final tree is a ‘constparty’ object:

```
R> (tree_final <- getTree(fit, "4"))

Model formula:
Surv(time, status) ~ x1 + x2 + x3 + x4 | id

Fitted party:
[1] root
|   [2] x2 > 0.3
|   |   [3] x1 > 0.5: 1.694 (n = 208)
|   |   [4] x1 <= 0.5: 0.702 (n = 292)
|   [5] x2 <= 0.3
|   |   [6] x1 > 0.5: 0.977 (n = 112)
|   |   [7] x1 <= 0.5: 0.334 (n = 188)
```

```
Number of inner nodes:    3
Number of terminal nodes: 4
```

Figure 1 illustrates the construction of the multivariate survival tree for the simulated data. Panel A gives the large initial tree, Panel B gives the splitting statistic, $G_\alpha(T)$, based on the number of terminal nodes, and Panel C gives the final tree. In this case, the four split penalties considered yield the same final tree. The final tree correctly identifies that patients with $x_1 \leq 0.5$ and $x_2 \leq 0.3$ are at a higher risk of failure and it does not include x_3 or x_4 which have no effect on failure. The marginal model is best in this situation because the simulated data was generated without a frailty term; however, this is typically unknown. The gamma and exponential frailty models often yield similar final trees. Panels A and C were produced using the `plot` function and panel B was created with the `plot.Ga = TRUE` command in the `MST` function with slight modifications to increase resolution and readability.

Users can go further by fitting a model using the terminal nodes of the tree as the predictors. For example, the code below can fit a marginal model using the final tree. Each observation is sent down the tree and assigned to a terminal node; the marginal model is then fit with the terminal node as the predictor. Users can use a frailty term in `coxph` or use the **frailtypack** R package (Rondeau, Mazroui, and Gonzalez 2016) to fit a frailty model. Note that, in practice, it is highly recommended that a new dataset be used since the data was already used to grow the tree and data reuse is known to result in inferences that are overoptimistic.

```
R> data$term_nodes <- as.factor(predict(tree_final, newdata = data,
+    type = "node"))
R> coxph(Surv(time, status) ~ term_nodes + cluster(id), data = data)

Call:
coxph(formula = Surv(time, status) ~ term_nodes + cluster(id),
    data = data)

             coef exp(coef) se(coef) robust se    z       p
term_nodes4 1.034     2.811    0.139     0.193 5.35 9.0e-08
term_nodes6 0.949     2.582    0.178     0.174 5.46 4.6e-08
term_nodes7 1.692     5.428    0.166     0.223 7.57 3.7e-14

Likelihood ratio test=115  on 3 df, p=0
n= 800, number of events= 410
```

4.2. Computational speed

A major limitation with constructing multivariate survival trees is the computational intensity. There are several factors affecting computation speed: the model chosen to handle the dependence, the number of categorical and continuous predictors, the number of cutpoints considered, and the minimum node size to continue splitting the branch. To assess computation time, we simulated survival data with a marginal dependence structure setting $\rho = 0.65$ and varying the number of observations and predictors. Half of the predictors were cluster-specific factors (constant within cluster) with either 75% or 25% being continuous. Each

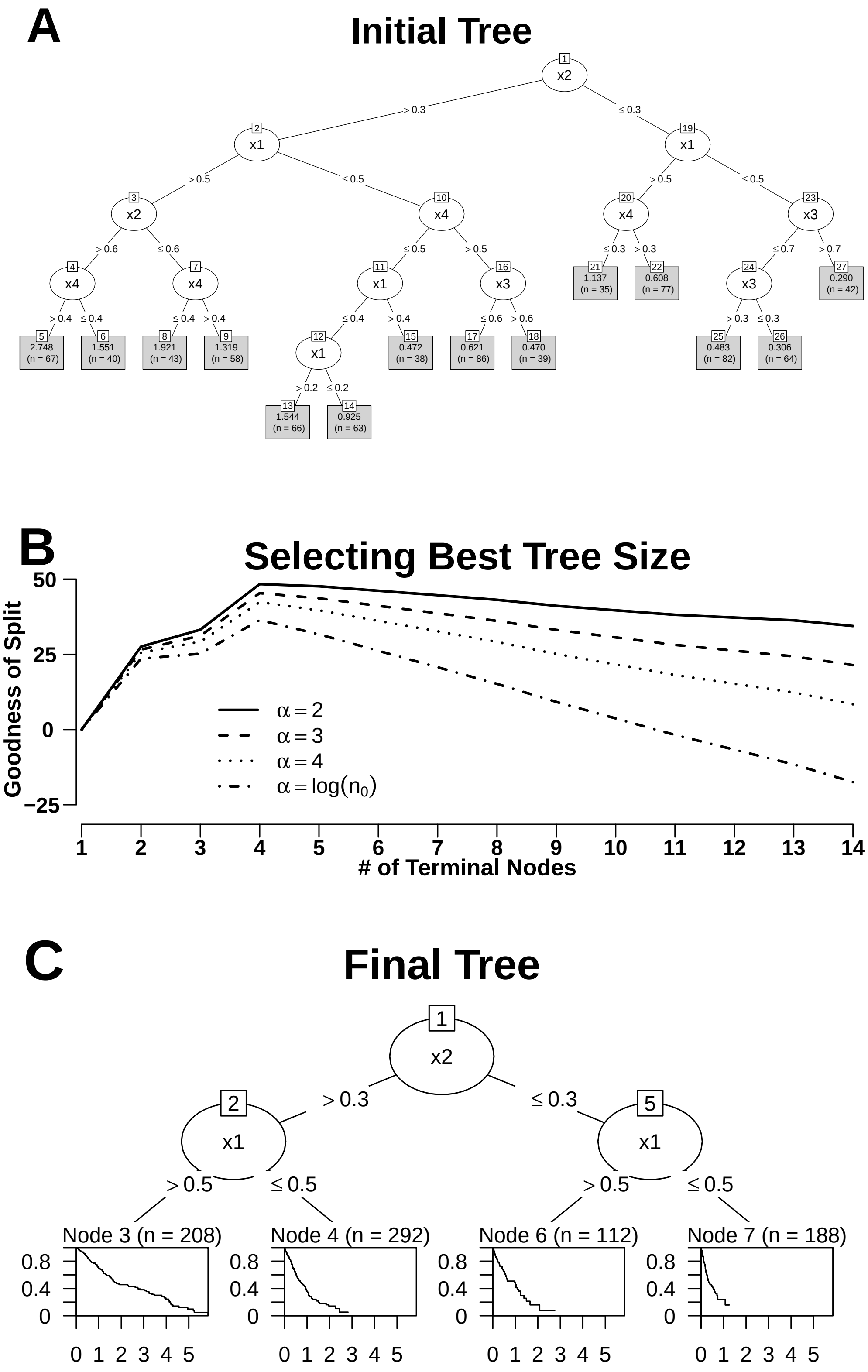

Figure 1: Construction of the multivariate survival tree. Panel A gives the large initial tree, Panel B gives the goodness of split based on the number of terminal nodes, and Panel C gives the final multivariate survival tree.

Dataset size ($N \times K$)	# of variables (# continuous vs. # nominal)	Marginal	Gamma frailty	Exponential frailty
200 × 4	8 (6 vs. 2)	0.9	1.7	0.6
200 × 4	8 (2 vs. 6)	0.4	0.8	0.2
200 × 4	24 (18 vs. 6)	2.8	7.0	1.7
200 × 4	24 (6 vs. 18)	1.3	3.0	1.1
200 × 4	64 (48 vs. 16)	7.3	18.7	4.9
200 × 4	64 (6 vs. 48)	3.5	7.5	2.4
500 × 10	8 (6 vs. 2)	5.8	22.0	3.5
500 × 10	8 (2 vs. 6)	2.8	8.0	1.8
500 × 10	24 (18 vs. 6)	24.4	67.3	8.0
500 × 10	24 (6 vs. 18)	8.4	26.0	5.3
500 × 10	64 (48 vs. 16)	52.0	195.7	25.1
500 × 10	64 (16 vs. 48)	24.0	75.5	11.2

Table 2: Approximate number of minutes to grow and select the best-sized tree.

continuous variable had around 100 distinct cutpoints and each categorical variable had 8 unordered levels. We set one continuous and one nominal failure-specific factor (which may vary within cluster) to be informative ($\beta = 1$). We used the default setting to grow the multivariate survival tree. Table 2 gives the number of minutes it takes to grow and select the best-sized tree using an Intel Core i7-4510U processor. While the **MST** package does not utilize more efficient software programs such as C++ or Fortran, most of the computation time was devoted to calculating the splitting statistic. Fitting thousands of Cox proportional hazards models, while adjusting for dependence, can be very time consuming.

The exponential frailty model was the fastest method, the marginal model was the second fastest, and the gamma frailty model was the slowest. Even for moderate sample sizes, the gamma frailty model can be very time consuming. The exponential frailty model was developed to improve computation time, while assuming a constant baseline hazard function.

The computational speed gained from using the marginal and exponential frailty models is essential for analyzing large datasets or using many bootstrap samples. Increasing the number of observations or predictors increases the computation time. Nominal categorical variables are converted to ordered by replacing levels with the estimated beta coefficients from a Cox proportional hazards model adjusting for the dependence. Thus, growing trees from datasets with more categorical predictors were faster due to having less distinct cutpoints. Using percentiles can greatly improve computation time. Users are advised to carefully consider the model dependence and input parameter settings to maximize accuracy and reduce computation time.

4.3. Analysis of tooth loss

We now construct multivariate survival trees to predict tooth loss from patients treated at the Creighton University School of Dentistry from August 2007 until March 2013. The data is available in the **MST** R package as object `Teeth`. Risk factors associated with tooth loss were grouped into four categories: demographic; health status; clinical variables at the patient level; and clinical variables at the tooth level. For many variables, a severity score for each

Variable	Name	Description
Tooth-level factors		
x_1	`mobil`	Mobility score (0–5)
x_2	`bleed`	Bleeding on probing (percentage)
x_3	`plaque`	Plaque score (percentage)
x_4	`pocket_mean`	Periodontal probing depth (tooth-level mean)
x_5	`pocket_max`	Periodontal probing depth (tooth-level max)
x_6	`cal_mean`	Clinical attachment level (tooth-level mean)
x_7	`cal_max`	Clinical attachment level (tooth-level max)
x_8	`fgm_mean`	Free gingival margin (tooth-level mean)
x_9	`fgm_max`	Free gingival margin (tooth-level max)
x_{10}	`mg`	Mucogingival defect
x_{11}	`filled`	Filled surfaces
x_{12}	`decay_new`	Decayed surfaces – new
x_{13}	`decay_recur`	Decayed surfaces – recurrent
x_{14}	`dfs`	Decayed and filled surfaces
x_{15}	`crown`	Crown (indicator)
x_{16}	`endo`	Endodontic therapy (indicator)
x_{17}	`implant`	Tooth implant (indicator)
x_{18}	`pontic`	Bridge pontic (indicator)
x_{19}	`missing_tooth`	Missing tooth (indicator)
x_{20}	`filled_tooth`	Filled tooth (indicator)
x_{21}	`decayed_tooth`	Decayed tooth (indicator)
x_{22}	`furc_max`	Furcation involvement (for molars only)

Table 3: Factors assessed that could potentially affect tooth loss – Part I.

tooth (tooth-level factor) and the average severity score over the entire mouth for each subject (patient-level factor) were included. Tables 3 and 4 give the fifty-one factors assessed that could potentially affect tooth loss. Factors were analyzed separately for molar and non-molar teeth.

The data consisted of 5,336 patients with periodontal disease with 25,694 molar teeth and 40,196 non-molar teeth. The average age was 56 years, with 49% men, 9% had Diabetes Mellitus, and 23% were smokers. For molars, 1,870 (7.3%) teeth were lost with the median tooth loss time of 0.53 years and the maximum tooth loss time of 5.58 years. For non-molars, 2,723 (6.8%) teeth were lost with the median tooth loss time of 0.54 years and the maximum tooth loss time of 5.59 years.

Due to the large sample size, a test sample was used to select the best-sized tree with one third of patients randomly assigned to the test sample. For the molar tooth analysis, data is partitioned with the following commands:

```
R> data("Teeth", package = "MST")
R> molar <- subset(Teeth, molar == 1)
R> set.seed(482046)
R> id.train <- sample(x = unique(molar$id), size = floor(nPatients * 2/3),
+    replace = FALSE)
```

Variable	Name	Description
Subject-level factors		
x_{23}	`bleed_ave`	Bleeding on probing (mean percentage)
x_{24}	`plaque_ave`	Plaque index (mean percentage)
x_{25}	`pocket_mean_ave`	Periodontal probing depth (mean of tooth mean)
x_{26}	`pocket_max_ave`	Periodontal probing depth (mean of tooth max)
x_{27}	`cal_mean_ave`	Clinical attachment level (mean of tooth mean)
x_{28}	`cal_max_ave`	Clinical attachment level (mean of tooth max)
x_{29}	`fgm_mean_ave`	Free gingival margin (mean of tooth mean)
x_{30}	`fgm_max_ave`	Free gingival margin (mean of tooth max)
x_{31}	`mg_ave`	Mucogingival defect (mean)
x_{32}	`filled_sum`	Filled surfaces (total)
x_{33}	`filled_ave`	Filled surfaces (mean)
x_{34}	`decay_new_sum`	New decayed surfaces (total)
x_{35}	`decay_new_ave`	New decayed surfaces (mean)
x_{36}	`decay_recur_sum`	Recurrent decayed surfaces (total)
x_{37}	`decay_recur_ave`	Recurrent decayed surfaces (mean)
x_{38}	`dfs_sum`	Decayed and filled surfaces (total)
x_{39}	`dfs_ave`	Decayed and filled surfaces (mean)
x_{40}	`filled_tooth_sum`	# of filled teeth
x_{41}	`filled_tooth_ave`	% of filled teeth
x_{42}	`decayed_tooth_sum`	# of decayed teeth
x_{43}	`decayed_tooth_ave`	% of decayed teeth
x_{44}	`missing_tooth_sum`	# of missing teeth
x_{45}	`missing_tooth_ave`	% of missing teeth
x_{46}	`total_tooth`	# of teeth
x_{47}	`dft`	# of decayed and filled teeth
Demographic factors		
x_{48}	`baseline_age`	Patient age at baseline (in years)
x_{49}	`gender`	Gender
Health status factors		
x_{50}	`diabetes`	Diabetes mellitus (indicator)
x_{51}	`tobacco_ever`	Tobacco use (indicator)

Table 4: Factors assessed that could potentially affect tooth loss – Part II.

```
R> molar_training <- molar[molar$id %in% id.train, ]
R> molar_test <- molar[!(molar$id %in% id.train), ]
```

To improve computational speed, only 50 equally spread out cutoff points from the 5th to the 95th percentiles were used as candidates for continuous variables with over 50 distinct cutoff points; this was specified via the option `distinct = FALSE`, together with `delta = 0.05` and `nCutPoints = 50`. A large number of observations was required in each terminal node, `minbucket = 100`, to ensure more accurate survival estimates. The exponential frailty model was used to handle the correlation and $\alpha = 4$ was used as the split penalty when selecting the best-sized tree. The multivariate survival trees are constructed with the following command:

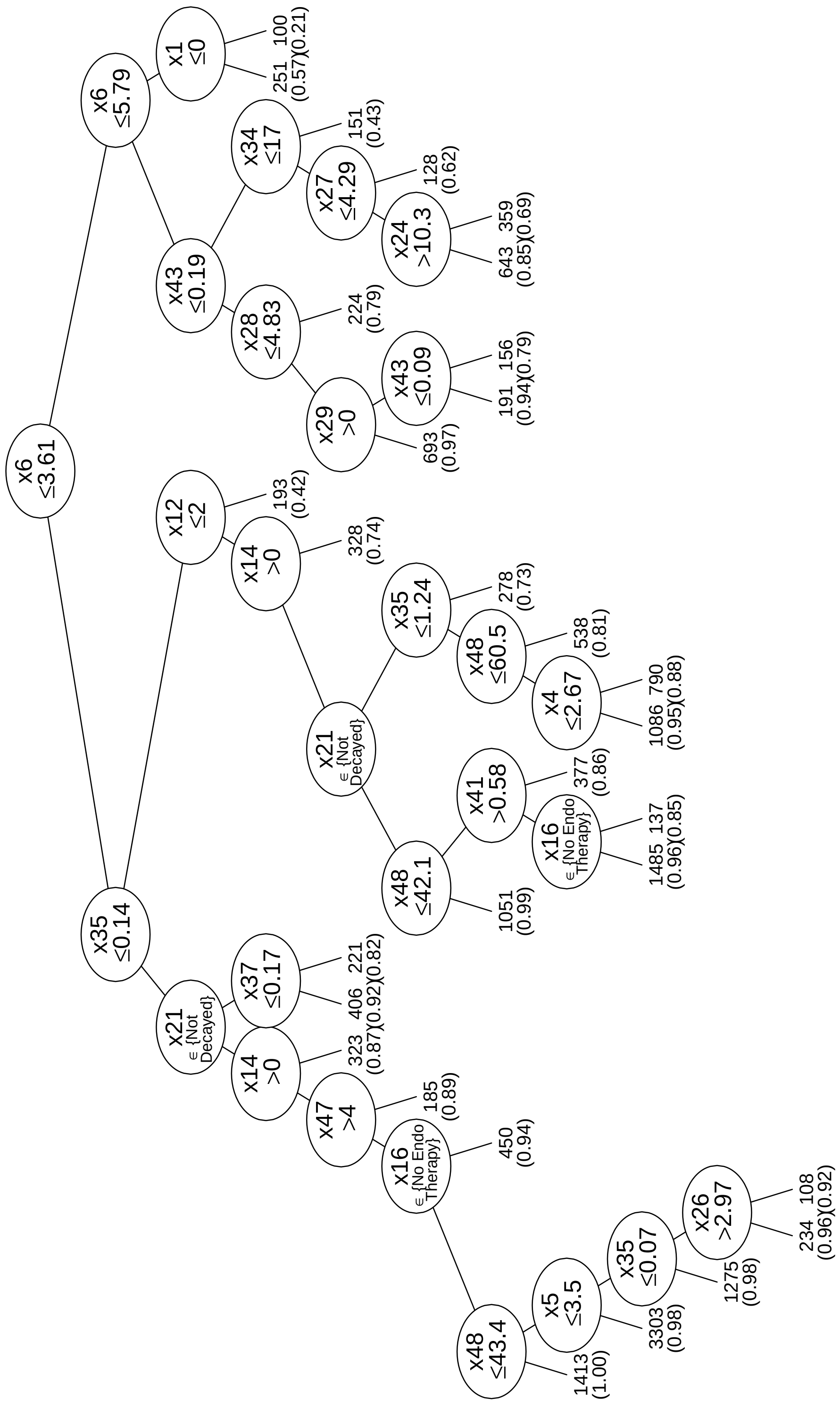

Figure 2: Multivariate survival tree for molar tooth loss. True statements move to the left; the number of teeth and estimated probability of survival at 3 years is reported in each terminal node.

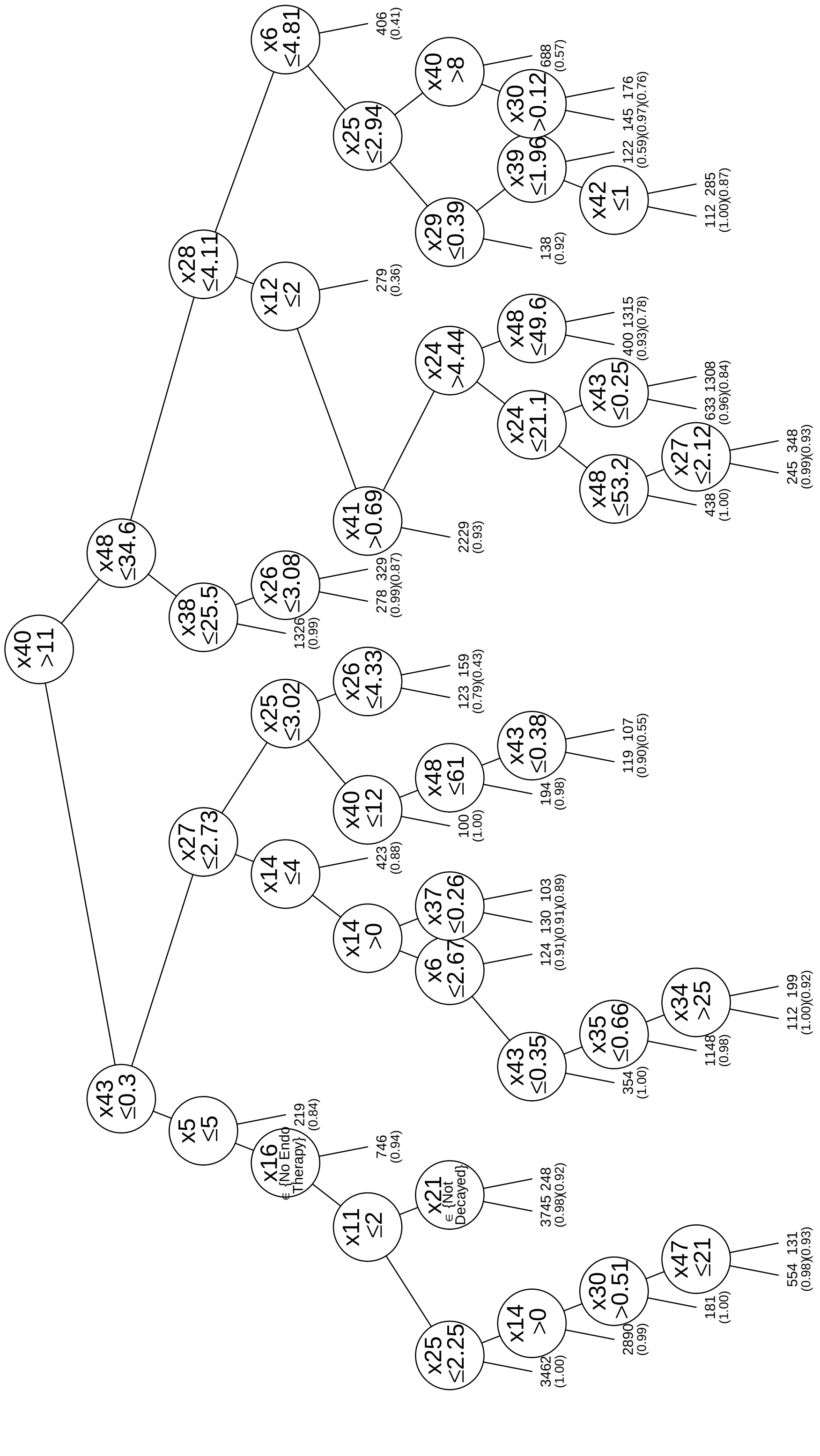

Figure 3: Multivariate survival tree for non-molar tooth loss. True statements move to the left; the number of teeth and estimated probability of survival at 3 years is reported in each terminal node.

```
R> form <- as.formula(paste0("Surv(time, event) ~ ",
+    paste0("x", 1:51, collapse = " + "), " | id"))
R> fit_molar <- MST(form, data = molar_training, test = molar_test,
+    method = "exp.frailty", selection.method = "test.sample",
+    distinct = FALSE, delta = 0.05, nCutPoints = 50, minevents = 10,
+    minbucket = 100)
```

Figures 2 and 3 give the multivariate survival trees for molar and non-molar teeth, respectively. Molar teeth are categorized into thirty possible terminal nodes. The risk of molar tooth loss is dependent on 20 factors (8 tooth-level factors and 12 subject-level factors). The tree for non-molar teeth is constructed with the same commands, except for using `molar == 0` and removing `x22` from the formula. Non-molar teeth are categorized into forty-three possible terminal nodes. Twenty-five factors (7 tooth-level factors and 18 subject-level factors) impacted the risk of non-molar tooth loss. Factors listed in the tree do not necessarily represent variable importance. It is possible that informative variables are not included in the final tree due to a masking effect. A detailed tooth analysis using multivariate survival random forests to assess variable importance will be studied in a subsequent paper.

The current dental prognosis is often based on clinical opinion where the clinician weighs several factors. However, an accurate prognosis is crucial to the development of an appropriate treatment plan (Mordohai, Reshad, Jivraj, and Chee 2007). There are many ways multivariate survival trees can be used to help clinicians make better decisions and improve patient care. One approach is to estimate the survival curves for each possible terminal node by fitting a marginal model that handles the correlation within subjects. The multivariate survival trees on tooth loss give the estimated survival rate for each terminal node. Each split to the left has higher rates of survival. The probability of survival at 3 years ranged from 0.999 to 0.211 for molar teeth and 1 to 0.358 for non-molar teeth. A combination of the survival curves and clinical experience can help prescribe better treatment.

5. Conclusions

To our knowledge, package **MST** is the only available package in R to construct multivariate survival trees. The **MST** package is easy-to-use and allows users to see and control how the survival trees are constructed. Tree based analysis automatically groups observations with differing outcomes and hence renders it an excellent research tool in medical prognosis or diagnosis. In addition, trees can handle data with a large number of predictors with mixed types (continuous, ordinal, and categorical with two or more levels) without the need for variable transformation, dummy variable creation, or variable selection. These off-the-shelf features of a data analysis tool are often required in the current era of big data. Over the years, we have received many requests from researchers in various fields to share our R codes. We hope that our newly developed R package, **MST**, will help facilitate the utilization of multivariate survival trees in many application fields.

Many factors must be considered when determining the appropriate model. The gamma frailty model performs well even when the underlying model is misspecified. However, the gamma frailty model takes much longer to fit than the marginal and exponential frailty models. The computational speed gained from using the marginal and exponential frailty models is essential for analyzing large datasets or using many bootstrap samples. Using percentiles as the candidate splits, instead of all distinct cutpoints, can improve computation speed at the expense of slightly less accuracy.

Future work will be devoted to improving computational efficiency and providing additional features and flexibility. Possible additions include allowing positive stable frailty models, providing more control when pruning a tree, and handling missing observations. The MST function also provides the framework to construct multivariate survival random forests. However, random forests are very computationally intensive, so methods and parameter settings that improve speed as assessed in Section 4.2 should be considered. Other faster ensemble methods, such as extremely randomized trees (Geurts, Ernst, and Wehenkel 2006), could also be implemented. While the main components to construct multivariate survival trees are available, we hope to continue improving computational speed and functionality in future versions of the **MST** package.

Acknowledgments

This research was supported in part by the National Institutes of Health grant R01-DE019656. We thank Ms. Nici Kimmes, DDS for her help in extracting the data from AxiUm.

References

Araújo AA, Meira-Machado L, Faria S (2015). ***genSurv**: Generating Multi-State Survival Data*. R package version 1.0.3, URL https://CRAN.R-project.org/package=genSurv.

Cox DR (1972). "Regression Models and Life-Tables." *Journal of the Royal Statistical Society B*, **34**(2), 187–202.

Davis R, Anderson Z (1989). "Exponential Survival Trees." *Statistics in Medicine*, **8**(8), 947–962. doi:10.1002/sim.4780080806.

Fan J, Nunn M, Su X (2009). "Multivariate Exponential Survival Trees and Their Application to Tooth Prognosis." *Computational Statistics & Data Analysis*, **53**(4), 1110–1121. doi:10.1016/j.csda.2008.10.019.

Fan J, Su X, Levine R, Nunn M, LeBlanc M (2006). "Trees for Correlated Survival Data by Goodness of Split, with Applications to Tooth Prognosis." *Journal of American Statistical Association*, **101**(475), 959–967. doi:10.1198/016214506000000438.

Geurts P, Ernst D, Wehenkel L (2006). "Extremely Randomized Trees." *Machine Learning*, **63**(1), 3–42. doi:10.1007/s10994-006-6226-1.

Gordon L, Olshen R (1985). "Tree-Structured Survival Analysis." *Cancer Treatment Reports*, **69**(10), 1065–1069.

Hothorn T, Hornik K, Zeileis A (2006). "Unbiased Recursive Partitioning: A Conditional Inference Framework." *Journal of Computational and Graphical Statistics*, **15**(3), 651–674. doi:10.1198/106186006x133933.

Hothorn T, Zeileis A (2015). "**partykit**: A Modular Toolkit for Recursive Partytioning in R." *Journal of Machine Learning Research*, **16**, 3905–3909.

LeBlanc M, Crowley J (1992). "Relative Risk Trees for Censored Survival Data." *Biometrics*, **48**(2), 411–425. doi:10.2307/2532300.

LeBlanc M, Crowley J (1993). "Survival Trees by Goodness of Split." *Journal of the American Statistical Association*, **88**(422), 457–467. doi:10.1080/01621459.1993.10476296.

Mayer P, Larocque D, Schmid M (2016). ***DStree**: Recursive Partitioning for Discrete-Time Survival Trees.* R package version 1.0, URL https://CRAN.R-project.org/package=DStree.

Mordohai N, Reshad M, Jivraj S, Chee W (2007). "Factors That Affect Individual Tooth Prognosis and Choices in Contemporary Treatment Planning." *British Dental Journal*, **202**, 63–72. doi:10.1038/bdj.2007.23.

Moriña D, Navarro A (2014). "The R Package **survsim** for the Simulation of Simple and Complex Survival Data." *Journal of Statistical Software*, **59**(2), 1–20. doi:10.18637/jss.v059.i02.

Nunn M, Fan J, Su X, Levine R, Lee H, McGuire M (2012). "Development of Prognostic Indicators Using Classification and Regression Trees for Survival." *Periodontology 2000*, **58**(1), 134–142. doi:10.1111/j.1600-0757.2011.00421.x.

R Core Team (2017). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Reulen H (2015). ***simMSM**: Simulation of Event Histories for Multi-State Models.* R package version 1.1.41, URL https://CRAN.R-project.org/package=simMSM.

Rondeau V, Mazroui Y, Gonzalez JR (2016). "**frailtypack**: An R Package for the Analysis of Correlated Survival Data with Frailty Models Using Penalized Likelihood Estimation or Parametrical Estimation." *Journal of Statistical Software*, **47**(4), 1–28. doi:10.18637/jss.v047.i04.

Su X, Calhoun P, Fan J (2018). ***MST**: Multivariate Survival Trees.* R package version 2.1, URL https://CRAN.R-project.org/package=MST.

Su X, Fan J (2004). "Multivariate Survival Trees: A Maximum Likelihood Approach Based on Frailty Models." *Biometrics*, **60**(1), 93–99. doi:10.1111/j.0006-341x.2004.00139.x.

Su X, Fan J, Wang A, Johnson M (2006). "On Simulating Multivariate Failure Times." *International Journal of Applied Mathematics & Statistics*, **6**(5), 8–18.

Sylvestre M, Evans T, MacKenzie T, Abrahamowicz M (2015). ***PermAlgo**: Permutational Algorithm to Generate Event Times Conditional on a Covariate Matrix Including Time-Dependent Covariates.* R package version 1.1, URL https://CRAN.R-project.org/package=PermAlgo.

Therneau T, Atkinson B, Ripley B (2017). ***rpart**: Recursive Partitioning and Regression Trees.* R package version 4.1-11, URL https://CRAN.R-project.org/package=rpart.

Therneau TM (2017). ***survival**: A Package for Survival Analysis in S.* R package version 2.41-3, URL `https://CRAN.R-project.org/package=survival`.

Therneau TM, Grambsch PM, Fleming T (1990). "Martingale-Based Residuals for Survival Models." *Biometrika*, **77**(1), 147–160. `doi:10.1093/biomet/77.1.147`.

Venables WN, Ripley BD (2002). *Modern Applied Statistics with S.* 4th edition. Springer-Verlag, New York. `doi:10.1007/978-0-387-21706-2`.

A. Splitting statistics formulas

Suppose that there are n units with K_i correlated failure times in the i-th unit, $i = 1, \ldots, n$. Let F_{ik} and C_{ik} be the failure time and censoring time corresponding to the k-th failure in the i-th unit, respectively. The observed data consist of $\{(Y_{ik}, \Delta_{ik}, \mathbf{X}_{ik}) : i = 1, \ldots, n; k = 1, \ldots, K_i\}$, where $Y_{ik} = \min(F_{ik}, C_{ik})$ is the observed failure time; $\Delta_{ik} = I(F_{ik} \leq C_{ik})$ is the failure indicator, which is 1 if $F_{ik} \leq C_{ik}$ and 0 otherwise; and $\mathbf{X}_{ik} = (x_{ik1}, \ldots, x_{ikp})$ denotes the pq-dimensional covariate vector associated with the k-th observation of the i-th unit. We consider only time-independent covariates. To ensure identifiability, it is assumed that the failure time vector $\mathbf{F}_i = (F_{i1}, \ldots, F_{iK_i})^\top$ is independent of the censoring time vector $\mathbf{C}_i = (C_{i1}, \ldots, C_{iK_i})^\top$ conditional on the covariate vector $\mathbf{X}_i = (\mathbf{X}_{i1}^\top, \ldots, \mathbf{X}_{iK_i}^\top)^\top$, $i = 1, \ldots, n$. We will use the following notations to write the splitting statistic. Let $\lambda_0(t)$ represent an unspecified baseline hazard function, β is an unknown regression parameter, and $I(\cdot)$ is the indicator function. For a constant change point c, x_{ikj} induces a binary partition of the data according to a continuous covariate X_j. If X_j is discrete, then the form $x_{ikj} \in A$ is considered where A can be any subset of its categories.

The marginal model develops the splitting statistic by formulating the hazard function of the k-th failure in the i-th unit as follows:

$$\lambda_{ik}(t) = \lambda_0(t) \exp(\beta \cdot I(x_{ikj} \leq c)).$$

Fan *et al.* (2006) showed that the splitting statistic, called the robust log-rank statistic, is:

$$G^M(c) = \frac{\left[\sum_{i=1}^n \sum_{k=1}^{K_i} \Delta_{ik} \left\{ I(x_{ikj} \leq c) - S^{(1)}(Y_{ik})/S^{(0)}(Y_{ik}) \right\}\right]^2}{\sum_{i=1}^n \sum_{k=1}^{K_i} \sum_{l=1}^{K_i} W_{ik} W_{il}}, \tag{S1}$$

where

$$W_{ik} = \Delta_{ik} \left\{ I(x_{ikj} \leq c) - \frac{S^{(1)}(Y_{ik})}{S^{(0)}(Y_{ik})} \right\} - \sum_{i'=1}^{n} \sum_{l=1}^{K_i} \frac{\Delta_{i'l} I(Y_{ik} > Y_{i'l})}{S^{(0)}(Y_{i'l})} \left\{ I(x_{i'kj} \leq c) - \frac{S^{(1)}(Y_{i'l})}{S^{(0)}(Y_{i'l})} \right\},$$

$$S^{(0)}(t) = \sum_{i=1}^{n} \sum_{k=1}^{K_i} I(Y_{ik} \geq t), \qquad S^{(1)}(t) = \sum_{i=1}^{n} \sum_{k=1}^{K_i} I(Y_{ik} \geq t) I(x_{ikj} \leq c).$$

While Equation S1 looks formidable, the calculation of $G^M(c)$ is computationally inexpensive because it is available in closed form and depends primarily on the evaluation of indicator functions.

The gamma frailty model develops the splitting statistic by formulating the hazard function as follows:

$$\lambda_{ik}(t) = \lambda_0(t) \exp(\beta \cdot I(x_{ikj} \leq c)) w_i,$$

where w_i denotes the frailty term for the i-th unit. The frailty term w_i is assumed to follow some known positive distribution; the **MST** package uses the common choice that $w_i \sim \Gamma(1/\nu, 1/\nu)$ where ν represents an unknown variance.

Su and Fan (2004) showed that the integrated log likelihood splitting statistic is:

$$\begin{aligned} G^G(c) = -n \ln \Gamma(\frac{1}{\nu}) - \frac{n}{\nu} \ln \nu + \\ \sum_{i=1}^{n} \left[\ln \Gamma(a_i) - a_i \ln b_i + \sum_{k=1}^{K_i} \Delta_{ik} \left\{ \ln \lambda_0(Y_{ik}) + I(x_{ikj} \leq c)\beta \right\} \right], \end{aligned} \tag{S2}$$

where

$$a_i = \frac{1}{\nu} + \sum_{k=1}^{K_i} \Delta_{ik}, \qquad b_i = \frac{1}{\nu} + \sum_{k=1}^{K_i} \Lambda_0(Y_{ik}) \exp(\beta \cdot I(x_{ikj} \leq c)),$$

and $\Lambda_0(\cdot)$ is the cumulative baseline hazard function.

The exponential frailty model develops the splitting statistic by formulating the hazard function of the k-th failure in the i-th unit as follows:

$$\lambda_{ik}(t) = \exp(\lambda_0 + \beta \cdot I(x_{ikj} \leq c)) w_i,$$

where $\exp(\lambda_0)$ is an unknown constant baseline hazard and w_i denotes the frailty term for the i-th unit, typically $w_i \sim \Gamma(1/\nu, 1/\nu)$. Note the exponential frailty model replaces $\lambda_0(t)$ in the gamma frailty model with $\exp(\lambda_0)$ for computational efficiency. We define $\theta = (\lambda_0, \nu)^\top$ and $m_i = \sum_{k=1}^{K_i} Y_{ik} I(x_{ikj} \leq c)$. Fan *et al.* (2009) showed that the splitting statistic, called the score test statistic, is:

$$G^E(c) = \left. \frac{U^2}{i_{\beta\beta} - \mathbf{i}_{\beta\theta}^\top \mathbf{I}_0^{-1} \mathbf{i}_{\beta\theta}} \right|_{(\lambda_0, \nu) = (\widehat{\lambda_0}, \widehat{\nu})}, \tag{S3}$$

where

$$\begin{aligned}
A_i &= \frac{1}{\nu} + \sum_{k=1}^{K_i} \Delta_{ik}, \\
B_i &= \frac{1}{\nu} + \exp(\lambda_0) \sum_{k=1}^{K_i} Y_{ik}, \\
U &= \sum_{i=1}^{n} \sum_{k=1}^{K_i} \Delta_{ik} - \exp(\lambda_0) \sum_{i=1}^{n} \frac{m_i A_i}{B_i}, \\
i_{\beta\beta} &= \exp(\lambda_0) \sum_{i=1}^{n} \frac{m_i A_i B_i - \exp(\lambda_0) m_i^2 A_i}{B_i^2}, \\
\mathbf{i}_{\beta\theta} &= (i_{\beta\lambda_0}, i_{\beta\nu})^\top, \\
i_{\beta\lambda_0} &= \exp(\lambda_0) \sum_{i=1}^{n} \frac{m_i A_i}{B_i^2} \left\{ B_i - \exp(\lambda_0) \sum_{k=1}^{K_i} Y_{ik} \right\}, \\
i_{\beta\nu} &= \frac{\exp(\lambda_0)}{\nu^2} \sum_{i=1}^{n} \frac{m_i}{B_i^2} (A_i - B_i), \\
\mathbf{I}_0 &= \begin{pmatrix} i_{\lambda_0\lambda_0} & i_{\lambda_0\nu} \\ i_{\nu\lambda_0} & i_{\nu\nu} \end{pmatrix}, \\
i_{\lambda_0\lambda_0} &= \frac{\exp(\lambda_0)}{\nu} \sum_{i=1}^{n} \frac{A_i}{B_i^2} \sum_{k=1}^{K_i} Y_{ik},
\end{aligned}$$

$$i_{\lambda_0\nu} = i_{\nu\lambda_0} = \frac{\exp(\lambda_0)}{\nu^2}\sum_{i=1}^{n}\frac{A_i - B_i}{B_i^2}\sum_{k=1}^{K_i} Y_{ik},$$

$$\begin{aligned} i_{\nu\nu} = {} & \frac{2n}{\nu^3}\frac{\Gamma'(\frac{1}{\nu})}{\Gamma(\frac{1}{\nu})} - \frac{3n}{\nu^3} + \frac{2n\ln(\nu)}{\nu^3} + \frac{n}{\nu^4}\frac{\Gamma''(\frac{1}{\nu})}{\Gamma'(\frac{1}{\nu})} - \frac{n}{\nu^4}\left(\frac{\Gamma'(\frac{1}{\nu})}{\Gamma(\frac{1}{\nu})}\right)^2 + \\ & \sum_{i=1}^{n}\left[\frac{2\Gamma'(A_i)}{\nu^3\Gamma(A_i)} - \frac{2\ln(B_i)}{\nu^3} - \frac{2A_i}{\nu^3 B_i} + \frac{1}{\nu^4}\frac{\Gamma''(A_i)}{\Gamma(A_i)} - \frac{1}{\nu^4}\left\{\frac{\Gamma'(A_i)}{\Gamma(A_i)}\right\}^2 - \frac{2}{\nu^4 B_i} + \frac{A_i}{\nu^4 B_i^2}\right]. \end{aligned}$$

Affiliation:

Peter Calhoun
Computational Science Research Center
San Diego State University
San Diego, CA, United States of America
E-mail: calhoun.peter@gmail.com

Xiaogang Su
Department of Mathematical Sciences
University of Texas
El Paso, TX, United States of America

Martha Nunn
Department of Periodontology
Creighton University
Omaha, NE, United States of America

Juanjuan Fan
Department of Mathematics and Statistics
San Diego State University
San Diego, CA, United States of America
E-mail: jjfan@mail.sdsu.edu
URL: https://edoras.sdsu.edu/~jjfan/

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
February 2018, Volume 83, Issue 12 *Submitted:* 2016-07-12
doi:10.18637/jss.v083.i12 *Accepted:* 2017-04-25