1 Introduction

1.1 Estimation road map for censored data

Our general strategy for estimator construction, selection, and performance assessment is entirely driven by the choice of a *loss function* for the full, uncensored data structure. Censored data can be handled simply by replacing the full data loss function by an observed data loss function with the same expectation. Our proposed estimation road map for censored data can be stated in terms of the following three main steps.

1. *Definition of the parameter of interest in terms of a loss function for the observed data.* For the full data structure, define the parameter of interest as the minimizer of the expected loss, or *risk*, for a loss function chosen to represent the desired measure of performance (e.g., squared error loss for a population mean). Apply the general estimating function methodology of van der Laan and Robins (2002) to map the *full, uncensored* data loss function into an *observed, censored* data loss function having the same expected value and leading to an efficient estimator of this risk.

2. *Construction of candidate estimators based on a loss function for the observed data.* Define a finite collection of candidate estimators for the parameter of interest based on a sieve of increasing dimension approximating the complete parameter space (e.g., recursive binary partitioning of the covariate space as in regression trees). For each element of the sieve, the candidate estimator is defined as the minimizer of the empirical risk based on the observed data loss function (e.g., within-node sample mean for the squared error loss).

3. *Cross-validation for estimator selection and performance assessment based on a loss function for the observed data.* Use cross-validation to estimate risk based on the observed data loss function and to select an optimal estimator among the candidates in Step 2. This step relies on the unified cross-validation methodology of van der Laan and Dudoit (2003) and their finite sample and asymptotic optimality results for cross-validation estimator selection for general data generating distributions, loss functions (possibly depending on a nuisance parameter), and estimators.

As described below, a number of common estimation procedures follow this approach in the full data situation, but depart from it when faced with the obstacle of evaluating the loss function in the presence of censoring. Tree-based methods, where candidates in Step 2 are generated by recursive binary partitioning of a suitably defined covariate space, provide a striking example of the chasm between estimation procedures for full data and censored data: regression trees for uncensored data (Breiman et al., 1984) vs. adaptations to censored data (Breiman, 2002, 2003; Ciampi et al., 1986; Davis and Anderson, 1989; Gordon and Olshen, 1985; LeBlanc and Crowley, 1992; Segal, 1988). Here, we argue that one can, and should, also adhere to the estimation road map in censored data situations. All that is required is to replace the full (uncensored) data loss function by an observed (censored) data loss function with the same expected value, i.e., risk. This key step can be achieved using the general estimating function methodology of van der Laan and Robins (2002). Note that we use the term estimator in a broad sense, to provide a unified treatment of multivariate prediction and density estimation based on censored data. Each of these problems can be dealt with according to the road map by the choice of a suitable loss function.

The present article introduces our general loss-based methodology for estimator construction, selection, and performance assessment in the context of tree-structured estimation with censored data. We focus on the choice of a loss function (Step 1 of the road map) and refer to van der Laan and Dudoit (2003) for details on the general methodology for generating candidates and for cross-validation selection (Steps 2 and 3). The remainder of this section reviews the literature on survival trees. The proposed methodology for tree-based multivariate regression and density estimation with censored data is described in Section 2. The approach is evaluated in Section 3 via simulation studies and analysis of CGH copy number and survival data from breast cancer patients. Section 4 summarizes our findings and discusses ongoing work. Section A is a technical appendix which includes example code for implementing the suggested approach.

1.2 Tree-based estimation

Presently, clinicians collect a tremendous amount of data on patients in the hopes of finding significant prognostic factors for diseases such as cancer. A common scenario in medical studies is that in which hundreds, possibly thousands, of covariates are collected on each patient along with a time

to event of interest. Examples of the event of interest can be recurrence of chronic illness, death from disease, or drop in a bodily measurement, e.g., white blood cell count. In addition to epidemiological, clinical, and histological variables, the covariates may include microarray measurements of transcript levels for thousands of genes or of DNA copy number for thousands of chromosomal locations. At the time of a study, some patients may have dropped out, been lost to follow-up, or not had the particular event. In this situation, the last date of follow-up is recorded and referred to as the censored time to event. One objective in these studies is to model time to event by the measured covariates for the purposes of predicting time to event for future patients and identifying which of the covariates are integral in affecting this outcome.

Over the past three decades, there have been numerous non-parametric and semi-parametric approaches suggested to deal with censored data. In tree-based estimation procedures, candidate estimators are generated by recursive binary partitioning of a suitably defined covariate space into *nodes* and an estimator is returned for each set in the final partition, i.e., each *terminal node* or *leaf*. Regression trees were first introduced by Morgan and Sonquist (1963) in their *Automatic Interaction Detection* (AID) program. The methodology was then generalized and formalized in the monograph on *Classification And Regression Trees* (CART) by Breiman et al. (1984). There are three main aspects to tree-structured estimation: (i) the splitting rule for generating partitions of the covariate space, i.e., generating the candidate estimators (cf. Step 2 of the road map); (ii) the selection of a 'right-sized' tree (cf. Step 3 of the road map); (iii) estimation of the parameter of interest within each node (cf. Step 1 of the road map). Solutions to each of these problems typically involve optimization of a loss-based criterion.

As suggested above, the CART methodology of Breiman et al. (1984) can be formulated in terms of the three main steps of our general road map. In the special case of regression trees for continuous outcomes, the loss function is the quadratic, or squared error, loss and the parameter of interest is the conditional expected value of an outcome given covariates. This loss function enters at two key stages of the tree building process: node *splitting* and tree *pruning* with cross-validation, corresponding to Steps 2 and 3 above. Specifically, the CART candidate estimators are generated by recursive binary partitioning of the covariate space using a splitting rule based on the decrease of within-node mean squared errors (MSE). The result is a sieve, starting with a single-node tree and running up to a very large tree

with numerous nodes. After growing a large tree, the loss function is used again for pruning and for selecting a 'right-sized' tree among the generated sequence of trees based on cross-validation risk estimation. The survival trees discussed in Breiman (2002, 2003) can also be viewed within this framework. In this context, the outcome is a right-censored survival time and parameters of interest may include the conditional expected value and median of the (log) survival time given covariates and the conditional survival function given covariates. Corresponding full data loss functions are the squared error, absolute error, and negative log-likelihood loss functions, respectively. However, an immediate difficulty arises with censored data when evaluating the loss function at the splitting and pruning stages. Common approaches for tree-based regression and density estimation bypass the risk estimation problem for censored outcomes by altering the splitting and pruning criteria in manners that are specific to right-censored survival times. As described next, some of these proposals deviate from the estimation road map in essential ways.

Previously proposed modifications to regression trees, often referred to as *survival trees*, fall into two categories based on their use of within-node homogeneity or between-node heterogeneity measures. Included in the first category are: Breiman (2002, 2003), Davis and Anderson (1989), Gordon and Olshen (1985), and LeBlanc and Crowley (1992). These approaches inherit the fundamental basis of CART, i.e., they rely on splitting rules which optimize a loss-based within-node homogeneity criterion, and use cost-complexity pruning and cross-validation to select a 'right-sized' tree from the sequence of candidate trees. However, they each propose a different loss function to accommodate censored survival data. Davis and Anderson (1989) base their split function on the negative log-likelihood of an exponential model; Gordon and Olshen (1985) use L_p, L_p Wasserstein, and Hellinger distances for within-node Kaplan-Meir estimates of the survival distribution; and LeBlanc and Crowley (1992) use the first step of a full likelihood estimation procedure for a Cox proportional hazards model with the same baseline hazard for each node implied by the partition of the covariate space. In the recent work of Breiman (2002, 2003) on survival trees and survival forests, the time-covariate space is partitioned by seeking splits that maximize the increase in the observed data log-likelihood for a constant hazards model within each node. In the case of random forests, maximal trees are grown until only one uncensored observation is left in each node and aggregated over bootstrap samples. The effect of covariates over time is traced by monitoring correlations of the con-

ditional cumulative hazard function with individual covariates based on only uncensored observations. Risk estimation for performance assessment and comparison with other procedures is based on the L_1 loss function between survival functions or predicted survival times, evaluated solely on uncensored observations (the implications of omitting censored observations in risk estimation are discussed in Section 2.2.2). Most of the methods described above thus rely on a negative log-likelihood loss function, with the explicit or implicit goal of estimating the conditional survival function given covariates, and differ mainly in their choice of model for the observed data likelihood within nodes. By partitioning the time-covariate space, rather than only the covariate space, the survival trees of Breiman (2002, 2003) seem to provide the least parametric estimation procedure. The choice of loss function is discussed further in Section 2.2.

In the second class of survival trees, Segal (1988) and Ciampi et al. (1986) employ two-sample log-rank test statistics for between-node heterogeneity measures. This approach leads to alternative methods for splitting and pruning, and thus deviates from standard tree methodology and all three steps in the road map.

Hothorn et al. (2002) consider bagging the survival trees produced by the above methods, with the aim of generating improved estimators of the conditional survival function. Given bootstrap partitions of the covariate space, a Kaplan-Meier estimator of the survival function is produced for each learning set observation based on bootstrap aggregated nodes, i.e., based on the union, over bootstrap survival trees, of nodes containing the given observation. Performance is assessed using the Brier score, which relies on the assumption of independent survival and censoring times (Graf et al., 1999).

In essence, existing survival tree methods all have in common that they bypass direct evaluation of the loss function in splitting and pruning, by replacing the full data loss-based criteria inherent in regression trees with alternatives specific to censored outcomes. In general, the splitting and pruning criteria seem to be chosen based on convenience for handling censored data and do not reduce to the preferred choice for uncensored data. That is, rather than specifying a loss function based on a parameter of interest as in the uncensored data case (e.g., L_2 loss for conditional expected value of survival time), the choice of a loss function seems to be limited by the ability to evaluate it on censored observations. In principle, one could be interested in other parameters than the conditional survival distribution (corresponding

to the negative log-likelihood loss function used in the above approaches), such as the conditional mean or median survival times, or the conditional survival function evaluated at a single point. In such cases, one should employ a different loss function, which is specific to the parameter of interest. Finally, existing methods do not provide adequate means for evaluating the performance of the resulting estimators: due to their inability to evaluate arbitrary loss functions for censored observations, the methods often produce risk estimates based on only the uncensored observations. Discarding censored observations could potentially lead to serious biases in performance assessment (Section 2.2.2). This general difficulty in evaluating risk for censored observations results in a discontinuity between the full and observed data worlds.

It is our intention to follow the loss-based estimation road map of Section 1.1 and derive estimators that link the full and censored data worlds with the following two requirements. First, when applied to uncensored data, the censored data methodology should reduce to the full data methodology for estimator construction, selection, and performance assessment. Second, we wish to incorporate external (to the estimator) covariate processes to allow for informative censoring and a gain in efficiency. Neither of these two requirements nor this methodology have been adopted by the aforementioned approaches. In contrast to these modifications, which depart from the standard tree building framework and the estimation road map, we propose to use the general estimating function methodology of van der Laan and Robins (2002) to map the full data loss function into an observed, censored data loss function having the same expected value and leading naturally to an efficient estimator of this risk to be used for tree building and performance assessment.

2 Tree-based estimation with right-censored data

This section elaborates on the main steps of our general approach to loss-based estimator construction, selection, and performance assessment with censored data. We emphasize the choice of a loss function and illustrate the methodology in the context of tree-structured estimators. In tree-based estimation procedures such as CART (Breiman et al., 1984), the candidate estimators in Step 2 of the road map are generated by recursive binary partition-

ing of a suitably defined covariate space. Univariate prediction, multivariate prediction, and density estimation can be handled within the same framework by simply specifying a suitable full data loss function for each of these problems. The estimating functions described in van der Laan and Robins (2002) yield efficient observed data loss functions to be used in the presence of censoring for node splitting, tree pruning, and performance assessment by cross-validation. The rest of the tree building procedure is retained and the reader is referred to Breiman et al. (1984) for details.

2.1 Model

2.1.1 Full data structure

In the full data world, let $\{X(t) : t \in I\!R^+\}$ be a multivariate stochastic process of interest, indexed by time t. Let T denote either a fixed endpoint of this stochastic process or a random survival time, and let $Z = \log T$. The *full data structure* is defined as $X = \bar{X}(T) = \{X(t) = (R(t), L(t)) : 0 \leq t \leq T\}$, where $R(t) = I(T \leq t)$, $L(t)$ is the covariate process, and T is now a function of X. Denote the distribution of the full data structure X by $F_{X,0}$. The covariate process $L(t)$ may contain both time-dependent and time-independent covariates. Denote the time-independent covariates by $W = L(0)$, a p-dimensional vector, measured at baseline. If T is fixed, then let $Z(t)$, $t \in \{t_0 = 0, \ldots, t_{m-1} = T\}$, be an outcome process of interest, included in $X(t)$.

2.1.2 Observed data structure

In the observed data world, we rarely see all of the relevant variables in the process $X = \bar{X}(T) = \{X(t) : 0 \leq t \leq T\}$. Rather, we observe the full data process $X(t)$ up to the minimum, $\tilde{T} = \min(T, C)$, of the survival time T and a univariate censoring variable C. This missing, or *censored*, survival data situation can be due to drop out or the end of follow-up. The *observed data structure* can be written as $O = (\tilde{T} = \min(T, C), \Delta = I(T \leq C), \bar{X}(\tilde{T}))$ and the censoring process as $A(t) = I(C < t)$. By convention, if T occurs prior to C, we set $C = \infty$; thus, C is always observed and one can rewrite the observed data structure as $O = (C, \bar{X}(C))$. The random variable O has a distribution $P_0 = P_{F_{X,0},G_0}$, indexed by the full data distribution, $F_{X,0}$, and the conditional distribution, $G_0(\cdot|X)$, of the censoring variable C given X.

Due to the fact that what we observe about X is determined by C, $G_0(\cdot|X)$ is referred to as the *censoring* or *coarsening mechanism*. The *survival function* for the censoring mechanism is denoted by $\bar{G}_0(c \mid X) = Pr_0(C \geq c \mid X)$.

We assume that for $c < T$, the Lebesgue hazard corresponding to the censoring mechanism given the full data X is:

$$\lambda_{C,0}(c|X) = Pr_0(C = c \mid C \geq c, X) = m(c, \bar{X}(c)),$$

for some measurable function, m. This assumption on the censoring mechanism, referred to as *coarsening at random* (CAR), holds if the censoring distribution only depends on the observed process $\bar{X}(c)$. If X does not include time-dependent covariates, then, under CAR, the censoring time C is conditionally independent of the survival time T given baseline covariates W. An important consequence of CAR is that it implies the following factorization for the density of the observed data $O = (C, \bar{X}(C))$ (with respect to a dominating measure satisfying CAR itself), into an F_X-part and a G-part,

$$p_{F_X,G}(o) = p_{F_X}(o)h(o),$$

where $h(o)$ is the density $g_{C|X}(c \mid x)$ and $p_{F_X}(o) = f_{F_X}(\bar{X}(t)) \mid_{t=c}$ only depends on the measure F_X. Denote by $\mathcal{G}(CAR)$ the set of all conditional distributions G of C given X satisfying CAR. Gill et al. (1997), van der Laan and Robins (2002) (Section 1.2.3, in particular), and Robins and Rotnitzky (1992) provide further, thorough explanations of CAR.

2.2 Definition of parameter of interest in terms of loss function

2.2.1 Full data loss function

In the full data world, assume that we have a sample, or learning set, of n independent and identically distributed (i.i.d.) observations, $X_1, \ldots, X_n$, from the distribution $F_{X,0}$. The parameter of interest, ψ_0, is a mapping $\psi : \mathcal{S} \to I\!R$, from a covariate space $\mathcal{S}$ into the real line $I\!R$. The space $\mathcal{S}$ is typically a subset of $I\!R^p$, corresponding to the baseline covariates W; $\mathcal{S}$ could also refer to other variables, such as the survival time T in survival function estimation or a time index t in multivariate prediction. Denote the parameter space by Ψ. The parameter ψ_0 is defined in terms of a *loss function*, $L(X, \psi)$,

as (one of) the minimizer(s) of the expected loss, or *risk*. That is, ψ_0 is such that

$$\begin{aligned} E_{F_{X,0}}[L(X,\psi_0)] &= \int L(x,\psi_0)dF_{X,0}(x) \\ &\equiv \min_{\psi\in\Psi} \int L(x,\psi)dF_{X,0}(x) = \min_{\psi\in\Psi} E_{F_{X,0}}[L(X,\psi)]. \end{aligned}$$

Note that we do not require uniqueness of the risk minimizer, rather we simply assume that there is a loss function whose risk is minimized by the parameter of interest ψ_0. To simplify notation, we may use the subscript 0 to refer to parameters of the underlying data generating distributions $F_{X,0}$ and G_0, that is, write $E_{F_{X,0}}[L(X,\psi)] = E_0[L(X,\psi)]$. The purpose of the loss function L is to quantify performance. Thus, depending on the parameter of interest, there could be numerous loss functions from which to choose. A common loss function is the squared error loss, $L(X,\psi) = (Z - \psi(W))^2$, corresponding to the conditional mean $\psi_0(W) = E_0[Z \mid W]$. As described in Sections 2.2.3 – 2.2.5 below, we focus on three types of full data loss functions, for the purposes of univariate prediction, multivariate prediction, and density estimation.

2.2.2 Observed data loss function

In the observed data world, we have a learning set of n i.i.d. observations, $O_1, \ldots, O_n$, from the right-censored data structure, $O_i \sim P_0 = P_{F_{X,0},G_0}$. Let the empirical distribution of $O_1, \ldots, O_n$ be denoted by P_n. The goal remains to find an estimator for a parameter ψ_0 defined in terms of the risk for a full data loss function $L(X,\psi)$, e.g., a predictor of the log survival time Z based on covariates W. An immediate problem is that a loss function such as the quadratic loss, $L(X,\psi) = (Z - \psi(W))^2$, cannot be evaluated for an observation O with censored survival time ($\Delta = 0$). Risk estimators based on only uncensored observations, such as $\frac{1}{n}\sum_i L(X_i,\psi)\Delta_i$, are biased for $E_0[L(X,\psi)]$ and, in particular, estimate the quantity $E_0[L(X,\psi)\bar{G}_0(T|X)]$ which is not minimized by the parameter of interest ψ_0.

The general solution is to replace the full (uncensored) data loss function by an observed (censored) data loss function with the same expected value, i.e., risk. The *general estimating function methodology* of van der Laan and Robins (2002) can be used to link the observed data world to

the full data world. Specifically, the methodology allows full data estimating functions, $D(X)$, to be mapped into observed data estimating functions, $IC(O \mid Q, G, D)$, indexed by *nuisance parameter* G and, possibly, $Q = Q(F_X)$. The estimating functions satisfy

$$E_{P_0}[IC(O \mid Q, G, D)] = E_{F_{X,0}}[D(X)], \qquad \text{if } G = G_0 \text{ or } Q = Q_0 = Q(F_{X,0}).$$

In our specific application, the full data estimating function is the loss function, $D(X) = L(X, \psi)$, and the risk for a given estimator ψ is viewed as the full data parameter of interest, $\theta_0 = E_0[D(X)] = E_0[L(X, \psi)]$. Observed data loss functions are obtained from the estimating functions IC, that is, $L(O, \psi \mid \eta_0) = IC(O \mid Q_0, G_0, L(\cdot, \psi))$ is an observed data loss function with the same risk as the full data loss function $L(\cdot, \psi)$, where η_0 denotes the nuisance parameters (Q_0, G_0)

$$\int L(o, \psi \mid \eta_0) dP_0(o) = \int L(x, \psi) dF_{X,0}(x).$$

Inverse probability of censoring weighted loss function. The *inverse probability of censoring weighted* (IPCW) estimating function was introduced by Robins and Rotnitzky (1992). Its name derives from the fact that the full data function $D(X)$ is weighted by the inverse of a censoring probability. This estimating function is written as:

$$IC(O \mid G, D) = D(X) \frac{\Delta}{\bar{G}(T|X)}, \tag{1}$$

where $\bar{G}$ is a conditional survival function for C given X and $\Delta = I(T \leq C)$ is the censoring indicator. Given that

$$E_0[\Delta|X] = Pr_0(C \geq T|X) = \bar{G}_0(T|X) > 0, \qquad F_{X,0}\text{-a.e.},$$

one has

$$E_0\left[\frac{D(X)\Delta}{\bar{G}_0(T|X)}\right] = E_0\left[E_0\left[\frac{D(X)\Delta}{\bar{G}_0(T|X)} \mid X\right]\right] = E_0\left[\frac{D(X)}{\bar{G}_0(T|X)} E_0[\Delta|X]\right] = E_0\left[D(X)\right].$$

This suggests the IPCW observed data loss function, $L(O, \psi \mid \eta_0) = IC(O \mid G_0, L(\cdot, \psi))$, with nuisance parameter $\eta_0 = G_0$. The corresponding risk estimator is the empirical mean

$$\hat{\theta}_n = \frac{1}{n} \sum_{i=1}^{n} L(O_i, \psi \mid \eta_n) = \frac{1}{n} \sum_{i=1}^{n} L(X_i, \psi) \frac{\Delta_i}{\bar{G}_n(T_i|X_i)},$$

where η_n represents $\bar{G}_n$, an estimator of the nuisance parameter $\bar{G}_0$ derived under the CAR assumption for the censoring mechanism, i.e., by considering censoring mechanisms $G \in \mathcal{G}(CAR)$. For such models, the estimator $\bar{G}_n(T|X)$ is a function of $O = (C, \bar{X}(C))$ and thus the resulting risk estimator $\hat{\theta}_n$ depends only on the observed data structure, $O_1, \ldots, O_n$. Conditions for the IPCW estimating function to provide a consistent risk estimator are that $\bar{G}_0(T|X) > \delta > 0$, $F_{X,0}$-a.e., for some $\delta > 0$, and that $\bar{G}_n$ is a consistent estimator for $\bar{G}_0$. For example, if a Cox proportional hazards model is assumed for the censoring mechanism G, then

$$\lambda_C(t \mid X) = \lambda_0(t) \exp(\beta^T J(t)),$$

where $J(t) = f(L(t))$ is a set of covariates extracted from the process $\bar{L}(t) = \{L(s) : 0 \leq s \leq t\}$ for some given $\mathbb{R}^k$-valued function f. Standard software can then be employed to obtain maximum (partial) likelihood estimators of the baseline hazard function λ_0 and the regression coefficients β (e.g., `coxph` function in R).

Doubly robust inverse probability of censoring weighted loss function. The *doubly robust inverse probability of censoring weighted* (DR-IPCW) estimating function is written as:

$$IC(O \mid Q, G, D) = \frac{D(X)\Delta}{\bar{G}(T|X)} + \int E_{Q,G}\left[\frac{D(X)\Delta}{\bar{G}(T|X)} \mid \bar{X}(u), \tilde{T} \geq u\right] dM_G(u), \tag{2}$$

where

$$dM_G(u) = I(\tilde{T} \in du, \Delta = 0) - I(\tilde{T} \geq u)\lambda_C(u|X)du$$

and Q is defined as $Q(F_X)(c, \bar{X}(c)) = p_{F_X}(o) = f_{F_X}(\bar{X}(t)) \mid_{t=c}$. The nuisance parameter Q thereby identifies the F_X-part of the density for the observed data, $O = (C, \bar{X}(C))$, under the CAR assumption, so that the pair of nuisance parameters $(Q(F_X), G)$ identify the data generating distribution $P_{F_X,G}$ (Section 2.1.2). The DR-IPCW estimating function yields the observed data loss function $L(O, \psi \mid \eta_0 = (Q_0, G_0)) = IC(O \mid Q_0, G_0, L(\cdot, \psi))$, with nuisance parameter $\eta_0 = (Q_0, G_0)$. The so-called *double robustness* property of the estimating function $IC(O \mid Q, G, D)$ refers to the fact that $E_{P_0}[L(O, \psi \mid \eta = (Q, G))] = E_{F_{X,0}}[L(X, \psi)]$ if either $G = G_0$ and $\bar{G}_0(T \mid X) > 0$, $F_{X,0}$-a.e., or $Q(F_X) = Q(F_{X,0})$, where G and $Q(F_X)$ refer, respectively, to candidates for the censoring mechanism and the full data generating distribution. Thus,

the double robustness property allows misspecification of either the censoring mechanism or of part of the full data generating distribution. The empirical mean

$$\hat{\theta}_n = \frac{1}{n} \sum_{i=1}^{n} L(O_i, \psi \mid \eta_n)$$

now represents a *locally efficient estimator* of the risk $\theta_0 = E_0[L(X, \psi)]$, where $\eta_n = (Q_n, G_n)$ denotes a locally consistent estimator of the nuisance parameter $\eta_0 = (Q_0, G_0)$ under CAR. That is, if either Q_n is consistent for Q_0 or G_n is consistent for G_0, then $\hat{\theta}_n$ is a consistent estimator of θ_0, and if both Q_n and G_n are consistent, then $\hat{\theta}_n$ is asymptotically efficient (van der Laan and Robins, 2002).

We stress that in the absence of censoring, i.e., when $\Delta \equiv 1$ and $C \equiv \infty$, both the IPCW and DR-IPCW observed data loss functions reduce to the full data loss function, $L(O, \psi \mid \eta_0) = L(X, \psi)$. This ensures that the censored and full data estimators coincide when there is no censoring. In addition, one can estimate the nuisance parameter $\bar{G}_0$ in the IPCW and DR-IPCW loss functions using other covariates than those for ψ, in order to allow for informative censoring and a gain in efficiency. The methodology is illustrated below for three types of loss functions using the simple IPCW estimating function; one can proceed similarly for the DR-IPCW estimating function. Properties of the IPCW and DR-IPCW estimating functions are discussed in detail in van der Laan and Robins (2002).

2.2.3 Univariate prediction

In the univariate prediction setting, the full data structure of interest is $X = (W, Z)$, and one is concerned with predicting a univariate outcome Z, such as the log survival time $Z = \log T$, based on a vector of covariates W. The parameter of interest in regression trees (continuous outcome Z) is typically the conditional expectation, $\psi_0(W) = E_0[Z \mid W]$, corresponding to the quadratic (L_2), or squared error, loss function, $L(X, \psi) = (Z - \psi(W))^2$. Another parameter of interest could be the conditional median, $\psi_0(W) = \text{Median}_0[Z \mid W]$, corresponding to the absolute error (L_1) loss function, $L(X, \psi) = |Z - \psi(W)|$.

The IPCW observed data loss function for the quadratic loss is

$$L(O, \psi \mid \eta_n) = (Z - \psi(W))^2 \frac{\Delta}{\bar{G}_n(T|X)},$$

where $\Delta = I(T \leq C)$ is the censoring indicator and η_n is an estimator of the nuisance parameter $\eta_0 = G_0$, corresponding to the conditional survival function $\bar{G}_0$ for the censoring time C given full data X. Under the coarsening at random (CAR) assumption, one can estimate $\bar{G}_0(\cdot|X)$ by $\bar{G}_n(\cdot|X)$, a function only of the observed data structure O.

In classification trees (categorical outcome Z), the parameter of interest involves the class conditional probabilities, $Pr_0(z|W)$. For the indicator loss function, $L(X, \psi) = I(Z \neq \psi(W))$, the optimal parameter is $\psi_0(W) = \text{argmax}_z Pr_0(z \mid W)$, the class with maximum probability given covariates W. One could also use a loss function which incorporates differential misclassification costs. Note that in the standard CART methodology, Breiman et al. (1984) favor replacing the indicator loss function in the splitting rule by measures of node impurity, such as the entropy, Gini, or twoing indices (Chapter 4). The indicator loss function is still used for pruning and performance assessment. It turns out that the entropy criterion corresponds to the negative log-likelihood loss function, $L(X, \psi) = -\log \psi(X)$, and parameter of interest $\psi_0(X) = Pr_0(Z|W)$. Likewise, the Gini criterion corresponds to the loss function $L(X, \psi) = 1 - \psi(X)$, with parameter of interest $\psi_0(X) = 1$ if $Z = \text{argmax}_z Pr_0(z \mid W)$ and 0 otherwise. These modifications thus fall within our framework and amount to using different loss functions for the same parameter at different stages of the tree building process.

2.2.4 Multivariate prediction

In the multivariate setting, consider m outcomes of interest, such as the time-dependent outcome process $Z(t)$ included in $X(t)$, with $t \in \{t_0 = 0, \ldots, t_{m-1} = T\}$ and T fixed. Here the parameter of interest is the $m \times 1$ conditional mean vector $\psi_0(\cdot, W) = E_0[Z(\cdot) \mid W]$. A corresponding loss function can be defined as $L(X, \psi) = (Z(\cdot) - \psi(\cdot, W))^\top \Omega(W)(Z(\cdot) - \psi(\cdot, W))$, for a symmetric matrix function $\Omega(W)_{m \times m}$. A natural choice for $\Omega(W)$ is the inverse of the conditional covariance matrix $\Sigma(W)$ of the outcome process $Z(t)$ given covariates W, $\Sigma(W) = E_0\Big[(Z(\cdot) - E_0[Z(\cdot) \mid W])(Z(\cdot) - E_0[Z(\cdot) \mid$

$W])^\top \mid W\Big]$. Such a loss function takes into account the dependence structure among responses. As in the univariate outcome case, the corresponding IPCW observed data loss function is

$$L(O, \psi \mid \eta_n) = (Z(\cdot) - \psi(\cdot, W))^\top \Omega(W)(Z(\cdot) - \psi(\cdot, W)) \frac{\Delta}{\bar{G}_n(T|X)}.$$

Note that using this type of loss function for regression trees amounts to creating partitions of the time-covariate space using transformed outcomes $\Omega(W)^{1/2} Z(\cdot)$, where different choices of $\Omega(W)$ correspond to different notions of distance. Although risk is minimized by the conditional mean vector $\psi_0(\cdot, W) = E_0[Z(\cdot) \mid W]$ for arbitrary $\Omega(W)$, different choices of $\Omega(W)$ lead to estimators with different properties. In practice, one may work with a matrix $\Omega(W)$ that is diagonal, constant in W, or has a particular parametric representation. Previous approaches for multivariate responses in the context of linear regression have relied on canonical analysis to perform regression on transformed versions of the responses (Breiman and Friedman, 1997).

2.2.5 Density estimation

Here the parameter of interest is the joint density $\psi_0(T, W) = f_0(T, W)$, and the loss function is the negative log-likelihood, $L(X, \psi) = -\log \psi(T, W)$ (cf. Kullback-Leibler divergence). Again, the corresponding IPCW observed data loss function is simply

$$L(O, \psi \mid \eta_n) = -\log \psi(T, W) \frac{\Delta}{\bar{G}_n(T|X)}.$$

The resulting joint density estimator can then be used to obtain the conditional survival or hazard functions given covariates W.

As in previously proposed survival tree methods, one could also use as loss function the negative log-likelihood for the observed data

$$L(O, f) = -\log p_f(o),$$

where $p_f(o) = p_{F_X}(o) = f_{F_X}(\bar{X}(t)) \mid_{t=c}$ is the F_X-part of the observed data likelihood under CAR and f denotes the joint density corresponding to F_X (Section 2.1.2). Indeed, the risk $E_0[L(O, f)]$ is minimized at the true underlying density $f = f_0$. Different procedures consider different models for f

within each node (Breiman, 2002, 2003; Davis and Anderson, 1989; LeBlanc and Crowley, 1992).

One should keep in mind the following issues when choosing between the IPCW loss function $L(O, f \mid \eta_0)$ and the observed data log-likelihood $L(O, f)$. Firstly, the choice $L(O, f)$ corresponds to minimizing the risk $E_0[L(O, f)] = -\int \log p_f(o) dP_0(o)$, which involves the underlying data generating distribution $P_0 = P_{F_{X,0},G_0}$, while we might only be concerned with the risk $E_0[L(X, f)] = -\int \log f(x) dF_{X,0}(x)$, which does not depend on G_0. Secondly, unlike the IPCW or DR-IPCW loss functions, the $L(O, f)$ choice has the advantage that it does not require estimating a nuisance parameter η_0. Thirdly, in order to handle censored observations in likelihood calculations, methods based on the observed data loss function $L(O, f)$ assume coarsening at random, i.e., independence of the survival and censoring times given covariates. For example, for a within-node Cox proportional hazards model, consistent estimation of the parameters relies on independence of the survival and censoring times given covariates in the model. The implications of the coarsening at random assumption depend on the complexity of the model under consideration and should be most problematic in the early stages of procedures based on forward selection, such as tree estimators. However, such modeling assumptions are made for the purpose of generating candidate estimators and the final selected estimator may still be a good estimator of the density f_0.

2.3 Constructing piecewise constant estimators based on censored data

In general, it is not feasible to consider all possible candidate estimators $\hat{\psi}$ in the parameter space Ψ and, in Step 2 of the road map, one generates a sequence of candidates according to some search procedure. Tree-based estimators correspond to one such procedure analogous to forward selection (followed by backward deletion at the pruning stage). Define a *sieve*, $\{\Psi_k\}$, $\Psi_k \subset \Psi$, of increasing dimension approximating the complete parameter space Ψ

$$\Psi_k \equiv \left\{ \psi_{I,\beta}(\cdot) = \sum_{j \in I} \beta_j \phi_j(\cdot) : \beta,\, I,\, |I| \leq k \right\},$$

where the basis functions are set indicators, $\phi_j(s) = I(s \in S_j)$, and the subsets $S_j \subset \mathcal{S}$, $j \in I$, of the covariate space $\mathcal{S}$ are disjoint ($S_j \cap S_{j'} = \emptyset$,

$j \neq j'$) and exhaustive ($\mathcal{S} = \cup_{j \in I} S_j$). The goal is to identify for each k the parameter $\psi_{0k} \in \Psi_k$ with minimum risk, $\psi_{0k} = \text{argmin}_{\psi \in \Psi_k} E_0[L(X, \psi)] = \text{argmin}_{\psi \in \Psi_k} E_0[L(O, \psi \mid \eta_0)]$. In practice, one seeks the empirical analogue, $\hat{\psi}_k = \psi_k(\cdot \mid P_n)$, which minimizes the *empirical risk*, i.e., the *resubstitution error*,

$$\psi_k(\cdot \mid P_n) \equiv \text{argmin}_{\psi \in \Psi_k} \int L(o, \psi \mid \eta_n) dP_n(o), \tag{3}$$

where η_n represents an estimator of the nuisance parameter η_0 derived under the CAR assumption for the censoring mechanism.

Tree-structured estimators such as CART (Breiman et al., 1984) do not search over all index sets I with $|I| \leq k$, but rather approximate the minimum by recursive binary partitioning of the covariate space $\mathcal{S}$ according to a loss-based node splitting rule. In this setting, the S_j correspond to terminal nodes and k indexes the 'size' of the tree, measured by the number of terminal nodes $|I| = k$ (or by the complexity parameter α, as described in Section 2.4); in particular, for $k = 1$, S_1 is the root node, $\mathcal{S}$. Thus, as detailed next, trees tackle the optimization problem in equation (3) in two steps: generation of the index sets I by a forward partitioning algorithm and minimization over coefficients β for a given index set I.

2.3.1 Within-node estimation: minimizing risk over coefficients β for a given index set I

Given an index set I, the node coefficients β are defined as the minimizers $\hat{\beta}_I = \beta_I(P_n)$ of the empirical risk

$$\begin{aligned} \hat{\beta}_I &= \text{argmin}_\beta \int L(o, \psi_{I,\beta} \mid \eta_n) dP_n(o) \\ &= \text{argmin}_\beta \sum_{i=1}^{n} L(X_i, \psi_{I,\beta}) \frac{\Delta_i}{\bar{G}_n(T_i|X_i)}. \end{aligned}$$

This generally involves solving the following estimating equation

$$0 = \sum_{i=1}^{n} \frac{d}{d\beta} L(X_i, \psi_{I,\beta}) \frac{\Delta_i}{\bar{G}_n(T_i|X_i)}.$$

The resulting estimator is denoted by $\hat{\psi}_I = \psi_I(\cdot \mid P_n)$. Below are solutions for each of the three loss functions defined in Sections 2.2.3 – 2.2.5.

Univariate prediction. For the quadratic loss function, $L(X_i, \psi_{I,\beta}) = (Z_i - \psi_{I,\beta}(W_i))^2 = (Z_i - \beta_j)^2$, if $W_i \in S_j$. Hence

$$\hat{\beta}_I = \text{argmin}_\beta \sum_{j \in I} \sum_{i=1}^{n} I(W_i \in S_j)(Z_i - \beta_j)^2 \frac{\Delta_i}{\bar{G}_n(T_i|X_i)}$$

and

$$\hat{\beta}_{I,j} = \sum_{i=1}^{n} \frac{I(W_i \in S_j)\frac{\Delta_i}{\bar{G}_n(T_i|X_i)}}{\sum_{i=1}^{n} I(W_i \in S_j)\frac{\Delta_i}{\bar{G}_n(T_i|X_i)}} \, Z_i, \qquad j \in I.$$

Thus, the coefficients $\hat{\beta}_I$ are weighted means of the outcome in nodes S_j, $j \in I$. In the absence of censoring ($\Delta_i \equiv 1$, $C_i \equiv \infty$), $\hat{\beta}_{I,j}$ reduces to the standard regression tree prediction, that is, to the average outcome in node S_j.

Multivariate prediction. In this setting, $\psi_{I,\beta}(t, W_i) = \beta_j$ if $(t, W_i) \in S_j$, thus, the same observation O_i can contribute to different nodes depending on time t. For the quadratic loss function, $L(X_i, \psi_{I,\beta}) = (Z_i(\cdot) - \psi_{I,\beta}(\cdot, W_i))^\top \Omega(W_i)(Z_i(\cdot) - \psi_{I,\beta}(\cdot, W_i))$ and $\psi_{I,\beta}(\cdot, W_i)$ can be rewritten as

$$\psi_{I,\beta}(\cdot, W_i) = \sum_{j \in I} I((\cdot, W_i) \in S_j)\beta_j = \tilde{W}_i(I)\beta,$$

for an $m \times k$ matrix of indicators, $\tilde{W}_i(I)$, with (t, j)th entry equal to $I((t, W_i) \in S_j)$ and row sums of one. Thus, the $k \times 1$ vector $\hat{\beta}_I$ has the form of a *generalized least squares estimator*

$$\begin{aligned}
\hat{\beta}_I &= \text{argmin}_\beta \sum_{i=1}^{n} (Z_i(\cdot) - \tilde{W}_i(I)\beta)^\top \Omega(W_i)(Z_i(\cdot) - \tilde{W}_i(I)\beta) \frac{\Delta_i}{\bar{G}_n(T_i|X_i)} \\
&= \left(\sum_{i=1}^{n} \frac{\Delta_i}{\bar{G}_n(T_i|X_i)} \tilde{W}_i(I)^\top \Omega(W_i) \tilde{W}_i(I) \right)^{-1} \\
&\qquad \left(\sum_{i=1}^{n} \frac{\Delta_i}{\bar{G}_n(T_i|X_i)} \tilde{W}_i(I)^\top \Omega(W_i) Z_i(\cdot) \right).
\end{aligned}$$

To see this, one can define a stacked $nm \times 1$ outcome vector, $Z = [Z_1(\cdot), \ldots, Z_n(\cdot)]^\top$, a stacked $nm \times k$ design matrix of indicators, $\tilde{W}(I) = [\tilde{W}_1(I), \ldots, \tilde{W}_n(I)]^\top$, and an $nm \times nm$ block diagonal matrix $\Omega(W)$ based on the $\Omega(W_i)$ and the

IPCW weights. The risk criterion then becomes the standard generalized least squares criterion

$$\hat{\beta}_I = \text{argmin}_\beta (Z - \tilde{W}(I)\beta)^\top \Omega(W)(Z - \tilde{W}(I)\beta).$$

Density estimation. For the negative log-likelihood loss function, $L(X_i, \psi_{I,\beta}) = -\log \psi_{I,\beta}(T_i, W_i) = -\log \beta_j$ if $(T_i, W_i) \in S_j$, hence

$$\hat{\beta}_I = \text{argmin}_{\{\beta:\beta_j \geq 0, \sum_j \beta_j = 1\}} - \sum_{j \in I} \sum_{i=1}^{n} I((T_i, W_i) \in S_j) \log \beta_j \frac{\Delta_i}{\bar{G}_n(T_i|X_i)}$$

and

$$\hat{\beta}_{I,j} = \frac{\sum_{i=1}^{n} I((T_i, W_i) \in S_j) \frac{\Delta_i}{\bar{G}_n(T_i|X_i)}}{\sum_{j \in I} \sum_{i=1}^{n} I((T_i, W_i) \in S_j) \frac{\Delta_i}{\bar{G}_n(T_i|X_i)}}, \qquad j \in I.$$

The coefficients $\hat{\beta}_I$ are simply weighted proportions of observations falling in each node S_j, $j \in I$. Details on within-node likelihood calculations for other types of observed data log-likelihood loss functions are given in Breiman (2002, 2003); Davis and Anderson (1989); LeBlanc and Crowley (1992).

2.3.2 Node splitting: minimizing risk over index sets I

In tree-based estimation, the index sets I are obtained by recursive binary partitioning of the covariate space $\mathcal{S}$. Specifically, a new index set I' is obtained from the current I by considering all possible binary splits of each mother node S_j into a left and a right daughter node, $S_{L(j)}$ and $S_{R(j)}$, respectively. The split which results in the maximum decrease in empirical risk yields the new index set I', that is, one seeks I' that maximizes the empirical risk difference between candidates $\hat{\psi}_I = \psi_I(\cdot \mid P_n)$ and $\hat{\psi}_{I'} = \psi_{I'}(\cdot \mid P_n)$

$$\int L(o, \hat{\psi}_I \mid \eta_n) dP_n(o) - \int L(o, \hat{\psi}_{I'} \mid \eta_n) dP_n(o).$$

In the univariate prediction problem with the squared error loss, the risk difference for the split of node S_j into nodes $S_{L(j)}$ and $S_{R(j)}$ simplifies to

$$\begin{aligned} &\int L(o, \hat{\psi}_I \mid \eta_n) dP_n(o) - \int L(o, \hat{\psi}_{I'} \mid \eta_n) dP_n(o) \\ &= \frac{1}{n} \sum_{i=1}^{n} I(W_i \in S_j) \frac{\Delta_i}{\bar{G}_n(T_i|X_i)} (Z_i - \hat{\beta}_{I,j})^2 \end{aligned}$$

$$- \frac{1}{n}\sum_{i=1}^{n} I(W_i \in S_{L(j)}) \frac{\Delta_i}{\bar{G}_n(T_i|X_i)}(Z_i - \hat{\beta}_{I',L(j)})^2$$
$$- \frac{1}{n}\sum_{i=1}^{n} I(W_i \in S_{R(j)}) \frac{\Delta_i}{\bar{G}_n(T_i|X_i)}(Z_i - \hat{\beta}_{I',R(j)})^2,$$

where $\hat{\beta}_I$ and $\hat{\beta}_{I'}$ are weighted node averages as derived in Section 2.3.1. Similarly, for density estimation, the risk difference only depends on observations in the mother node S_j. For multivariate prediction, however, the same observation can contribute to different nodes. Candidate splits of a node S_j into daughter nodes $S_{L(j)}$ and $S_{R(j)}$ are generated based on the values of individual covariates. For example, in the case of an ordered variable W, one considers binary partitions of S_j according to whether or not $W \leq d$, where the cut-offs d are chosen to be halfway between consecutive, distinct values of W in the learning set. For categorical variables, all possible subsets of the categories are considered. Details and extensions (e.g., splits based on linear combinations and Boolean combinations of variables) are discussed in Chapters 4 and 5 of Breiman et al. (1984).

2.4 Cross-validation for estimator selection and performance assessment with censored data

2.4.1 The estimator selection problem

The approaches described in Section 2.3 can be used to construct a sequence of candidate tree estimators, $\hat{\psi}_k = \psi_k(\cdot \mid P_n)$, $k \in \{1, \ldots, K(n)\}$, up to a maximal tree, $\psi_{max} = \psi_{K(n)}$. Here, k indexes the 'size' of the tree, measured by the number of terminal nodes. The size $K(n)$ of the maximal tree is typically determined by criteria such as minimal terminal node size for continuous outcomes or terminal node homogeneity (purity) for categorical outcomes. Our goal is to select a data adaptive $\hat{k} = k(P_n) \in \{1, \ldots, K(n)\}$, such that the risk for $\psi_{\hat{k}}(\cdot|P_n)$ converges to that for the parameter ψ_0 in an optimal manner. In order to address this selection problem, and akin to the uncensored data situation (Dudoit and van der Laan, 2003; Keleş et al., 2003; van der Laan and Dudoit, 2003; van der Laan et al., 2003), define the *conditional risk* of the estimator $\psi_k(\cdot \mid P_n)$ based on the observed data loss function as

$$\tilde{\theta}_n(k) \equiv \int L(o, \psi_k(\cdot \mid P_n) \mid \eta_0) dP_0(o).$$

For the observed data loss functions introduced in Section 2.2.2, the conditional risks in the full and observed data worlds coincide, that is,

$$\begin{aligned}\tilde{\theta}_n(k) &= \int L(o, \psi_k(\cdot \mid P_n) \mid \eta_0) dP_0(o) \\ &= \int IC(o \mid Q_0, G_0, L(\cdot, \psi_k(\cdot|P_n))) dP_0(o) \\ &= \int L(x, \psi_k(\cdot|P_n)) dF_{X,0}(x).\end{aligned}$$

Also define the *optimal risk*, θ_{opt}, as the risk of the parameter of interest,

$$\theta_{opt} = \min_{\psi \in \Psi} \int L(x, \psi) dF_{X,0}(x) = \min_{\psi \in \Psi} \int L(o, \psi \mid \eta_0) dP_0(o).$$

Let

$$\tilde{k}_n \equiv \text{argmin}_k \; \tilde{\theta}_n(k) = \text{argmin}_k \int L(o, \psi_k(\cdot \mid P_n) \mid \eta_0) dP_0(o)$$

be the *optimal benchmark selector* which chooses the estimator with minimal conditional risk $\tilde{\theta}_n(k)$, for *each given dataset*. If the minimum is not unique, then the argmin is defined as the smallest k achieving the minimum. A selector $\hat{k} = k(P_n)$ is said to be *asymptotically equivalent* with the optimal benchmark $\tilde{k}_n$ if

$$\frac{\tilde{\theta}_n(\hat{k}) - \theta_{opt}}{\tilde{\theta}_n(\tilde{k}_n) - \theta_{opt}} \to 1 \text{ in probability.} \tag{4}$$

In particular, then it is *asymptotically optimal*.

Note that the optimal benchmark selector $\tilde{k}_n$ depends on the unknown data generating distribution P_0. The selection problem therefore involves estimating the unknown conditional risk $\tilde{\theta}_n(k)$ for each candidate estimator $\hat{\psi}_k = \psi_k(\cdot|P_n)$. As described next, cross-validation provides a general approach for estimating the conditional risk and producing a data adaptive selector $\hat{k}$ which is asymptotically equivalent to the oracle selector $\tilde{k}_n$ based on the true data generating distribution P_0.

2.4.2 Cross-validation for estimator selection with censored data

The main idea in *cross-validation* (CV) is to divide the available learning set into two sets: a *training set* and a *validation set*. Observations in the training set are used to compute (or *train*) the estimator(s) and the *validation*

set is used to assess the performance of (or *validate*) this estimator(s). The cross-validation estimator $\hat{\psi}_{\hat{k}}$ is chosen to have the best performance on the validation sets.

To derive a general representation for cross-validation, introduce a binary random n-vector, or *split vector*, $S_n \in \{0,1\}^n$, independent of the empirical distribution P_n. A realization of $S_n = (S_{n,1}, \ldots, S_{n,n})$ defines a particular split of the learning sample of n observations into a training set, $\{i \in \{1,\ldots,n\} : S_{n,i} = 0\}$, and a validation set, $\{i \in \{1,\ldots,n\} : S_{n,i} = 1\}$. The empirical distributions of the training and validation sets are denoted by P^0_{n,S_n} and P^1_{n,S_n}, respectively. Let $p = p_n = n_1/n$ be the proportion of observations in the validation set. The particular distribution of the split vector S_n defines the type of cross-validation procedure. As described in van der Laan and Dudoit (2003), this representation covers a broad range of CV procedures, including V-fold, leave-one-out, and Monte Carlo cross-validation.

For a given sequence of candidate estimators, $\hat{\psi}_k = \psi_k(\cdot|P_n)$, indexed by $k \in \{1,\ldots,K(n)\}$, cross-validation risk estimators based on the observed data loss function, $L(o, \psi \mid \eta_0 = (Q_0, G_0)) = IC(o \mid Q_0, G_0, L(\cdot, \psi))$, are given by

$$\begin{aligned}\hat{\theta}_{n(1-p)}(k) &\equiv E_{S_n} \int L(o, \psi_k(\cdot \mid P^0_{n,S_n}) \mid \eta^0_{n,S_n}) dP^1_{n,S_n}(o) \\ &= E_{S_n} \int IC(o \mid Q^0_{n,S_n}, G^0_{n,S_n}, L(\cdot, \psi_k(\cdot|P^0_{n,S_n}))) dP^1_{n,S_n}(o),\end{aligned}$$

where G^0_{n,S_n} and Q^0_{n,S_n} are estimators of G_0 and $Q_0 = Q(F_{X,0})$, respectively, based on the training sample empirical distribution P^0_{n,S_n}. The *cross-validation selector* is defined as the minimizer of the risk estimators from cross-validation

$$\hat{k} = \text{argmin}_k \, \hat{\theta}_{n(1-p)}(k).$$

van der Laan and Dudoit (2003) establish finite sample and asymptotic optimality results for the cross-validation selector $\hat{k}$ for general data generating distributions, loss functions (possibly depending on a nuisance parameter), and estimators. The asymptotic optimality result states that the cross-validation selector $\hat{k}$ performs asymptotically as well as an optimal benchmark selector $\tilde{k} = \text{argmin}_k \int L(o, \hat{\psi}_k \mid \eta_0) dP_0(o)$, based on the unknown data generating distribution P_0. That is,

$$\frac{\int L(o, \hat{\psi}_{\hat{k}} \mid \eta_0) dP_0(o) - \int L(o, \psi_0 \mid \eta_0) dP_0(o)}{\int L(o, \hat{\psi}_{\tilde{k}} \mid \eta_0) dP_0(o) - \int L(o, \psi_0 \mid \eta_0) dP_0(o)}$$

$$= \frac{\int L(x, \hat{\psi}_{\hat{k}}) dF_{X,0}(x) - \int L(x, \psi_0) dF_{X,0}(x)}{\int L(x, \hat{\psi}_{\tilde{k}}) dF_{X,0}(x) - \int L(x, \psi_0) dF_{X,0}(x)} \to 1 \text{ in probability,}$$

provided that, as $n \to \infty$, $p_n \to 0$, $\log(K(n))/np_n$ and $\int (\bar{G}_n - \bar{G}_0)^2 (T \mid X) dF_{X,0}$ both converge to zero faster than the rate at which the estimator $\hat{\psi}_{\tilde{k}}$ converges to the parameter ψ_0 in risk distance, i.e., faster than $\int L(x, \hat{\psi}_{\tilde{k}}) dF_{X,0}(x) - \int L(x, \psi_0) dF_{X,0}(x) \to 0$, where p_n denotes the proportion of observations in the validation sets (see van der Laan and Dudoit (2003) for full statements and proofs of the results).

2.4.3 Cross-validation for estimator performance assessment with censored data

Suppose that an estimator $\hat{\psi} = \psi(\cdot \mid P_n) = \psi_{\hat{k}}(\cdot \mid P_n)$ of the optimal parameter ψ_0 has been selected as described above by cross-validation on a learning set of n observations. The overall performance of this 'final' estimator now needs to be assessed based on an independent *test set*. A *double* or *nested cross-validation* study can be performed, where the learning set is obtained from a partition of the complete dataset of $n^\star$ observations into a learning set and a test set. Let $P_{n^\star}$ denote the empirical distribution of the complete dataset of $n^\star$ observations. The CV risk estimator of the overall performance of the selected estimator is simply

$$E_{S_n^\star} \int L(o, \psi(\cdot \mid P^0_{n^\star, S_n^\star}) \mid \eta^0_{n^\star, S_n^\star}) dP^1_{n^\star, S_n^\star}(o),$$

where $S_n^\star$ refers to binary split vectors for the entire dataset of $n^\star$ observations and $P^0_{n^\star, S_n^\star}$ corresponds to the empirical distribution P_n of a learning set of n observations. Note that the entire estimation procedure (i.e., all three steps in the road map) is now applied to each $P^0_{n^\star, S_n^\star}$. Risk confidence intervals can be obtained as described in Dudoit and van der Laan (2003).

2.4.4 Estimator selection and cross-validation for estimator selection and performance with censored data and CART

In the standard CART methodology, once a maximal tree is grown, a *minimal cost-complexity pruning* algorithm is applied to generate a new sequence of candidate estimators indexed by a complexity parameter α. Specifically, a

cost-complexity measure $R_\alpha(\psi)$ is defined for each candidate tree ψ as

$$R_\alpha(\psi) = \int L(o, \psi \mid \eta_n) dP_n(o) + \alpha|\psi|,$$

where $|\psi|$ denotes the number of terminal nodes in the tree. Minimal cost-complexity pruning is applied to yield a nested decreasing sequence of subtrees $\{\hat{\psi}_\alpha\}$ as candidate estimators and cross-validation is used to select the complexity parameter α which minimizes risk (Chapter 3 in Breiman et al. (1984)). Note that CART's approach for generating candidate estimators can be viewed as forward selection (splitting) all the way to a maximal tree, followed by backward elimination (pruning), where the stopping rule in backward elimination is determined by cross-validation. The unified cross-validation methodology of van der Laan and Dudoit (2003) can be readily applied to extend the CART framework for pruning and performance assessment to multivariate prediction and density estimation with censored data. All that is required is to replace the full data loss function used in CART by one of the observed (censored) data loss functions described in Section 2.2.

3 Simulations and Data Analysis

To evaluate the proposed approach and demonstrate its utility we present the following results. In Section 3.1.1, the asymptotic optimality results briefly described in Section 2.4.2 are illustrated via a simulation study. In Section 3.1.2, our proposed regression tree approach for censored data is compared to that of the default in the R `rpart` function (Ihaka and Gentlemen, 1996; Therneau and Atkinson, 1997) by simulation. Lastly, in Section 3.2, the proposed method is applied to a breast cancer dataset with CGH copy number and survival data.

3.1 Simulation studies

The full data is simulated as follows: $Z \equiv \log T = W^2 + \epsilon$, where $W \sim U(0,1)$, $\epsilon \sim N(0, \sigma^2)$, $\sigma^2 = 0.25$. Thus, $E_0[Z|W] = \text{Median}_0[Z|W] = W^2$ and the conditional survival function is given by $S_0(z \mid W) = Pr_0(Z \geq z \mid W) = 1 - \Phi((z - W^2)/\sigma)$, where $\Phi(\cdot)$ denotes the standard normal cumulative distribution function. Censoring times C are simulated using a mixture of three uniform distributions: $Cens_1 \sim U(\min(Z), cut.dat)$, $Cens_2 \sim$

$U(cut.dat, \max(Z))$, and $Cens_3 \sim U(\max(Z), \max(Z)+2)$, where $\min(Z)$ and $\max(Z)$ refer, respectively, to the minimum and maximum of a random sample of Z's. The mixing proportions are fine-tuned to achieve a desired level of censoring, $Pr_0(\Delta = 0) = Pr_0(C \leq Z)$, and to ensure that $Pr_0(\bar{G}_0(Z|W) > 0.1) = 1$, a condition for the IPCW method.

3.1.1 Asymptotic Optimality Illustration

As referenced in Section 2.4.2, van der Laan and Dudoit (2003) derive an asymptotic optimality result which shows that the CV selector $\hat{k}$ performs asymptotically as well as the optimal benchmark $\tilde{k_n}$, in the sense that the ratio of risk differences in Equation 4 converges to 1 in probability as n goes to infinity. We employ the following simulation to illustrate this result.

Given the model outlined above, the parameter of interest is the conditional expectation, $\psi_0(W) = E_0[Z \mid W]$, corresponding to the quadratic loss function, $L(X, \psi) = (Z - \psi(W))^2$. The candidate estimators as generated by CART are defined as:

$$\psi_k(\cdot|P_n) = \sum_{p=1}^{k+1} I(a_{p-1} \leq X_1 \leq a_p) B_p$$

for a sequence of subtrees where a_p and a_{p-1} define the pth node of the subtree and B_p is the predicted log survival time for the pth node.

The optimal and conditional risks can be analytically written in this simulation. The optimal risk, θ_{opt}, defined as the risk of the parameter of interest, is:

$$\begin{aligned} \theta_{opt} &= E[(Z - \psi_{opt}(W))^2] \\ &= E[(Z - E(Z \mid W])^2] \\ &= E[\epsilon^2] = 0.25 \end{aligned}$$

The conditional risk of the observed data loss functions as described in Section 2.4.1 is :

$$\begin{aligned} \tilde{\theta}_n(k) &= E\left[\left(Z - \sum_{p=1}^{k+1}\{I(a_{p-1} \leq W \leq a_p) B_p\}\right)^2\right] \\ &= E\left[E\left[Z^2 - 2Z\sum_{p=1}^{k+1} I(a_{p-1} \leq W \leq a_p) B_p + (\sum_{p=1}^{k+1} I(a_{p-1} \leq W \leq a_p) B_p)^2 \mid W\right]\right] \end{aligned}$$

$$
\begin{aligned}
&= E[E[Z^2 \mid W] - 2E[Z \mid W] \sum_{p=1}^{k+1} I(a_{p-1} \leq W \leq a_p) B_p \\
&\qquad + (\sum_{p=1}^{k+1} I(a_{p-1} \leq W \leq a_p) B_p)^2] \\
&= E[Z^2] - 2E[W^2 \sum_{p=1}^{k+1} I(a_{p-1} \leq W \leq a_p) B_p] + E[\sum_{p=1}^{k+1} I(a_{p-1} \leq W \leq a_p) B_p]^2 \\
&= E[Z^2] - 2 \sum_{p=1}^{k+1} E[W^2 I(a_{p-1} \leq W \leq a_p)] B_p + \sum_{p=1}^{k+1} P(a_{p-1} \leq W \leq a_p) B_p^2 \\
&= E[Z^2] - 2 \sum_{p=1}^{k+1} \frac{a_p^3 - a_{p-1}^3}{3} B_p + \sum_{p=1}^{k+1} (a_p - a_{p-1}) B_p^2
\end{aligned}
$$

Where $E[Z^2] = E\left[(W^2 + \epsilon)^2\right] = E[W^4] + 0.25 = \int_a^b \frac{W^4}{b-a} dx + 0.25 = \frac{1}{5} + .25 = .45$. As previously defined, the optimal benchmark estimator $\tilde{k_n}$ minimizes the conditional risk $\tilde{\theta}_n(k)$.

For a given v-fold cross-validation, G_0 is estimated for the $IPCW$ estimating equation with a Cox proportional hazard model based on the training sample. Sequences of subtrees are generated by inserting the $IPCW$ weights into the `weight` argument and the `anova` option for the `method` argument in the R `rpart` function. The cross-validation risk estimators, $\hat{\theta}_{n(1-p)}$, are evaluated as described in Section 2.4.2 for each subtree. The data adaptive selector $\hat{k}$ is then chosen as the k which minimizes the cross-validation risk estimators.

Given all the necessary components, the ratio in Equation 4 was evaluated over five different sample sizes (250, 600, 1250, 6000, and 12000), three v-fold cross-validations (5, 10, and 15), and three levels of censoring (0%, 10%, and 20%). For each simulation there were 100 repetitions. Table 1 shows the means of the ratios over the repetitions, while table 2 shows the variances. We observe that as $n \to \infty$ the ratio converges to 1 and the variance of the ratio converges to 0.

3.1.2 Comparison of Approaches

We compare our proposed method to that of LeBlanc and Crowley (1992) which is implemented as a default in the R `rpart` function (Ihaka and Gen-

Table 1: *Simulation study 3.1.1.* Mean of $\frac{\tilde{\theta}_n(\hat{k})-\theta_{opt}}{\tilde{\theta}_n(\tilde{k}_n)-\theta_{opt}}$ over 100 repetitions of five sample sizes (column 1), three v-fold cross-validations (column 2), and three levels of censoring (columns 3-5).

Sample		Censoring		
Size	$v-fold$	0%	10%	20%
250	5	1.125	1.143	1.075
	10	1.124	1.107	1.127
	15	1.11	1.168	1.154
600	5	1.088	1.071	1.076
	10	1.093	1.107	1.116
	15	1.128	1.162	1.076
1250	5	1.064	1.073	1.067
	10	1.046	1.083	1.069
	15	1.092	1.106	1.073
6000	5	1.039	1.042	1.042
	10	1.042	1.046	1.034
	15	1.057	1.059	1.039
12000	5	1.035	1.034	1.029
	10	1.024	1.044	1.033
	15	1.048	1.049	1.039

Table 2: *Simulation study 3.1.1.* Variance of $\frac{\tilde{\theta}_n(\hat{k})-\theta_{opt}}{\tilde{\theta}_n(\tilde{k}_n)-\theta_{opt}}$ over 100 repetitions of five sample sizes (column 1), three v-fold cross-validations (column 2), and three levels of censoring (columns 3-5).

Sample		Censoring		
Size	$v-fold$	0%	10%	20%
250	5	0.056	0.059	0.011
	10	0.054	0.028	0.051
	15	0.025	0.081	0.072
600	5	0.031	0.015	0.019
	10	0.029	0.034	0.053
	15	0.063	0.091	0.01
1250	5	0.029	0.02	0.01
	10	0.006	0.017	0.015
	15	0.025	0.026	0.013
6000	5	0.005	0.006	0.005
	10	0.004	0.005	0.003
	15	0.009	0.009	0.005
12000	5	0.003	0.004	0.002
	10	0.002	0.004	0.003
	15	0.005	0.005	0.003

tlemen, 1996; Therneau and Atkinson, 1997). The main difference between the two methods is in the observed data loss function used for splitting and pruning. Recall that the loss function for the survival trees of LeBlanc and Crowley (1992) is the observed data negative log-likelihood for a Cox proportional hazards model with the same baseline hazard for each node implied by the partition of the covariate space. Risk estimates used in splitting and pruning are based on the first step of a full likelihood estimation procedure. Default `rpart` survival trees are compared to regression trees generated using the observed data IPCW squared error loss function (Section 2.2.3). Denote the two different loss functions by *NLL_PH* and *square_IPCW*, respectively. One can use the `rpart` function to build trees based on the new *square_IPCW* loss function, by setting the `method` argument to the `anova` value and by providing the IPCW weights through the `weights` argument. In the IPCW loss function, the survival function for the censoring mechanism, $\bar{G}_0$, is estimated by fitting a Cox proportional hazards model for each training sample, possibly based on other covariates than those used to build the tree. Tree selection by cross-validation using the new loss function requires minor modifications to `rpart` (see details in Section A).

The use of the two different loss functions *NLL_PH* and *square_IPCW*, along with the standard node splitting and tree pruning rules in `rpart`, leads to two different partitions of the covariate space, i.e., different assignments of observations to terminal nodes. For a given partition, we explore two survival estimation methods: the IPCW mean survival time and the Kaplan-Meier (KM) median survival time for each terminal node. These two types of estimators correspond to full data parameters defined in terms of the squared and absolute error loss functions, i.e., to the conditional mean and median survival time given covariates, respectively.

Learning samples were simulated from an observed data distribution with 20% censoring, for four sample sizes, $n = 250$, 600, 1250, and 6000. One hundred simulated learning samples were generated for each sample size. For both the *NLL_PH* and *square_IPCW* loss functions, trees were selected based on five-fold cross-validation. For the *NLL_PH* loss function, the default $1 - SE$ rule in `rpart` was used. The minimum number of observations in any terminal node was set to the default 7 (`minbucket` argument).

For each of the $B = 100$ simulated learning sets, performance was assessed by generating an independent test sample of size $N = 5000$ from the full data distribution. The estimated survival times were evaluated with the L_2 loss function for the IPCW mean estimation method and the L_1 loss function for

Table 3: *Simulation study 3.1.2.* Comparison of survival trees with observed data negative log-likelihood loss function (*NLL_PH*, `rpart` default) and observed data IPCW squared error loss function (*square_IPCW*). Risk ratios for the *square_IPCW* loss function to the *NLL_PH* loss function are displayed for two types of survival estimation methods and for four sample sizes, n. Individual entries of the table are ratios of $\sum_{b=1}^{B} \int L(x, \psi(\cdot \mid P_n^b)) dP_N^b(x)$, where P_n^b and P_N^b denote, respectively, the learning sample and test sample empirical distributions in the bth simulation, $N = 5000$, $B = 100$. For the KM estimation method (column 2), L is the absolute error loss, and for the IPCW mean method (column 3), L is the squared error loss.

Sample	Survival estimation method	
size, n	KM Median	IPCW Mean
250	0.9422	0.8838
600	0.9524	0.9062
1250	0.9629	0.9244
6000	0.9767	0.9533

the Kaplan-Meier median estimation method. Within each sample size, the test sample risk estimates were averaged over the $B = 100$ repetitions; ratios of the risk for the *square_IPCW* loss function to the risk for the *NLL_PH* loss function are displayed in Table 3 for the two types of estimation methods. Ratios less than one correspond to improved accuracy for trees based on the new IPCW loss function.

The results illustrate the impact on accuracy of the choice of loss function used for splitting, pruning, and within-node estimation. As expected, when the parameter of interest is the conditional mean survival, the risk is smaller for partitions generated by the *square_IPCW* loss function ("IPCW Mean" column). This loss function also corresponds to lower risk when interest is in estimating the median survival. The difference in risk decreases with increasing sample size.

3.2 Breast cancer survival and CGH copy number dataset

Our censored regression tree method was also applied to a dataset from a Comparative Genomic Hybridization (CGH) study of breast cancer patients.

Data were collected on 152 patients, all with initial occurrences of breast cancer, 52 who subsequently recurred. Time to event (in months) was defined as time to recurrence or, if no recurrence, time to final follow-up or time of death (with no disease observed). Patients with no recurrence are censored. According to these definitions, the censoring percentage is 66%. The explanatory variables include, histopathologic variables (e.g., tumor stage, grade), epidemiological variables (e.g., age at diagnosis, race), and DNA copy number measures from a CGH array with 2,254 bacterial artificial chromosomes (BAC). Details on CGH and on the particular dataset are described in a forthcoming manuscript by members of the UCSF Comprehensive Cancer Center (Waldman et al., in preparation).

The 152 observations were split at random into a learning set and a test set of 135 and 27 (i.e., five sixths and one sixth) observations, respectively, retaining the appropriate level of censoring. Trees were grown using the learning sample and their overall performance assessed on the test sample. Five-fold cross-validation of the learning sample was used to select the 'best' tree (again, retaining the appropriate level of censoring). The censoring mechanism survival function, $\bar{G}_0$, used in the IPCW loss function was estimated separately for each of the five training samples in the cross-validation, by fitting a Cox proportional hazards model to the histological and epidemiological variables. Maximal exploratory trees were grown using the R `rpart` function with the `weight` argument set to the appropriate IPCW estimate and with `cp=0` (Therneau and Atkinson, 1997). The estimate for $\bar{G}_0$ from the training sample was maintained in the IPCW loss function used on the validation sample to evaluate the candidate estimators, or subtrees. The risk for each subtree was averaged over the five validation samples. For the *square_IPCW* loss function, the minimum risk was achieved with only one split. A subsequent tree was grown with the entire learning sample and the predictor was assessed by the independent test sample. The possible numbers of splits for this tree were 0, 2, 3, or 4, with corresponding test sample IPCW mean squared error loss 2.530699, 2.349634, 2.535852, 2.701512, respectively. Since cost complexity pruning does not always return a sequence of trees corresponding to all possible numbers of splits, one would need to choose between zero and two splits. The full tree is shown in Figure 1, with filled circles for the two split subtree. Each terminal node is described by the IPCW mean log survival time (in months) and the number of observations. The legend in the bottom left corner indicates the chromosomal location of each BAC. The first two splits are based on BACs that fall in chromosomal

regions known to contain genes related to breast cancer (personal communication with Joe Gray and Fred Waldman). The tree suggests that copy number gains in both regions are associated with longer survival. The default `rpart` method, based on the *NLL_PH* loss function, selected only the root node; the maximal tree was based on different variables (BACs) than those for the *square_IPCW* trees.

4 Discussion

We have described an application of our unified loss-based estimation methodology for censored data to tree-structured estimators. The approach encompasses univariate prediction, multivariate prediction, and density estimation, simply by defining a suitable loss function for each of these problems. Censored data are handled by mapping the full, uncensored data loss function into an observed, censored data loss function having the same expected value (van der Laan and Robins, 2002). This approach reconciles censored and full data estimation methods, and, in particular, the standard full data estimators are recovered as special cases of the censored data estimators. In addition, the IPCW and DR-IPCW loss functions allow for informative censoring and can be used for any type of prediction method, including standard linear regression, logic regression, and bagging and boosting procedures. Previously proposed survival trees, such as those of Davis and Anderson (1989), LeBlanc and Crowley (1992), and Breiman et al. (1984), correspond to different choices for the observed data log-likelihood loss function.

The simulation study of Section 3 illustrates that the choice of loss function used for splitting and pruning can have a significant impact on accuracy. It also shows that gains in accuracy can be obtained by using a loss function that is specific to the parameter of interest. Preliminary analysis of a breast cancer survival and CGH dataset using trees built with the IPCW squared error loss function identified two BACs implicated in breast cancer. Improved prediction accuracy and more information on chromosomal regions related to breast cancer survival may be obtained from aggregation methods such as bagging and boosting. We are also exploring more aggressive procedures for generating candidate estimators (see below), that include "OR" in addition to "AND" statements and are more specific to CGH data (Molinaro et al., in preparation).

Tree-structured estimators correspond to one particular type of sieve for

the candidates in Step 2 of the road map, analogous in some sense to forward selection (splitting) followed by backward elimination (pruning). Current problems in genomics (e.g., DNA microarray and SNP data) involve the analysis of high-dimensional datasets with complex interactions among variables. In this setting, it is particularly important to perform an efficient search of the parameter space to generate a good sequence of candidate estimators. van der Laan and Dudoit (2003) and Sinisi and van der Laan (in preparation) discuss more general sieves and more aggressive search strategies based on addition/deletion/substitution algorithms capable of revealing high-order interactions among variables.

Section 2.2 alluded to the fact that a given parameter of interest, ψ_0, can arise as the risk minimizer for a number of different loss functions, say $L_1, \ldots, L_m$ (e.g., different loss functions for classification trees in Section 2.2.3; different choices of quadratic loss function for multivariate prediction in Section 2.2.4; different models for the log-likelihood loss function in density estimation in Section 2.2.5). Natural questions then include: choosing suitable full data loss functions for generating candidate estimators and for overall performance assessment and, given a particular choice of loss function, obtaining an efficient estimator of the corresponding risk. While the later question was discussed in Sections 2.2.2 and 2.4, the former deserves further study. Although risk is minimized by the same parameter ψ_0 for each L_j, i.e., $\psi_0 = \text{argmin}_{\psi \in \Psi} \int L_j(x, \psi) dP_0(x)\ \forall j$, different choices for the loss function lead to estimators of ψ_0 with different properties. In particular, minimizing the empirical risk for a loss function L_1 could yield an estimator with lower risk for a second loss function L_2, than the empirical risk minimizer for L_2, i.e., one can have $\int L_2(x, \hat{\psi}_1) dP_0(x) \leq \int L_2(x, \hat{\psi}_2) dP_0(x)$, where $\hat{\psi}_j = \text{argmin}_{\psi \in \Psi} \int L_j(x, \psi) dP_n(x)$, $j = 1, 2$ (cf. generalized least squares estimation). In other words, it may be advantageous to use a different loss function for generating candidate estimators and for overall performance assessment. One could employ a collection of loss functions, $L_1, \ldots, L_m$, to generate candidate estimators and then use cross-validation to select among these candidates using the loss function L^* of interest for overall performance assessment. We are further investigating the loss function selection issue. Other ongoing efforts include deriving loss-based measures of variable importance and the development of software implementing the new methodology.

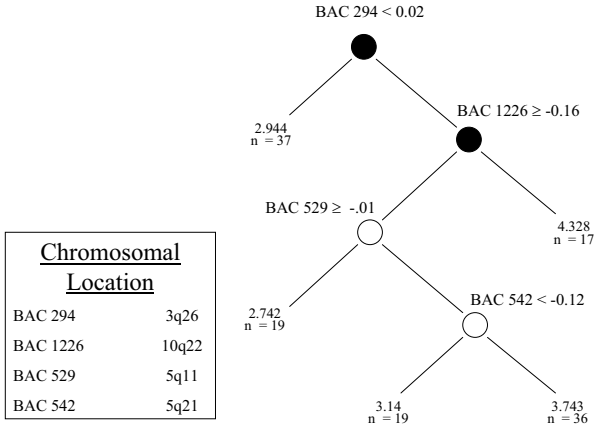

Figure 1: *Breast cancer survival and CGH copy number dataset.* Survival tree built from the learning sample of 135 patients, using the `square_IPCW` loss function. Each terminal node is described by the IPCW mean log survival time (in months) and the number of observations.

Acknowledgment

We would like to thank Joe Gray, Dan Moore, and Fred Waldman, from the Comprehensive Cancer Center at the University of California, San Francisco, for graciously providing the CGH dataset, biological insight, and fruitful discussions. We are also grateful to Terry Therneau and Elizabeth Atkinson for their thorough explanation of the R `rpart` package.

A Program Code

In this Appendix we present an example of code written for R to perform a data analysis using our proposed regression tree approach for censored data. This section contains the program code while the following section, B, displays the necessary functions called within the program code. The example dataset for this code contains four covariates, $x1, \cdots, x4$, a minimum

of the survival and censoring times, *ttilde*, and an indicator of event, *delta*.

Due to the limitation of choosing the best model for $\bar{G}$ for each training sample the program code is written to be evaluated interactively. The `for` loop must then be incremented by hand for each of the v repetitions in the v-fold cross validation. This code will soon be available as a function on the author's webpage.

```
options(contrasts=c("contr.treatment","contr.poly"))
library(survival)
library(rpart)


# Data

# ttilde is the minimum of the survival and censoring time
# delta is the indicator that ttilde is equal to the survival time
# x1 - x4 are the covariates

data <- cbind(ttilde, delta, x1, x2, x3, x4)
deltac <- 1 - delta    # Indicator of censoring
w1 <- cbind(x1,x2,x3,c4)

#################################################################
# Set up cross validation while retaining censoring percentage #
#################################################################

# Choose level of v-fold cross validation
v <- 5
get.sam <- rep(c(1:v),sum(delta==1)/v)
if(length(get.sam)<sum(delta==1)){
   get.sam <- c(get.sam,c(1:(sum(delta==1)-length(get.sam))))
 }
delta.1 <- sample(get.sam,sum(delta==1),replace=F)
delta.0 <- sample(rep(c(1:v),sum(delta==0)/v),sum(delta==0),replace=F)

k <- 1
l <- 1
grp.delt <- NULL
for(m in 1:nrow(data)){
  if(delta[m]==0){
    grp.delt[m] <- delta.0[k]
    k <- k+1
  }
  if(delta[m]==1){
```

```
    grp.delt[m] <- delta.1[l]
    l <- l+1
  }
}

## Cross Validation - by hand to choose an appropriate model for \bar{G}
for(i in 1:v){
  sel.risk.ts <- NULL

  ##########################################################################
  # Estimate \bar{G} & generate candidate estimators with training sample#
  ##########################################################################
  tr.set.w <- (w1[grp.delt!=i,])   # Training sample covariates
  tr <- grp.delt!=i                # Indicator of training sample


  # Chose covariates to estimate \bar{G}

  coxph(Surv(ttilde[tr], deltac[tr]) ~ x1[tr])
  coxph(Surv(ttilde[tr], deltac[tr]) ~ x2[tr])
  coxph(Surv(ttilde[tr], deltac[tr]) ~ x3[tr])
  coxph(Surv(ttilde[tr], deltac[tr]) ~ x4[tr])

  num.of.covariates <- 2  # number of covariates to retain for \bar{G}
                          # model
                          # For example, keep x1 and x2 to predict \bar{G}

  # set up contrast matrix for \bar{G}
  cov <- matrix(0,nrow=nrow(tr.set.w),ncol=num.of.covariates)
  cov[,1] <- ifelse(x1[tr]==1,1,0)
  cov[,2] <- ifelse(x2[tr]==9,1,0)
  cov <- as.matrix(cov)

  # estimate \bar{G} and build weights
  surv.cox <- coxph(Surv(ttilde[tr], deltac[tr]) ~ cov)
        # Get coefficients from cox model
          coeff.cox <- surv.cox$coefficients
        # Get baseline survival
          surv.cens <- survfit(surv.cox, newdata = data.frame(cov = 0),
                       type = "kaplan-meier")
        # Vector of baseline survival
          basesurv <- get.at.surv.times(surv.cens, ttilde[tr])
        # Weights for each subject
          ipcw.wt<- get.init.am(basesurv,coeff.cox,cov,delta[tr],ttilde[tr])
```

```
# Generate candidate estimators with rpart and IPCW weights
cpval <- 0.000
r.x.cens <- rpart(log(ttilde[tr])~x1[tr] + x2[tr] + x3[tr] + x4[tr],
            method=alist,cp = cpval, xval=0,weight= ipcw.wt)

# Number of subtrees (candidate estimators)
inde.c <-  r.x.cens$cptable[,"CP"]

################################################
# Get validation sample for estimator selection #
################################################

ts.set.w <- w1[grp.delt==i,]  # Covariates for validation sample
ts <- grp.delt==i             # Indicator of validation sample

# Get contrast matrix for \bar{G} for validation sample using model
#  chosen by training sample

cov.ts <- matrix(0,nrow=nrow(ts.set.w),ncol=num.of.covariates)
cov.ts[,1] <- ifelse(x1[ts]==1,1,0)
cov.ts[,2] <- ifelse(x2[ts]==9,1,0)
cov.ts <- as.matrix(cov.ts)

# estimate \bar{G} and build weights
surv.cox <- coxph(Surv(ttilde[ts], deltac[ts]) ~ cov.ts)
    # Get coefficients from cox model
      coeff.cox <- surv.cox$coefficients
    # Get baseline survival
      surv.cens <- survfit(surv.cox, newdata = data.frame(cov.ts = 0),
                   type = "kaplan-meier")
    # Vector of baseline survival
      basesurv <- get.at.surv.times(surv.cens, ttilde[ts])
    # Weights for each subject
      ipcw.wt.ts <- get.init.am(basesurv,coeff.cox,cov.ts,delta[ts],ttilde[ts])

# Calculate the cross-validation risk estimate for each subtree
for(k in (1:length(inde.c))){
  get.k <- inde.c[k]
  new.cB <- prune(r.x.cens,
            cp=(mean(c(inde.c[k],ifelse(k==1,inde.c[k]+1,inde.c[k-1])))))
  sel.risk.ts[k] <- get.emp.risk(rpart.tr=new.cB,ts.n=nrow(ts.set.w),
                    a1.ts=ts.set.w,c.dat.time=log(ttilde[ts]),c.wt=ipcw.wt.ts,
                    num.surrogates=10)
}
```

```
  # Put results in a matrix which has rows which correspond with the
  # number of subtrees
  for(l in 1:length(r.x.cens$cptable[,"nsplit"])){
    right.row <- r.x.cens$cptable[l,"nsplit"]+1
    keep.sel.risk[right.row,i] <- sel.risk.ts[l]
  }

  for(m in 3:nrow(keep.true.risk)){
    if(is.na(keep.sel.risk[(m-1),i])) keep.sel.risk[(m-1),i] <- keep.sel.risk[(m-2),i]
  }
}

# Choose the data adaptive k which minimizes the cross validation
# risk estimates
 sr.mean <- apply(keep.sel.risk[1:10,],1,mean,na.rm=TRUE)

 if(length(c(1:length(sr.mean))[sr.mean==min(sr.mean)])==1){
     ind.tr <- c(1:length(sr.mean))[sr.mean==min(sr.mean)]
  }
 else {ind.tr <- min(c(1:length(sr.mean))[sr.mean==min(sr.mean)])}


########################################
### Now do test set                  ###
########################################


i <- 6

sel.risk.ts <- NULL
set.w <- (w1[grp.delt!=i,])
tr <- grp.delt !=i

# Chose covariates to estimate \bar{G}

  coxph(Surv(ttilde[tr], deltac[tr]) ~ x1[tr])
  coxph(Surv(ttilde[tr], deltac[tr]) ~ x2[tr])
  coxph(Surv(ttilde[tr], deltac[tr]) ~ x3[tr])
  coxph(Surv(ttilde[tr], deltac[tr]) ~ x4[tr])

  num.of.covariates <- 2  # number of covariates to retain for \bar{G}
                          # model
                          # For example, keep x1 and x2 to predict \bar{G}
```

```
# set up contrast matrix for \bar{G}
cov <- matrix(0,nrow=nrow(set.w),ncol=num.of.covariates)
cov[,1] <- ifelse(x1[tr]==1,1,0)
cov[,2] <- ifelse(x2[tr]==9,1,0)
cov <- as.matrix(cov)

# estimate \bar{G} and build weights
surv.cox <- coxph(Surv(ttilde[tr], deltac[tr]) ~ cov)
      # Get coefficients from cox model
        coeff.cox <- surv.cox$coefficients
      # Get baseline survival
        surv.cens <- survfit(surv.cox, newdata = data.frame(cov = 0),
                     type = "kaplan-meier")
      # Vector of baseline survival
        basesurv <- get.at.surv.times(surv.cens, ttilde[tr])
      # Weights for each subject
        ipcw.wt<- get.init.am(basesurv,coeff.cox,cov,delta[tr],ttilde[tr])


# Generate candidate estimators with rpart and IPCW weights
cpval <- 0.000
r.x.cens <- rpart(log(ttilde[tr])~x1[tr] + x2[tr] + x3[tr] + x4[tr],
            method=alist,cp = cpval, xval=0,weight= ipcw.wt)

inde.c <-  r.x.cens$cptable[,"CP"]  # test samples

set.w <- w1[grp.delt==i,]
ts <- grp.delt==i

# Get contrast matrix for \bar{G} for validation sample using model
#  chosen by training sample

cov.ts <- matrix(0,nrow=nrow(ts.set.w),ncol=num.of.covariates)
cov.ts[,1] <- ifelse(x1[ts]==1,1,0)
cov.ts[,2] <- ifelse(x2[ts]==9,1,0)
cov.ts <- as.matrix(cov.ts)

# estimate \bar{G} and build weights
surv.cox <- coxph(Surv(ttilde[ts], deltac[ts]) ~ cov.ts)
    # Get coefficients from cox model
      coeff.cox <- surv.cox$coefficients
    # Get baseline survival
      surv.cens <- survfit(surv.cox, newdata = data.frame(cov.ts = 0),
                   type = "kaplan-meier")
    # Vector of baseline survival
```

```
      basesurv <- get.at.surv.times(surv.cens, ttilde[ts])
    # Weights for each subject
      ipcw.wt.ts <- get.init.am(basesurv,coeff.cox,cov.ts,delta[ts],ttilde[ts])

  sel.risk.ts <- NULL
  for(k in (1:length(inde.c))){
     get.k <- inde.c[k]
     new.cB <- prune(r.x.cens,cp = (mean(c(inde.c[k],
               ifelse(k==1,inde.c[k]+1,inde.c[k-1])))))
     sel.risk.ts[k]<- get.emp.risk(new.cB,nrow(set.w),set.w,
                      log(ttilde.ds[ts]),cox.wt1.ts,num.surrogates=10)
    }

   sel.risk.ts
```

B Functions

B.1 Estimate $\bar{G}$

```
#######################################################################
## Functions to estimate G_0 with Cox Proportional Hazards models  ##
#######################################################################

get.at.surv.times <- function(surv.cens, times)
{
# This function returns the baseline survival
#
# surv.cens is an object created by survfit
# times is a vector of times for which you want an estimate of the
#       survival function
#
  nt <- length(times)
  outs <- rep(0, nt)
  survv <- summary(surv.cens)$surv
  ns <- length(survv)
  timev <- summary(surv.cens)$time
  for(i in 1:nt) {
     if(times[i] < timev[1]) {
        outs[i] <- 1
      }
     else if(times[i] >= timev[ns]) {
               outs[i] <- survv[ns]
```

```
          }
          else {
               outs[i] <- survv[timev == max(timev[timev <= times[i]])][1]
          }
     }
     no <- length(outs[outs == 0])
     outs[outs == 0] <- rep(survv[ns - 1], no)
     return(outs)
}
```

B.2 Evaluate IPCW weights

```
get.init.am <- function(surv.cens, coeff.cox, w, delta, ttilde)
{
# This function returns the IPCW weights
#
# surv.cens is an object created by survfit
# coeff.cox are the coefficients evaluated by coxph
# w are the covariates
# delta is the indicator of event
# ttilde is the minimum of the censoring and survival times
#

  w <- ifelse(is.na(w),0,w)
  if(!is.matrix(w)){
     w <- matrix(w,nrow=length(w),ncol=1)
   }
  nn <- length(ttilde)
  coeff.cox <- matrix(coeff.cox,nrow=length(coeff.cox),ncol=1)
  coeff.cox[is.na(coeff.cox)] <- 0
  linpred <- w %*%coeff.cox
  sum.surv.cond <- surv.cens^(exp(linpred))
  sum.surv.cond <- ifelse(sum.surv.cond<.2,.2,sum.surv.cond)
  if(is.na(min(sum.surv.cond))){
     if(delta[is.na(sum.surv.cond)]==0){
     sum.surv.cond[is.na(sum.surv.cond)] <- 1
   }
        }
  B <- (delta)/sum.surv.cond
  return(B)
}
```

B.3 Evaluate cross-validation risk estimates

```
get.emp.risk <- function(rpart.tr,ts.n,a1.ts,c.dat.time,c.wt,num.surrogates=5) {

if (nrow(rpart.tr$frame)==1){
    pred.y <- rpart.tr$frame[,"yval"]
    node.dest <- cbind(pred.y,c.dat.time,c.wt,((c.dat.time-pred.y)^2)*c.wt)
    dimnames(node.dest)[[2]] <- c("pred.y" ,  "real.y"    ,"IPCWwt",   "L2" )
    return(sum(node.dest[,"L2"])/length(node.dest[,"L2"]))
}
else{
  # Get tree that rpart built
  # Get var split on, n, yval, index=value var split on, which direction = ncat
    sp <- rpart.tr$splits[,c("index","ncat","adj")]
    fr <- rpart.tr$frame[,c("var","n","yval")]
    tog <- cbind(fr,NA,NA)
    num.splits <- nrow(tog[fr[,"var"]!="<leaf>",])
    if(num.surrogates>0){
        pick.splits <- c(1:nrow(sp))[sp[,"adj"]==0]
        pick.splits <- pick.splits[!is.na(pick.splits)]
    }
    if(num.surrogates==0) pick.splits <- num.splits
    tog[fr[,"var"]!="<leaf>",4:5] <- sp[pick.splits,c("index","ncat")]
    dimnames(tog)[[2]][4:5] <- c("index","ncat")
    sp <- cbind(var=rownames(sp),sp)

    if(num.surrogates>0){
        surs <- list()
        ind.i <- 1
        sub.t <- 4
        sub.t2 <- 0
        for(i in c(c(1:nrow(tog))[tog[,"var"]!="<leaf>"])){
            surs[[i]] <- list()
            if(ind.i == length(c(c(1:nrow(tog))[tog[,"var"]!="<leaf>"]))
               && (ind.i*(num.surrogates+1)-sub.t-1)== nrow(sp)){
            surs[[i]] <- NA}
            else{
                if(sp[(ind.i*(num.surrogates+1)-sub.t),"adj"] != "0"){
                    for(spn in 1:num.surrogates){
                        surs[[i]][[spn]] <- list()
                        surs[[i]][[spn]][[1]] <- sp[(ind.i*(num.surrogates+1)
                                                  -sub.t)  : (ind.i*(num.surrogates+1)
                                                  -sub.t2),"var"][spn]
                        surs[[i]][[spn]][[2]] <- as.numeric(sp[(ind.i*(num.surrogates
                                                  +1)-sub.t)  : (ind.i*(num.surrogates+1)
                                                  -sub.t2),"index"][spn])
                        surs[[i]][[spn]][[3]] <- as.numeric(sp[(ind.i*(num.surrogates+1)
                                                  -sub.t)  : (ind.i*(num.surrogates+1)
                                                  -sub.t2),"ncat"][spn])
                    }
                }
                else if(sp[(ind.i*(num.surrogates+1)-sub.t),"adj"] == "0"){
                    sub.t <- sub.t + 5
                    sub.t2 <- sub.t2 + 5
                }
            }
            ind.i <- ind.i+1
```

```
    } # end for loop
}

# Build list to emulate rparts tree
gi <- list()

for (i in 1:nrow(tog)){
    gi[[i]] <- list()
    gi[[i]][[1]] <- as.numeric(rownames(tog[i,]))
}

ind <- c(1:ifelse(is.matrix(a1.ts),nrow(a1.ts),length(a1.ts)))
nind <- c(1:nrow(tog))
for(j in 1:nrow(tog)){
# First track the branches that each node follows
  track <- NULL
  first <- gi[[j]][[1]]
  while (first!= 1){
   track <- c(first,track)
   if((first%%2)==0)  first <- first/2
   else first <- (first-1)/2
  }
  gi[[j]][[2]] <- c(1,track)
  if(j==1) {
      gi[[j]][[3]] <- ind
      gi[[j]][[4]] <- tog[j,"yval"]
      gi[[j]][[5]] <- "root"
      gi[[j]][[6]] <-  tog[j,"var"]
      gi[[j]][[7]] <- tog[j,"index"]
      gi[[j]][[8]] <- tog[j,"ncat"]
      gi[[j]][[9]] <- length(gi[[j]][[3]])
  }
  else {  # if j!=1
      last.node <- nind[rownames(tog)==(paste(gi[[j]][[2]]
                   [length(gi[[j]][[2]])-1]))]
      if( gi[[last.node]][[9]] > 1){
          if( (gi[[last.node]][[8]]==1 && gi[[j]][[1]]%%2==0)
             || (gi[[last.node]][[8]]== -1 && gi[[j]][[1]]%%2==1))
          {
              gi[[j]][[3]] <- gi[[last.node]][[3]][a1.ts[gi[[last.node]][[3]],
                              (as.numeric(gi[[last.node]][[6]])-1)]>=
                              gi[[last.node]][[7]]]
              if(sum(is.na(a1.ts[gi[[last.node]][[3]],
                            (as.numeric(gi[[last.node]][[6]])-1)]))>0){

                  xy <- a1.ts[gi[[last.node]][[3]],
                       (as.numeric(gi[[last.node]][[6]])-1)]
                       [is.na(a1.ts[gi[[last.node]][[3]],(as.numeric(gi[[last.node]][[6]])-1)])]
                  send.maj.keep <- 0
                  send.check <- NULL
                  for(xy.1 in 1:length(xy)){
                      test.na <- 1
                      send.maj <- 0
                      if(is.na(surs[[last.node]]) || is.null(as.pairlist(surs[[last.node]]))){
                          send.maj <- 1
                          send.maj.keep <- 1
                          send.check <- c(send.check,xy.1)
```

```
            }
            else {
                while(is.na(a1.ts[names(xy)[xy.1],
                      substr(surs[[last.node]][[test.na]][[1]],7,
                      nchar(surs[[last.node]][[test.na]][[1]]))]))
                {
                    test.na <- test.na+1
                    if(test.na > num.surrogates) {
                        send.maj <- 1
                        send.maj.keep <- 1
                        send.check <- c(send.check,xy.1)
                        break()
                    }
                }
            }
            if (!send.maj)
            {
                if( ((surs[[last.node]][[test.na]][[3]]==1 &&
                      gi[[j]][[1]]%%2==0) || (surs[[last.node]][[test.na]][[3]]== -1
                      && gi[[j]][[1]]%%2==1)) && (  a1.ts[names(xy)[xy.1],
                      substr(surs[[last.node]][[test.na]][[1]],7,
                      nchar(surs[[last.node]][[test.na]][[1]]))] >=
                      surs[[last.node]][[test.na]][[2]])) {
                    gi[[j]][[3]] <- c(gi[[j]][[3]],c(1:nrow(a1.ts))
                                    [rownames(a1.ts)==names(xy)[xy.1]])
                }
                else if(((surs[[last.node]][[test.na]][[3]]==1 &&
                        gi[[j]][[1]]%%2==1) || (surs[[last.node]]
                        [[test.na]][[3]]== -1 && gi[[j]][[1]]%%2==0))&&
                        (a1.ts[names(xy)[xy.1],substr(surs[[last.node]]
                        [[test.na]][[1]],7,nchar(surs[[last.node]][[test.na]]
                        [[1]]))] < surs[[last.node]][[test.na]][[2]])  )
                {
                    gi[[j]][[3]] <- c(gi[[j]][[3]],c(1:nrow(a1.ts))
                                    [rownames(a1.ts)==names(xy)[xy.1]])
                }
            }
        }
        if(send.maj.keep){
            if(length(gi[[j]][[3]][!is.na(gi[[j]][[3]])])/ gi[[last.node]][[9]] > .5) {
                for(jk in 1:length(send.check)){
                    gi[[j]][[3]] <- c(gi[[j]][[3]],c(1:nrow(a1.ts))
                                    [rownames(a1.ts)==names(xy)[send.check[jk]]])
                }
            }
        }
        gi[[j]][[3]] <- gi[[j]][[3]][!is.na(gi[[j]][[3]])]
    } # end if there are surrogates
}
else if((gi[[last.node]][[8]]==1 && gi[[j]][[1]]%%2==1) ||
        (gi[[last.node]][[8]]== -1 && gi[[j]][[1]]%%2==0)  )
{
    gi[[j]][[3]] <- gi[[last.node]][[3]][a1.ts[gi[[last.node]][[3]],
                    (as.numeric(gi[[last.node]][[6]])-1)] < gi[[last.node]][[7]]]
    if(sum(is.na(a1.ts[gi[[last.node]][[3]],(as.numeric(gi[[last.node]][[6]])-1)]))>0){
        xy <- a1.ts[gi[[last.node]][[3]],(as.numeric(gi[[last.node]][[6]])
              -1)][is.na(a1.ts[gi[[last.node]][[3]],(as.numeric(gi[[last.node]][[6]])-1)])]
```

```
            send.maj.keep <- 0
            send.check <- NULL
            for(xy.1 in 1:length(xy)){ # fill in missing with surrogate splits
                test.na <- 1
                send.maj <- 0
                if(is.na(surs[[last.node]])|| is.null(as.pairlist(surs[[last.node]]))){
                    send.maj <- 1
                    send.maj.keep <- 1
                    send.check <- c(send.check,xy.1)
                }
                else {
                    while(is.na(a1.ts[names(xy)[xy.1],substr(surs[[last.node]][[test.na]][[1]],
                          7,nchar(surs[[last.node]][[test.na]][[1]]))])){
                        test.na <- test.na+1 # make sure there is a non missing value in surrogate
                        if(test.na > num.surrogates) {
                            send.maj <- 1
                            send.maj.keep <- 1
                            send.check <- c(send.check,xy.1)
                            break()
                        }
                    }
                }
                if (!send.maj){
                    if(((surs[[last.node]][[test.na]][[3]]==1 &&
                          gi[[j]][[1]]%%2==0) || (surs[[last.node]][[test.na]][[3]]
                          == -1 && gi[[j]][[1]]%%2==1)) && (  a1.ts[names(xy)[xy.1]
                          ,substr(surs[[last.node]][[test.na]][[1]],7,
                          nchar(surs[[last.node]][[test.na]][[1]]))] >=
                          surs[[last.node]][[test.na]][[2]])){
                        gi[[j]][[3]] <- c(gi[[j]][[3]],c(1:nrow(a1.ts))
                                          [rownames(a1.ts)==names(xy)[xy.1]])
                    }
                    else if(((surs[[last.node]][[test.na]][[3]]==1 &&
                            gi[[j]][[1]]%%2==1) || (surs[[last.node]][[test.na]][[3]]==
                            -1 && gi[[j]][[1]]%%2==0)) && (a1.ts[names(xy)[xy.1]
                            ,substr(surs[[last.node]][[test.na]][[1]],7,
                            nchar(surs[[last.node]][[test.na]][[1]]))] <
                            surs[[last.node]][[test.na]][[2]])  )  {
                        gi[[j]][[3]] <- c(gi[[j]][[3]],c(1:nrow(a1.ts))
                                          [rownames(a1.ts)==names(xy)[xy.1]])
                    }
                }
            }
            if(send.maj.keep){
                if(length(gi[[j]][[3]][!is.na(gi[[j]][[3]])])/ gi[[last.node]][[9]] > .5) {
                    for(jk in 1:length(send.check)){
                        gi[[j]][[3]] <- c(gi[[j]][[3]],c(1:nrow(a1.ts))
                                    [rownames(a1.ts)==names(xy)[send.check[jk]]])
                    }
                }
            }
            gi[[j]][[3]] <- gi[[j]][[3]][!is.na(gi[[j]][[3]])]
        } # end if there are surrogates
    } # end else
} # end if gi[[last.node]][[9]] > 1  - so don't carry an observation beyond it's respective node
gi[[j]][[4]] <- tog[j,"yval"]
gi[[j]][[5]] <- ifelse(as.numeric(tog[paste(gi[[j]][[1]]),"var"])==1,"LEAF", "INT NODE")
```

```
      if (gi[[j]][[5]] != "LEAF")
      {
          gi[[j]][[6]] <-  tog[j,"var"]
          gi[[j]][[7]] <- tog[j,"index"]
          gi[[j]][[8]] <- tog[j,"ncat"]
      }
      gi[[j]][[9]] <- length(gi[[j]][[3]])
  } # end else j != 1
} # end j for loop

    elem <- NULL
    term <- NULL
    pred.y <- NULL
    node <- NULL
    for(ii in 2:nrow(tog)){
        if(gi[[ii]][[5]]=="LEAF") {
            gi[[ii]][[10]] <- ii
            elem <- c(elem,gi[[ii]][[3]])
            term <- c(term,rep(ii,length(gi[[ii]][[3]])))
            node <- c(node,rep(gi[[ii]][[1]],length(gi[[ii]][[3]])))
            pred.y <- c(pred.y,rep(gi[[ii]][[4]],length(gi[[ii]][[3]])))
        }
    }
    node.dest <- cbind(elem,term,node,pred.y,c.dat.time[elem],((c.dat.time[elem]-pred.y)^2)*c.wt[elem])
    dimnames(node.dest)[[2]] <- c("elem" ,  "term" ,  "node"  , "pred.y", "real.y"    ,   "L2")

    return(sum(node.dest[,"L2"])/length(node.dest[,"L2"]))
} # end else
}
```

B.4 User Defined Functions for `rpart`

The functions listed here are provided by Beth Atkinson and Terry Therneau in *http://cran.r-project.org/src/contrib/PACKAGES.html#rpart* with one small adaptation. If a node were ever chosen to only represent censored observations the weight would be 0 which would result in an error when calculating the predicted value for the node. To prevent this we have added the `if/else` in `temp1`. Now, given the hypothetical setting, the predicted value for the node is the maximum censored time over all observations in the node. Notwithstanding the change in `temp1`, this code emulates method `anova` in `rpart`.

```
temp1 <- function(y, wt, parms) {
    if(sum(wt)==0){
        wmean <- max(y)
        rss <- sum((y-wmean)^2)
    }
    else{
        wmean <- sum(y*wt)/sum(wt)
        rss <- sum(wt*((y-wmean)^2))
    }
```

```
    list(label= wmean, deviance=rss)
  }

temp2 <- function(y, wt, x, parms, continuous) {
    # Center y
    n <- length(y)
    y <- y- sum(y*wt)/sum(wt)

    if (continuous) {
        # continuous x variable
        temp <- cumsum(y*wt)[-n]

        left.wt  <- cumsum(wt)[-n]
        right.wt <- sum(wt) - left.wt
        lmean <- temp/left.wt
        rmean <- -temp/right.wt
        goodness <- (left.wt*lmean^2 + right.wt*rmean^2)/sum(wt*y^2)
        list(goodness= goodness, direction=sign(lmean))
        }
    else {
        # Categorical X variable
        ux <- sort(unique(x))
        wtsum <- tapply(wt, x, sum)
        ysum  <- tapply(y*wt, x, sum)
        means <- ysum/wtsum

        # For anova splits, we can order the categories by their means
        #  then use the same code as for a non-categorical
        ord <- order(means)
        n <- length(ord)
        temp <- cumsum(ysum[ord])[-n]
        left.wt  <- cumsum(wtsum[ord])[-n]
        right.wt <- sum(wt) - left.wt
        lmean <- temp/left.wt
        rmean <- -temp/right.wt
        list(goodness= (left.wt*lmean^2 + right.wt*rmean^2)/sum(wt*y^2),
             direction = ux[ord])
        }
    }



temp3 <- function(y, offset, parms, wt) {
    if (!is.null(offset)) y <- y-offset
    list(y=y, parms=0, numresp=1, numy=1,
```

```
             summary= function(yval, dev, wt, ylevel, digits ) {
                 paste("  mean=", format(signif(yval, digits)),
                       ", MSE=" , format(signif(dev/wt, digits)),
                       sep='')
             },
         text= function(yval, dev, wt, ylevel, digits, n, use.n ) {
             if(use.n) {paste(formatg(yval,digits),"\nn=", n,sep="")}
             else{paste(formatg(yval,digits))}
           })
    }


alist <- list(eval=temp1, split=temp2, init=temp3)
```

A. M. was supported by a grant from the Institute for Scientific Computing Research at Lawrence Livermore National Laboratory. S.D. is correspondence author.

References

L. Breiman. Software for the masses. In *Wald Lectures*. Meeting of the Institute of Mathematical Statistics, Banff, Canada, 2002. URL `www.stat.berkeley.edu/~breiman`.

L. Breiman. *How to use survival forests*. Department of Statistics, UC Berkeley, 2003. URL `www.stat.berkeley.edu/~breiman`.

L. Breiman and J. H. Friedman. Predicting multivariate responses in multiple linear regression. *Journal of the Royal Statistical Society, Series B*, 59(1): 3–54, 1997.

L. Breiman, J. H. Friedman, R.A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth and Brooks/Cole, Monterey, CA, 1984.

A. Ciampi, J. Thiffault, J. P. Nakache, and B. Asselain. Stratification by stepwise regression, correspondence analysis and recursive partition. *Computational Statistics and Data Analysis*, 4:185–204, 1986.

R. Davis and J. Anderson. Exponential survival trees. *Statistics in Medicine*, 8:947–961, 1989.

S. Dudoit and M. J. van der Laan. Asymptotics of cross-validated risk estimation in model selection and performance assessment. Technical Report 126, Division of Biostatistics, University of California, Berkeley, 2003. URL `www.bepress.com/ucbbiostat/paper126/`.

R. D. Gill, M. J. van der Laan, and J. R. Robins. Coarsening at random: Characterizations, conjectures and counter-examples. In D. Y. Lin and T. R. Fleming, editors, *Proceedings of the First Seattle Symposium in Biostatistics, 1995*, Springer Lecture Notes in Statistics, pages 255–294, 1997.

L. Gordon and R. Olshen. Tree-structured survival analysis. *Cancer Treatment Reports*, 69:1062–1069, 1985.

E. Graf, C. Schmoor, W. Sauerbrei, and M. Schumacher. Assessment and comparison of prognostic classification schemes for survival data. *Statistics in Medicine*, 18:2529–2545, 1999.

T. Hothorn, B. Lausen, A. Benner, and M. Radespiel-Troger. Bagging survival trees. Technical report, Depratment of Medical Informatics, Biometry and Epidemiology, Friedrich-Alexander-University Erlangen-Nuremberg, 2002.

R. Ihaka and R. Gentlemen. R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, 5:299–314, 1996.

S. Keleş, M. J. van der Laan, and S. Dudoit. Asymptotically optimal model selection method for regression on censored outcomes. Technical Report 124, Division of Biostatistics, University of California, Berkeley, 2003. URL `www.bepress.com/ucbbiostat/paper124/`.

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

M. Morgan and J. A. Sonquist. Problems in the analysis of survery data and a proposal. *Journal of the American Statistical Association*, 58:415–434, 1963.

J. Robins and A. Rotnitzky. *Recovery of information and adjustment for dependent censoring using surrogate markers*, chapter AIDS Epidemiology, Methodological issues. Bikhauser, 1992.

M. Segal. Regression trees for censored data. *Biometrics*, 44:35–48, 1988.

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

M. J. van der Laan and S. Dudoit. Unified cross-validation methods for selection among estimators: Finite sample results, asymptotic optimality, and applications. Technical Report 130, Division of Biostatistics, University of California, Berkeley, 2003. URL `www.bepress.com/ucbbiostat/paper130/`.

M. J. van der Laan, S. Dudoit, and S. Keleş. Asymptotic optimality of likelihood based cross-validation. Technical Report 125, Division of Biostatistics, University of California, Berkeley, 2003. URL `www.bepress.com/ucbbiostat/paper125/`.

M. J. van der Laan and J. Robins. *Unified Methods for Censored Longitudinal Data and Causality*. Springer, 2002.