Electronic Journal of Applied Statistical Analysis
EJASA, Electron. J. App. Stat. Anal.
http://siba-ese.unisalento.it/index.php/ejasa/index
e-ISSN: 2070-5948
DOI: DOI–10.1285/i20705948v15n3p479

Survival trees: a pathway among features and open issues of the main *R* packages
By Macis

Published: November 20, 2022

Electronic Journal of Applied Statistical Analysis
Vol. 15, Issue 03, November 2022, 479-509
DOI: DOI–10.1285/i20705948v15n3p479

Survival trees: a pathway among features and open issues of the main *R* packages

Ambra Macis*

University of Brescia, Department of Economics and Management
C.da Santa Chiara 50, 25122 Brescia, Italy

Published: November 20, 2022

Survival analysis aims to study the occurrence of a particular event during a follow-up period. Recently, many machine learning methods have been used for analyzing right-censored data. Among these, survival trees are a useful tool of recursive partitioning for defining homogeneous groups in terms of survival probability. However, there are still some unclear points on how to work with these methods from a practical point of view. Indeed, even if there are a lot of proposed methods, many of these present little documentation, mainly concerning the corresponding *R* functions. Moreover, there does not exist an harmonization of all these proposals. This work aims to shed light on the topic and to provide a practical guide for simulating survival data, fitting survival trees and evaluating their performance with the statistical software *R*.

keywords: Survival Data, Recursive Partitioning, Machine Learning, Simulations.

1 Introduction

Survival analysis has been developed since the XVII century with the aim of studying the occurrence of a particular event (endpoint) during a given observed period of time (follow-up).
The particular feature of survival data is *censoring*. Censoring occurs when the endpoint of interest has not been observed for a certain subject under study during the observation

*Corresponding author: a.macis@unibs.it

ISSN: 2070-5948
http://siba-ese.unisalento.it/index.php/ejasa/index

time; so, the only known thing about a censored subject is the last time he did not experience the event (Collett, 2015).
Survival analysis is widely used in medicine for studying events as death or recurrence of symptoms. However, it can also be used for many other settings as franchising research (Perrigot et al., 2004), e.g. for predicting survival of a store, and criminology studies, e.g. for predicting time until recidivism (Wang et al., 2019).

During the years many approaches have been proposed for analyzing survival data. In particular, both the two cultures of statistical modeling (Breiman, 2001), i.e. statistical and machine learning methods, have been used. Statistical models can be divided into nonparametric, semiparametric and parametric. The former are the simplest ones and are usually used to estimate the survival function of a group of subjects without the need to use specific assumptions about the underlying distribution of the survival times. The most used nonparametric methods are the Kaplan-Meier (Kaplan and Paul, 1958) and the Nelson-Aalen (Nelson, 1969, 1972; Aalen, 1978) estimators. Among semiparametric methods the most famous is the Cox proportional hazards (PH) regression model (Cox, 1972), which allows to estimate the hazard function on the basis of a set of explanatory variables without any assumption on the distribution of survival times. Finally, parametric models are used to estimate the survival or hazard function on the basis of a particular distributional assumption on survival times, e.g. Exponential, Weibull and Gompertz distributions (Collett, 2015). Among machine learning methods several approaches have been used; examples include survival trees, random survival forests, neural networks and support vector machines (Wang et al., 2019).

This work focuses on survival trees, i.e. tree-based algorithms for censored data. During the years many proposals have been advanced; many of these are extensions of Classification and Regression Trees (CART) (Breiman et al., 1984) to survival data. Survival trees were firstly developed by Gordon and Olshen in 1985 (Gordon and Olshen, 1985); following, many other suggestions have been introduced (Ciampi et al., 1986, 1987; Segal, 1988; Davis and Anderson, 1989; LeBlanc and Crowley, 1992, 1993; Molinaro et al., 2004), usually based on modifications of the splitting criterion and the pruning algorithm.

In the following years other algorithms that modified the CART one have been proposed. Among these there are conditional inference trees (Hothorn et al., 2006; Kundu and Ghosh, 2021), the partitioning Deletion-Substitution-Addition algorithm (Lostritto et al., 2012) and ROC-guided survival trees (Sun et al., 2020a).
Extensions of survival trees for interval-censored data and for left-truncated and right censored data have also been provided (Fu and Simonoff, 2017a,b). Furthermore, other recursive partitioning algorithms, different from CART, can be found in literature. Examples are model-based recursive partitioning (MOB) (Zeileis et al., 2008) and Bayesian survival trees (Clarke and West, 2008). Finally, algorithms for discrete survival times have also been proposed (Bou-hamad et al., 2009). For a structured review about survival trees please see Bou-Hamad et al., 2011.

Despite up to now many proposals have been advanced, it is still unclear which algorithm performs better. Simulation studies are a powerful tool for fully understanding and evaluating the behavior of statistical methods (Crowther and Lambert, 2012); there-

fore, a key step in this setting is to perform simulations of survival data. Moreover, also from a practical point of view there are some unclear points related to model fitting and performance evaluation that need to be solved.

This work aims to provide a practical guide for simulating right-censored data, fitting survival trees and evaluating their performance with the statistical software *R* (R Core Team, 2021). In particular, interest has been focused on issues that can be met when (i) generating censored data; (ii) fitting survival trees, with particular interest on relative risk trees (LeBlanc and Crowley, 1992) and conditional inference trees (Hothorn et al., 2006; Kundu and Ghosh, 2021); and (iii) evaluating models' performance.

The article is organized as follows. In the next section the methodological framework is presented. Then, Section 3 shows how to simulate right-censored data, how to fit survival trees and how to evaluate their performance in *R*, with particular attention to problems and possible solutions involved in these steps. Following, an example is provided in Section 4. The paper ends with the final discussion.

2 Methodological framework

2.1 Survival data

In survival analysis, the i^{th} subject is represented by a triplet $(\boldsymbol{x}_i, \tau_i, \delta_i)$ in which (i) $\boldsymbol{x}_i$ is the vector of observed covariates; (ii) τ_i is the observed time (survival time t_i for an uncensored subject or censoring time c_i for a censored one) and (iii) δ_i is a binary event indicator that assumes value 1 if the subject is uncensored and 0 otherwise:

$$\tau_i = min(t_i, c_i) = \begin{cases} t_i & \text{if } \delta_i = 1 \\ c_i & \text{if } \delta_i = 0 \end{cases} .$$

2.2 Relative Risk Trees

Relative risk trees (RRTs) have been introduced by LeBlanc and Crowley in 1992 (LeBlanc and Crowley, 1992) as an extension of CART. This recursive partitioning algorithm is based on the Cox PH model (Cox, 1972).

In particular, it exploits the connection between the proportional hazards full likelihood and the Poisson model likelihood, allowing to estimate survival trees by fitting Poisson trees.

The algorithm splits the covariate space into regions that maximize the reduction in one-step deviance realized by the split. The binary splitting continues until a large binary tree is grown and then the tree is pruned through the cost-complexity pruning algorithm of CART (Breiman et al., 1984).

The resulting node summary is the quantity in (1), which can be interpreted as the ratio between observed and expected number of events in the node under the assumption of no structure in survival times, providing an estimate of relative risk between different nodes (LeBlanc and Crowley, 1992):

$$\theta_k = \frac{\sum_{i \in S_k} \delta_i}{\sum_{i \in S_k} \hat{H_0}^1(t_i)} \,, \tag{1}$$

where S_K is the index set of k^{th} node, δ_i is the event indicator (equal to 1 for events and 0 for censoring) and $\hat{H_0}^1$ is an estimate of the cumulative baseline hazard (see LeBlanc and Crowley, 1992).

2.3 Conditional Inference Trees - `ctree` Algorithm

Conditional inference trees have been firstly proposed by Hothorn et al. in 2006 (Hothorn et al., 2006) to overcome the issue of selection bias of classical CART. Their proposal, the `ctree` algorithm, is based on the theory of permutation tests developed by Strasser and Weber (Strasser and Weber, 1999) and it can be applied to all kinds of outcomes, including censored data.

Differently from CART, conditional inference trees based on the `ctree` algorithm (CITs) are based on two different steps for variable and split point selection. In the first step, the global null hypothesis of independence between any of the covariates and the outcome variable is tested and the covariate with the strongest association to the response (usually the one with the lowest p-value) is chosen as splitting variable. Once a splitting variable is selected, the best split point is chosen by a splitting criterion, e.g. the log-rank test (Peto and Peto, 1972). Finally, the recursive partitioning algorithm is stopped when the null hypothesis of independence cannot be rejected.
At each terminal node the estimated Kaplan-Meier (K-M) survival curve (Kaplan and Paul, 1958) and median survival time are provided.

Hothorn et al. showed that CITs, beyond overcoming the selection bias issue, have a predictive performance equivalent to that of optimally pruned CART; thus, they represent a computationally efficient and intuitive solution also to the overfitting problem (Hothorn et al., 2006).

2.4 Conditional Inference Trees - `SurvCART` Algorithm

Very recently another proposal for growing survival trees has been advanced by Kundu and Ghosh in 2021 (Kundu and Ghosh, 2021). The proposed algorithm, `SurvCART`, stands in the same conditional inference framework of `ctree`. The main differences between the two algorithms are two. The first one is that while CITs use a nonparametric permutation test, conditional inference trees based on `SurvCART` (SCTs) assume a particular distribution for survival times and incorporate it in the parameter instability tests. The second difference is that SCTs allow to take into account also censoring heterogeneity. Thus, they represent a useful tool when censoring mechanism is dependent on baseline covariates, inducing a condition of *conditionally independent censoring* (also known as *dependent censoring*), so that censoring is independent of the time-to-event distribution only conditional on the set of covariates.

In detail, the parameter instability test is performed for each variable to test heterogeneity on both the time-to-event and censoring distribution. The most significant

variable (if exists) is selected and then the best cut-off value is chosen through the maximization of a dissimilarity measure, usually the log-rank statistic (Peto and Peto, 1972). In particular, if the censoring distribution was found more heterogeneous than the time-to event distribution for the chosen partitioning variable, the log-rank statistic is computed assuming censoring as an event; otherwise the log-rank is evaluated as usual. The procedure is then repeated until the parameter instability tests fail to reject the null hypothesis of homogeneity. Once the tree has been grown, the median survival time is estimated at each terminal node.

2.5 Performance Evaluation

Performance evaluation of survival methods can be carried out through many kinds of measures. The most used (Rahman et al., 2017) are:

- *discrimination measures*, to assess the ability of the model to distinguish between low and high risk subjects;
- *overall performance measures*, which quantify both discrimination and calibration (agreement between the observed and the predicted outcomes), evaluating the distance between observed and predicted outcome.

Among the discrimination measures the Concordance index (C-index), the time-dependent Area Under Curve (AUC) and the Somer's delta are noteworthy. Among the most used overall performance measures there are the Brier Score (BS) and its integrated version, the Integrated Brier Score (IBS). Finally, a scaled version of BS, the Index of Prediction Accuracy (IPA) has been introduced recently.

Concordance Index. Concordance measures are the most used indices for evaluating the discrimination ability of a model. C-index is a rank order statistic based on the ratio of all the concordant pairs to the total comparable pairs (all possible combinations). The most used C-index is that proposed by Harrell et al. in 1982 (Harrell et al., 1982). This index deals with censored survival times only when censoring occurs later than an event. Thus, survival times of two subjects can be compared only if (i) both the observations are uncensored or if (ii) the observed event time of the uncensored observation is smaller than the censoring time of the censored one.

Assuming to have a comparable pair of subjects (i, j), with t_i and t_j the actual observed times and $\hat{S}(t_i)$ and $\hat{S}(t_j)$ the respective estimated survival times, the pair is concordant if $t_i > t_j$ and $\hat{S}(t_i) > \hat{S}(t_j)$; otherwise, the pair is discordant. Similarly, if the model outcome is the hazard function (e.g. Cox PH model), a pair is concordant if $t_i > t_j$ and $\hat{h}(t_i) < \hat{h}(t_j)$ and discordant otherwise, with $\hat{h}(t_i)$ and $\hat{h}(t_j)$ the estimated hazards for the i^{th} and j^{th} subject respectively.

The Harrell's C-index uses as weight the number of comparable pairs at each time point. It can then be evaluated as:

$$\hat{C}_H = \frac{1}{C} \sum_{t:\delta_i=1} \sum_{j:t_i<t_j} I[\hat{S}(t_i) < \hat{S}(t_j)] = \frac{\sum_{i \neq j} \delta_i I(T_i < T_j) I(\hat{S}(T_i) < \hat{S}(T_j))}{\sum_{i \neq j} \delta_i I(T_i < T_j)} \quad (2)$$

or, equivalently, as:

$$\hat{C}_H = \frac{1}{C} \sum_{t:\delta_i=1} \sum_{j:t_i<t_j} I[\hat{h}(t_i) > \hat{h}(t_j)] = \frac{\sum_{i \neq j} \delta_i I(T_i < T_j) I(\hat{h}(T_i) > \hat{h}(T_j))}{\sum_{i \neq j} \delta_i I(T_i < T_j)}, \quad (3)$$

where C is the overall number of comparable pairs, $I[\cdot]$ is the indicator function, δ_i is the indicator of event, T_i is the observed time for the i^{th} subject and $\hat{S}(T_i)$ and $\hat{h}(T_i)$ are respectively the estimated survival and hazard function for the i^{th} subject.

A modification to this index has been proposed by Uno et al. (Uno et al., 2011) for considering also the pairs where the censored observed time is shorter than the event time. To this extent the proposed index is based on censoring probability weights; in particular, the used weight is C/G^2, with C being the number of comparable pairs and G the censoring distribution (Therneau and Atkinson, 2020). The index can then be defined as:

$$\hat{C}_U = \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} \delta_i \hat{G}(T_i)^{-2} I(T_i < T_j, T_i < \kappa) I(\hat{h}(T_i) > \hat{h}(T_j))}{\sum_{i=1}^{n} \sum_{j=1}^{n} \delta_i \hat{G}(T_i)^{-2} I(T_i < T_j, T_i < \kappa)} \quad (4)$$

where $\hat{G}(\cdot)$ is the Kaplan-Meier estimator for the censoring distribution, $G(t) = Pr(D > t)$ (D censoring variable) and κ is a pre-specified time point such that $Pr(D > \kappa) > 0$.

A measure of concordance can then be evaluated at each time point using a time-truncated version of the C-index (Gerds et al., 2013). The idea is to consider as events only the subjects for which the event occurred before the time point of interest and truncating the time-interval to that time point. So, all the subjects who experienced the event after the specific time point are considered censored. Thus, the truncated C-index measures the ability of the model to rank the event times that occurred before the time point of interest (Gerds et al., 2013).

The C-index (both Harrell's and Uno's version) ranges between 0 and 1, with 0 and 1 indicating that all the pairs are discordant or concordant respectively. A value of 0.5 indicates that the model's performance is equivalent to that of a flip coin.

Time-Dependent ROC Curves and Related Area Under the Curve. Time-dependent AUC is another discrimination measure that is obtained evaluating the area under the ROC curve at any time point. Therefore, AUC values can be provided as a function of time. The main differences with classical ROC analysis are that (i) the outcome of an observation can change over time, and (ii) the presence of censoring (Park et al., 2021).

The AUC indicates the model's performance for discriminating the binary outcome (event/no event) at a given time point; values closer to 1 indicate better performance.

For evaluating the model's performance during an interval of time the integrated AUC can be obtained considering the integration of multiple time-dependent AUC values across the time period of interest.

Somer's delta. Somer's delta (d) is an alternative measure of concordance that considers not only concordant and discordant pairs, but also tied pairs for the predicted outcome (i.e. pairs that have the same predicted survival or hazard) (Therneau and Atkinson, 2020):

$$d = \frac{N_C - N_D}{N_C + N_D + T_P} \, ,$$

where N_C and N_D are respectively the number of concordant and discordant pairs and T_P is the number tied pairs for the predicted outcome. This index ranges between -1 and 1 and it is related to the Harrell's C-index as follows:

$$C_H = \frac{d+1}{2} \, . \tag{5}$$

Brier Score and Integrated Brier Score. The BS is an index firstly introduced by Brier et al. in 1950 (Brier et al., 1950) and then extended to censored data by Graf et al. (Graf et al., 1999). This index quantifies the distance between observed ($I(T_i > t^*)$) and predicted outcomes ($\hat{S}(t^*|\boldsymbol{X}_i)$) through a quadratic loss function. For compensating loss of information due to censored data each individual contribution is then weighted through the inverse of the respective censoring distribution.

For a fixed time point t^*, the individual contributions to BS can be distinguished into three subgroups:

1. uncensored observations for which the event occurred before t^*, i.e. subjects for which $\delta_i = 1$ and $\tau_i = min(T_i, C_i) = T_i \leq t^*$;

2. all observations for which the event/censoring time occurred after t^*, i.e. subjects for which $\tau_i = min(T_i, C_i) \geq t^* \; \forall \; \delta_i$;

3. censored observations for which censoring occurred before t^*, i.e. subjects for which $\delta_i = 0$ and $\tau_i = min(T_i, C_i) = C_i \leq t^*$.

For subjects in the first group $I(T_i > t^*) = 0$, thus the contribution to the BS is $(0 - \hat{S}(t^*|\boldsymbol{X}_i))^2$; in the second group the contribution is instead $(1 - \hat{S}(t^*|\boldsymbol{X}_i))^2$. For the third group censoring occurred before t^* so that their event status at t^* is unknown and its contribution to BS cannot be evaluated. Then, the BS can be evaluated as follows:

$$\begin{aligned} BS(t^*) = \frac{1}{n} \sum_{i=1}^{n} \frac{I(\tau_i \leq t^*, \delta_i = 1)(I(T_i > t^*) - \hat{S}(t^*|\boldsymbol{X}_i))^2}{\hat{G}(\tau_i)} + \\ + \frac{I(\tau_i > t^*)(I(T_i > t^*) - \hat{S}(t^*|\boldsymbol{X}_i))^2}{\hat{G}(t^*)} = \\ = \frac{1}{n} \sum_{i=1}^{n} \frac{I(\tau_i \leq t^*, \delta_i = 1)(0 - \hat{S}(t^*|\boldsymbol{X}_i))^2}{\hat{G}(\tau_i)} + \frac{I(\tau_i > t^*)(1 - \hat{S}(t^*|\boldsymbol{X}_i))^2)}{\hat{G}(t^*)} \end{aligned}$$

Since the BS quantifies the distance between the observed and predicted outcome, lower values indicate better performance. The BS can be evaluated at a single time point t^*. If interest is on a given period of time it is possible to integrate the BS over the time period of interest with respect to a weight function $w(t)$, obtaining the IBS:

$$IBS(t^*) = \int_0^{t^*} BS(t)dw(t) \ .$$

Natural choices of $w(t)$ are t/t^* or $(1 - \hat{S}(t))/(1 - \hat{S}(t^*))$ (Graf et al., 1999).

Index of Prediction Accuracy. More recently a modification of BS has been proposed to have a more interpretable index, the IPA (Kattan and Gerds, 2018). The index is obtained rescaling the BS with the null model, i.e. the model without predictors (estimated, in a no competing risk setting, with the K-M estimator), used as benchmark value:

$$IPA = 1 - \frac{\text{model BS}}{\text{null model BS}} \ .$$

This formulation allows an easier interpretation of BS and also permits to distinguish useful models from harmful and useless ones. An IPA value of 100% indicates a perfect model, while a negative value indicates a useless or harmful model (Kattan and Gerds, 2018).

3 Performing Simulations in *R* with Survival Data

The main steps in a simulation study are three: (i) data simulation; (ii) model fitting and (iii) performance assessment. There are many R packages that allow to carry out these three steps; however, there are many unclear points on how to perform these steps practically. In the next sections a presentation of many of the available packages, together with possible related issues and solutions, is reported.

3.1 Data simulation

There are many packages that allow to simulate survival data in R. Among these there are `simsurv` (Brilleman et al., 2021), `survsim` (Moriña et al., 2014), `coxed` (Kropko and Jeffrey, 2019) and `rocTree` (Sun et al., 2020b) (see Table 1).

The `simsurv` package allows to simulate survival data from standard parametric distributions (Weibull, Exponential and Gompertz), two-component mixture distributions or a user-defined hazard function. Under the proportional hazards assumption it allows to consider a set of baseline (time-independent) covariates (randomly generated in a separate step). Furthermore, it can also take into account time-dependent covariates. With this function, observations with an event time greater than the duration of follow-up are defined as censored; so, only right-censored data are admitted.

Differently, the `survsim` package allows to simulate both event and censoring times from a given set of standard distributions, i.e. Weibull, Log-Logistic and Log-Normal.

Its particular feature is that, beyond simple survival data, it also allows to simulate data with multiple and recurrent events. It also permits to generate baseline covariates (possible distributions are Uniform, Normal and Bernoulli). However, it does not allow to directly generate time-varying covariates.

The `coxed` package allows to simulate survival data from Cox PH regression model; in addition it can also accept a user-supplied hazard function. It manages both time-independent and time-dependent covariates (that can be generated separately). The advantage is that the desired proportion of censoring can be fixed. Interestingly, it also allows to consider right-censoring conditional on the covariates.

Finally, the `rocTree` package allows to simulate survival data from the hazard function through different pre-specified scenarios, including the PH model and the Accelerated Failure Time model (for more details see Sun et al., 2020b). Similarly to the other packages, it also allows to consider time-dependent covariates. Finally, it permits to specify a specific percentage of censoring (0%, 25% and 50% values are admitted).

Table 1: Packages and functions in *R* for simulating right-censored data

R package (function)	Features
`simsurv` (`simsurv`)	Parametric distributions (Weibull, Exponential, Gompertz) Two-component mixture dstributions User-defined hazard/log-hazard function Time-independent & time-dependent covariates
`survsim` (`simple.surv.sim`) (`mult.ev.sim`) (`rec.ev.sim`)	Parametric distributions (Weibull, Log-normal, Log-logistic) Simple Survival data Multiple events Recurrent events
`coxed` (`sim.survdata`)	Cox PH model User-supplied hazard function Time-independent & time-dependent covariates Specific percentage of censoring
`rocTree` (`simu`)	Hazard function modeled by different scenarios 0%, 25% or 50% of censoring

After having presented some of the available *R* packages for simulating survival data, the main steps to which pay attention when simulating survival data are shown below.

First step. Choice of the Data Generating Process. The first step when simulat-

ing survival data is the choice of the correct Data Generating Process (DGP) depending on the model of interest. Indeed, each model requires a particular specification of data. If, for example, the focus is on the parametric PH model (for theoretical details see Collett (2015)), a vector of coefficients associated to the explanatory variables has to be defined together with the distribution of survival times. On the contrary, if interest is in a recursive partitioning algorithm, the partitions induced by the theoretical model should be defined. A way to do this is to define different parameters for the distribution of related survival times in each partition (an example is provided in Section 4).

Second step. Choice of the covariates. Once the DGP has been chosen, the covariates to be entered in the model can be generated. In performing this step attention has to be paid to the covariates features. For example, when the sample size is small there is the risk that variables with too low variance (e.g. Bernoulli distribution with parameter π equal to 0.1 or 0.9) lead to issues in model estimation and to singularities that would not permit to perform the next steps (as model fitting or performance assessment). This because it could happen, for example, to observe samples in which only 1 or 0 values are observed in correspondence of the event.

Then, when simulating data from a survival regression model (e.g. the parametric PH model), another related aspect to consider is the choice of covariates coefficients. In a survival regression PH model the β_j coefficient represents the estimated change in the logarithm of the hazard ratio as the corresponding covariate X_j is increased by one unit (if the variable is continuous), or the estimated logarithm of the relative hazard for an individual in group j relative to an individual in the reference group (if X_j is categorical). Usually, the exponential of β, representing an hazard ratio, is considered due to the easier interpretation. It is better that the β coefficients do not assume too high values because otherwise the effect size measured through hazard ratios would be too high and the risk of overestimating the model performance increases in both the training and test set.

This step is not necessary when dealing with recursive partitioning algorithms because covariates coefficients have not to be specified in the survival data simulation phase. Indeed covariates are only involved in the definition of partitions and in model fitting.

Once these two aspects have been considered, covariates can be generated.

Third step. Choice of the distribution of survival times. The key element for right-censored data simulation, common for both the two DGPs, is the choice of the distribution of survival times and correspondent parameters. In particular, it is important to choose appropriate parameter values. For example, considering the Exponential distribution, the λ parameter defines the theoretical hazard of the simulated data. When choosing a value for this parameter, it is necessary to take into account that a too low value would lead to very long survival times, while a too high one would lead to a great reduction of event-free subjects during the follow-up. Consequently, depending on the value of λ chosen for simulating survival data, a suitable value for the length of the follow-up must be defined taking into account also the desired amount of censoring in the simulated dataset. If λ is low a longer follow-up is required for observing more events

than censored observations; similarly, if λ is high, a shorter observation time is needed (see Table 2).

Table 2: Mean percentage of censoring for different values of follow-up length (*maxtime*) and λ. Values obtained through bootstrap from 1000 simulated datasets of size N=1000 for each combination of values

	λ						
maxtime	0.5	1.0	1.5	2.0	2.5	3.0	3.5
0.5	77.871%	60.626%	47.177%	36.705%	28.578%	22.263%	17.348%
1.0	60.626%	36.705%	22.263%	13.540%	8.224%	4.985%	3.033%
1.5	47.177%	22.263%	10.544%	4.985%	2.364%	1.111%	0.526%
2.0	36.705%	13.540%	4.985%	1.844%	0.674%	0.249%	0.087%
2.5	28.578%	8.224%	2.364%	0.674%	0.190%	0.050%	0.015%
3.0	22.263%	4.985%	1.111%	0.249%	0.050%	0.011%	0.003%
3.5	17.348%	3.033%	0.526%	0.087%	0.015%	0.003%	0.001%

Moreover, an example is shown in Figure 1. The two curves represent the survival function for two groups: the dashed line represents the survival probability of a sample with survival time $T \sim Exp(0.4)$ and with a follow-up duration of 2.5 years; the solid line, instead, represents survival for a group of observations with $T \sim Exp(3)$ and with a follow-up length of 3.5 years. It can be seen that in the first case there are many censored subjects (survival probability is equal to about 40% at the end of follow-up); on the contrary, in the second group all observations experienced the event before the end of follow-up.

When dealing with a parametric PH model only one distribution is required to be assumed for the entire simulated sample. On the contrary, when dealing with tree-based algorithms, as many distributions as the number of theoretical partitions must be set. Of course, parameters close to each other will lead to a lower difference of effect sizes in the data, with less separate and identifiable partitions, due to approximately equal survival curves. So, it is important to test and distinguish situations in which the effect sizes are close each other and those in which the effect sizes are more distant to better evaluate model performance. An example is provided in Figure 2. The two panels represent two different settings. The left graph shows three well distinguished survival curves, obtained assuming exponentially distributed survival times with λ equal to 2, 0.9 and 1 respectively. Differently, the right graph shows three survival curves obtained assuming exponentially distributed survival times with λ equal to 0.8, 0.7 and 0.5 respectively, which are approximately equal each other. The tables reported below each figure present the results obtained from a simulation study. In particular,

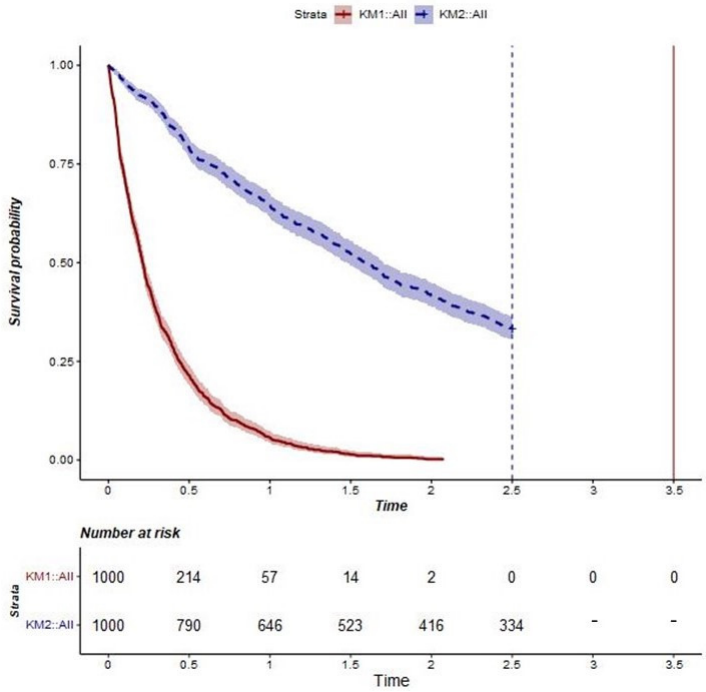


Figure 1: Example of K-M survival curves for two different settings. The dashed vertical line indicates the fixed duration of follow-up (*maxtime*) for the dashed survival curve ($T \sim Exp(\lambda = 0.4)$, $maxtime = 2.5$). Similarly the solid vertical line indicates the fixed *maxtime* for the solid survival curve ($T \sim Exp(\lambda = 3)$, $maxtime$=3.5)

the tables report, for different sample sizes, the percentage of trees (grown using the three tree algorithms) that identified the right theoretical partition. It can be seen that when the effect sizes are distant and survival curves are well distinguished survival trees perform well. Differently, when the effect sizes are close each other, the fitted survival trees do not identify very often the right partition.

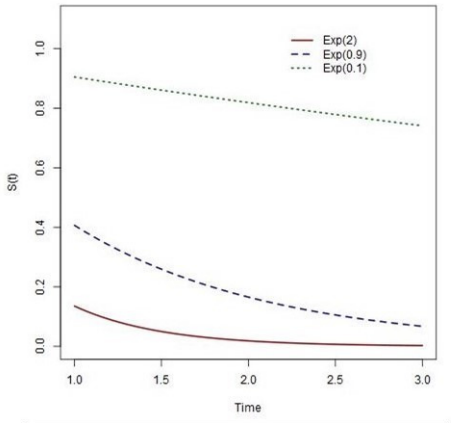
After having taken into account all these aspects, data can be simulated and then the model can be fitted.

3.2 Model fitting

If the model of interest is a parametric PH model, the `flexsurv` package can be used. In particular, the `flexsurvreg` function allows parametric modeling or regression for time-to-event data by using both usual distributions (e.g. Exponential, Gamma, Weibull, Log-Normal) and user-defined distributions (Jackson, 2016).

For what concerns recursive partitioning there are many *R* packages that allow to build survival trees. Among these, many allow to grow trees as extensions of CART (see Table

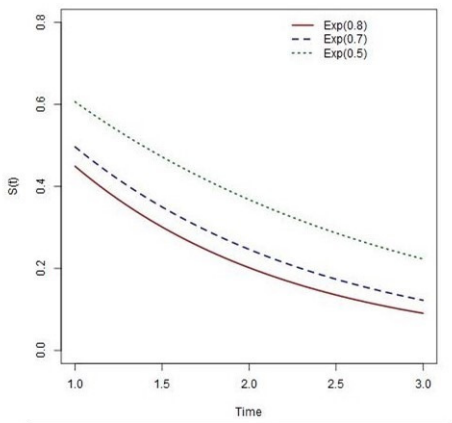
a



λ_1: 2 / λ_2: 0.9 / λ_3: 0.1

Sample size	Relative risk trees rpart	Conditional Inference trees ctree	Conditional Inference trees SurvCART
N=100	49.3%	76.9%	48.6%
N=250	69.6%	86.6%	79.6%
N=500	87.0%	88.0%	86.2%
N=1000	99.7%	84.7%	84.3%

b



λ_1: 0.8 / λ_2: 0.7 / λ_3: 0.5

Sample size	Relative risk trees rpart	Conditional Inference trees ctree	Conditional Inference trees SurvCART
N=100	0.1%	0.1%	0.0%
N=250	0.0%	0.8%	0.6%
N=500	3.2%	8.8%	13.0%
N=1000	25.2%	49.1%	53.6%

Figure 2: Survival curves in two different settings and results of a simulation study after having grown Relative Risk Trees (through the `rpart` function in *R*), Conditional Inference Trees with the *ctree* algorithm and Conditional Inference Trees with the *SurvCART* algorithm. a) Example of distant effect sizes with three well distinguished survival curves and related table reporting the percentage of fitted trees identifying the right partitions (values from a simulation study). b) Example of close effect sizes with approximately equal survival curves and related table reporting the percentage of fitted trees identifying the right partitions (values from a simulation study)

3). Two of the most used functions are `rpart` in the homonymous package (Therneau et al., 2015) and `ctree` (in the `partykit` package) (Hothorn et al., 2015b; Hothorn and Zeileis, 2015), which allow respectively to grow RRTs (LeBlanc and Crowley, 1992) and CITs (Hothorn et al., 2006). Finally, another noteworthy function is `SurvCART` in the `LongCART` package (Kundu, 2021), which has been introduced very recently to grow SCTs (Kundu and Ghosh, 2021).

Moreover, an extension of RRTs and CITs for left-truncated and right censored data has been provided in the `LTRCtrees` package (`LTRCART` and `LTRCIT` functions) (Fu et al., 2021). Other modified versions of CART for survival data are provided in the `partDSA` and `rocTree` packages (Lostritto et al., 2012; Sun et al., 2020b). The first one allows to implement the partitioning Deletion-Substitution-Addition algorithm, an algorithm that grows trees on the basis of both "and" and "or" conjunction of predictors (Lostritto et al., 2012); the second one, instead, grows survival trees maximizing a ROC-related

measure (Sun et al., 2020a).
Finally, another interesting recursive partitioning algorithm, different from CART, is the model-based algorithm, implementable with `mob` function (Zeileis and Hothorn, 2015) in the `partykit` package, which fits a segmented parametric model by growing a tree in which every leaf is associated with a specific model.

In this work only packages providing an extension to CART for right-censored data in a setting of independent censoring are presented. In particular, the focus is on `rpart`, `ctree` and `SurvCART` (due to the examined setting, for the last function only independent censoring is considered). This choice has been taken for the following reasons:

1. The `LTRCtrees` package is based on the `rpart` and `ctree` functions; so, for right-censored data `LTRCART` and `LTRCIT` provide identical results to those of `rpart` and `ctree`. The only difference in the functions is that `LTRCtrees` requires to specify a starting time (that, for right-censored data, can be set equal to 0);

2. The `rocTree` package provides a lot of splits and its computation time is slow. The algorithm can be of interest but it seems that it still needs some improvements (e.g. computation speed, clarity of the documentation);

3. The `partDSA` package provides little documentation for implementation of trees for censored data.

Table 3: Packages and functions in *R* for fitting survival trees for right-censored data (extensions of CART)

R package (function)	Algorithm
`rpart` (`rpart`)	Relative risk trees
`partykit` (`ctree`)	Conditional inference trees
`partDSA` (`partDSA`)	Partitioning Deletion-Substitution-Addition algorithm
`LTRCtrees` (`LTRCART`)	RRTs[a] for left-truncated and right-censored data
`LTRCtrees` (`LTRCIT`)	CITs[b] for left-truncated and right-censored data
`rocTree` (`rocTree`)	ROC-guided survival trees
`LongCART` (`SurvCART`)	Conditional inference trees for accounting censoring heterogeneity

[a] Relative Risk Trees; [b] Conditional Inference Trees with the `ctree` algorithm

Relative risk trees in *R*. Fitting a RRT requires, as already hinted, the use of the `rpart` function in the homonymous package. The resulting output is a survival tree whose terminal nodes contain (i) a measure of risk (LeBlanc and Crowley, 1992); (ii) the ratio between the number of events and observations in the node; and (iii) the number and percentage of observations included in the node (see Figure 3a). The function is

easy to implement; however, some doubts emerge concerning the interpretation of the node outcome. It should represent a measure of relative risk of the node with respect to the overall sample. However, in the available *R* documentation there are not clear examples about survival trees. It is straightforward to deduce that RRTs are grown as Poisson trees, but it is not exactly clear what the "estimated response rate" (outcome of a Poisson tree) means in survival analysis.

The most interesting elements (mainly to extract useful information for evaluating the performance with simulated data) of an `rpart` object are the following: (i) `frame`, a dataframe including the splitting variables used for growing the tree and the predicted outcome in each corresponding node and (ii) `splits`, a matrix that reports, for continuous variables, the cut-off used for splitting a node. For categorical variables this last matrix should be examined together with the `csplit` object (for more details see Therneau et al. (2015)). In particular, these objects are the most useful to examine if the fitted tree has the right structure we are simulating.

Conditional inference trees in *R* with `ctree`. Fitting a CIT can be done by using the `ctree` function in the `partykit` package. The resulting tree is a survival tree whose terminal nodes show the corresponding K-M survival curves (see Figure 3b). It is interesting to know that it is possible to obtain a plot similar to that of `ctree` also for RRTs, coercing the `rpart` object into a `partykit` one with the `as.party` function.
Moreover, as default node outcome it provides the median survival time, i.e. the time point at which 50% of the population has experienced the event. It is important to note that when the length of follow-up is shorter than the theoretical median survival time the corresponding predicted value will be equal to $+\infty$. For each observation it is also possible to obtain the estimated K-M survival function through the use of the `predict` function specifying `type="prob"`.

The algorithm provides very intuitive results. The only drawback is that the inner structure of a `ctree` object is quite complex and it seems that there is little documentation about it, making it difficult to extrapolate much information about trees structure. A possible way to go inside the object is reported hereafter. An alternative to understand how the tree has been grown and to extract the splitting variables is to look at the results of the structural change test (default test used by the function for growing the tree) through the `sctest` function in the `strucchange` package (Zeileis et al., 2015). This function shows the results (test statistic and p-value) of the parameter stability tests for any given node (Hothorn and Zeileis, 2015); therefore the most significant variables are easily identifiable. Moreover, the tree's path can be extracted through the `partykit:::list.rules.party` function. A tricky solution is to treat the resulting outcome (a character) as a text to extract the corresponding cut-off values.

Conditional inference trees in *R* with `SurvCART`. As already hinted, SCTs can be grown through the `SurvCART` function in the `LongCART` package. The resulting outcome is a list of many arguments. Among these, the `Treeout` matrix contains all the necessary information for evaluating, in a simulation study, if the fitted tree has the same structure of the theoretical one. Indeed, it shows summary information for each node (both

terminal and non-terminal) of the fitted tree. In particular, it includes the splitting variable (`var`), the cut-off value used for binary partitioning (`index`) and the indicator (True or False) of terminal node (`Terminal`).
Then, the `SurvCART` object also includes some `rpart` compatible objects, as `frame`, `splits` and `cptable`. Finally, it includes the matrix `subj.class`, which also reports the node assignment for each observation in the training set.

A simple plot can be obtained from this object, as shown in Figure 3c. The node outcome is the median survival time. Similarly to CITs, if the observed median survival time is greater than the length of follow-up, the algorithm returns NA. Moreover, K-M curves for each terminal node can be easily obtained through the `KMPlot.SurvCART` function.

The `SurvCART` function is easy to implement and, in addition, it is easily manageable and interpretable.

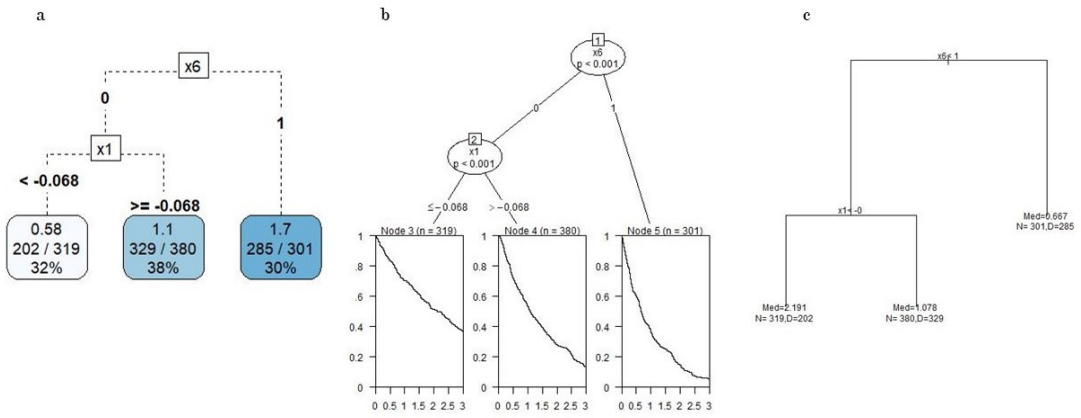


Figure 3: Example of plots of survival trees. a) Plot of a Relative Risk Tree, obtained through the `rpart` function. b) Plot of a Conditional Inference Tree obtained with the `ctree` algorithm. c) Plot of a Conditional Inference Tree obtained with the `SurvCART` algorithm

3.3 Performance evaluation

The last step involves performance evaluation of the resulting trees. Different packages are available in *R* for evaluating the performance of survival trees (see Table 4). Among these there are `pec` (Gerds, 2021), `SurvMetrics` (Zhou et al., 2022), `Hmisc` (Harrell Jr and Dupont, 2006), `survAUC` (Potapov et al., 2012) and `riskRegression` (Gerds et al., 2015).

More in detail, the C-index can be evaluated through: (i) `Hmisc::rcorr.cens`, (ii) `pec::cindex`, (iii) `SurvMetrics::Cindex` and (iv) `survAUC::UnoC`. The time-dependent AUCs can be instead evaluated through the `AUC.hc` and `Score` functions in the `survAUC`

and `riskRegression` packages respectively. Furthermore, the BS can be evaluated through the `Brier` function in the `SurvMetrics` package and `pec` function in the homonymous package. Moreover, the IBS can be evaluated through the `SurvMetrics::IBS` and `pec::pec` functions. Finally, the `riskRegression::IPA` function allows to evaluate the IPA; unfortunately it does not support survival trees. An important drawback when evaluating these performance measures in *R* concerns little documentation about almost all the available functions. Concerning this point, doubts emerged about three main aspects:

1. The type of weights used for evaluating the indices (e.g. Harrell's or Uno's C-index).

2. The type of predicted values required. For many functions it is not specified if risk or survival predicted values are required, implying possible wrong evaluation assessments. A clear example is provided by the C-index, that is a measure based on ranking; providing either survival or risk estimates (two quantities that have an inverse trend) would substantially change the results.

3. The censoring model used for evaluating the BS and IBS.

For what concerns the first point, after an in-depth analysis it has been verified that `Hmisc::rcorr.cens` and `SurvMetrics::Cindex` evaluate Harrell's C-index (Harrell et al., 1982); while `survAUC::UnoC` provides Uno's C-index (Uno et al., 2011). Moreover, `pec::cindex` provides a time-truncated C-index, that can be seen as a general case of Uno's C-index (Gerds et al., 2013). Somer's d can be easily obtained from Harrell's C-index, as shown in Equation (5); it is also provided by `Hmisc::rcorr.cens` (D_{xy} quantity).

For the second point, it has been empirically verified, using simulated data, that `Hmisc` and `SurvMetrics` require survival probabilities, while `survAUC` requires a risk estimate. If it is not possible to obtain this measure from trees, a tricky solution is to provide as argument a decreasing function (e.g. just the opposite) of survival probabilities, maintaining therefore the right ranking (of course also the vice-versa holds).

For the last point, after the analysis of the `SurvMetrics` functions' code (no information was provided in the documentation) it has been verified that BS (and, consequently, IBS) is evaluated using K-M estimator for obtaining the inverse probability censoring weights (IPCWs). Differently, as it can be seen from `pec` documentation, this last one allows the use of various censoring models (Gerds, 2021).

An important difference between `pec` and the other packages is that it requires to transform the fitted model in a compatible one. Two functions exist for RRTs and CITs, i.e. `pecRpart` and `pecCtree`. In particular, the last function grows CITs through the oldest version of the package, i.e. `party` (Hothorn et al., 2015a). Unfortunately `pec` does not work with SCTs (introduced more recently than `pec`). Differently from `pec` all the other packages only require a vector of predicted values. Unfortunately, also the main classical functions for obtaining predictions, as `predict`, are not compatible with `SurvCART`. Future research will involve the construction of a new function that

allows to evaluate SCTs' performance. However, it is still possible to evaluate an overall concordance measure for SCTs extracting the estimated median survival time for each observation from the `subj.class` matrix of the `SurvCART` object.

Another issue concerning performance assessment is related to the estimation of survival probabilities at given time points. A possible solution is to use the `predictSurvProb` function in `pec`; however, in some specific cases (e.g. when all subjects in a node experienced the event) this function returns erroneously NA (see the Example in Supplementary File 1, available at `https://bodai.unibs.it/ml4sd/`). This implies the impossibility of evaluating many metrics; future research work will address this aspect.

Also due to this reason, `pec::cindex` and `survAUC::UnoC` do not provide the same results for survival trees, even if they should (during a simulation study it has been verified, for example, that the two functions provide the same results if a Cox PH regression model is assessed). In particular, it seems that `pec` presents some problems after given time points.

Finally, for all the functions that allow to evaluate BS and IBS, an issue related to estimated survival probability was encountered. Indeed, as already hinted, `predictSurvProb` (the function used internally by `pec` and used for providing survival probabilities to `SurvMetrics`) returns, for example, NA after the time point in which the last events occurred in a node with only uncensored subjects. The presence of these values implies, then, the impossibility of estimating these indices after that time point. The solution to this problem is deferred to future research work.

Table 4: Packages and functions in *R* for performance evaluation of survival trees

R package (function)	Performance measure
`Hmisc` (`rcorr.cens`)	Harrell's Concordance index & Somer's d
`SurvMetrics` (`Cindex`)	Harrell's Concordance index
`pec` (`cindex`)	Time-Truncated Concordance index
`survAUC` (`UnoC`)	Uno's Concordance index
`riskRegression` (`Score`)	Time-dependent AUC
`survAUC` (`AUC.hc`)	Time-dependent AUC
`SurvMetrics` (`Brier`)	Brier Score at a single time-point with Kaplan-Meier IPCW
`pec` (`pec`)	Brier Score
`SurvMetrics` (`IBS`)	Integrated Brier Score with Kaplan-Meier IPCW
`pec` (`pec`)	Integrated Brier Score

In addition to all these measures, in a simulation study it is interesting to examine how many fitted trees report the right data structure as well. To this extent it is also

interesting to consider all the possible equivalent structures of a tree. Indeed, a same partition can be induced by trees with a different structure. An example of this is reported in Figure 4. The figure shows that two different trees, with different number of terminal nodes, lead to the same partition. Indeed, in the less parsimonious tree (right side figure) the observations in the two nodes corresponding to the same theoretical partition, i.e. $X_2 = 1 \cap X_1 \geq 0$ and $X_2 = 1 \cap X_1 < 0$, come from the same distribution (they have the same λ parameter).

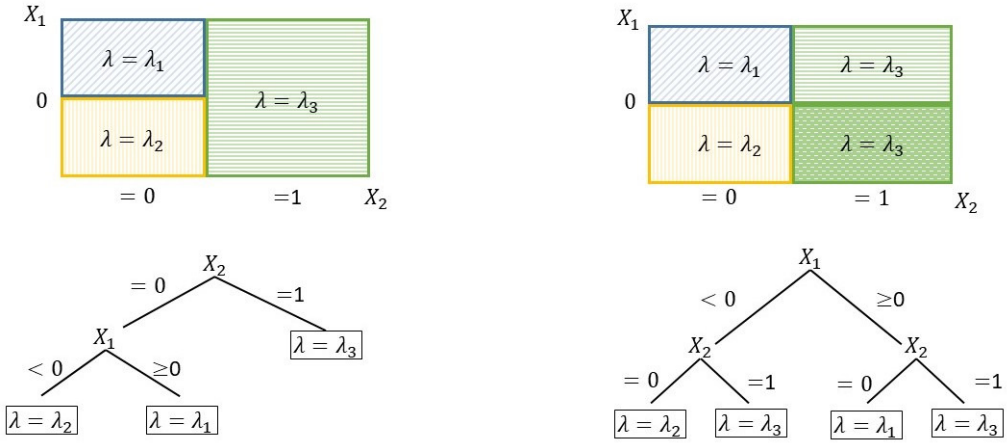


Figure 4: Example of two equivalent trees induced by the same theoretical partition. X_1 is a continuous variable split into two partitions by the cut-off 0; X_2 is a dichotomous variable. Each partition is characterized by a given parameter λ. The two resulting trees have a different structure (different root node, different order of splitting variables, different number of terminal leaves) but they are equivalent

4 Example

Below a practical example of how simulating survival data, fitting RRTs, CITs and SCTs and evaluating their performance with *R* is provided (additional code and interpretation of results is available at `https://bodai.unibs.it/ml4sd/` - Supplementary File 1).
Firstly, a set of covariates is generated. In particular, 10 covariates were randomly generated: the first half followed a Standard Normal distribution (X_1, X_2, X_3, X_4 and X_5) and the second half a Bernoulli distribution with different probability parameters p ranging respectively from 0.3 to 0.7 with steps of 0.1 (X_6, X_7, X_8, X_9 and X_{10}).

```
> n_rv <- 10                    # Number of covariates
> prob_rvar <- seq(0.3,0.7,0.1) # Parameters for the Bernoulli r.v.
```

```
> N <- 1000                        # Sample size

> library(mvtnorm)
> set.seed(70522)
> Norm_RV <- rmvnorm(n=N, mean=rep(0,n_rv/2), sigma=diag(n_rv/2))

> library(MultiRNG)
> set.seed(70522)
> Ber_RV <- draw.correlated.binary(no.row=N, d=n_rv/2, prop.vec=prob_rvar,
                                   corr.mat=diag(n_rv/2))

> cov <- as.data.frame(cbind(Norm_RV,Ber_RV))
> colnames(cov) <- paste0("X", 1:n_rv)
```

Since the aim is to fit recursive partitioning models, different distributions for survival times in the terminal nodes must be assumed. In this example a Data Generating process with a tree structure was used; the tree was characterized by three terminal nodes with the following partitions and distribution of the survival times:

- $X6 = 1$: $T_i \sim Exp(2)$;
- $X6 = 0$ and $X1 \geq 0$: $T_i \sim Exp(0.9)$;
- $X6 = 0$ and $X1 < 0$: $T_i \sim Exp(0.1)$.

```
> node1 <- cov[,"X6"]==1
> node2 <- cov[,"X6"]==0 & cov[,"X1"]>=0
> node3 <- cov[,"X6"]==0 & cov[,"X1"]<0
```

Finally, the follow-up duration was fixed equal to 3 years.

```
> maxtime <- 3
> tvec <- seq(1,maxtime,0.1)
```

During a simulation study many datasets of different sample sizes should be examined. After the covariates have been randomly generated, survival data can be simulated. For simulating survival data the `simsurv` package was used. This function requires the following arguments: (i) the distribution assumed for survival times; (ii) the corresponding parameters; (iii) a dataframe `x` with the set of covariates (if not - as in the recursive partitioning setting - only an `id` column is required); (iv) `maxt`, that corresponds to the length of follow-up (up to now called `maxtime`); and (v) the seed to use in order to get reproducible results. As outcome a dataset with three columns (`id`, `eventtime` and `status`) is returned.

```
> simdata <- as.data.frame(matrix(data=NA, nrow=N, ncol=2))
> colnames(simdata) <- c("eventtime","status")
```

```
> library(simsurv)
# Node1
> simdata[node1,1:2] <- simsurv(dist="exponential", lambda=2,
         x=as.data.frame(1:sum(node1)), maxt=maxtime, seed=7052022)[,2:3]

# Node2
> simdata[node2,1:2] <- simsurv(dist="exponential", lambda=0.9,
         x=as.data.frame(1:sum(node2)), maxt=maxtime, seed=7052022)[,2:3]

# Node3
> simdata[node3,1:2] <- simsurv(dist="exponential", lambda=0.1,
         x=as.data.frame(1:sum(node3)), maxt=maxtime, seed=7052022)[,2:3]

> data <- cbind(simdata,cov) # Dataset with survival data and covariates
> data[,8:12] <- lapply(data[,8:12], as.factor)
```

Once the different datasets have been simulated, they can be split in training and test set and survival trees can be fitted using the training set:

```
> set.seed(70522)
# To take a similar percentage of events in the training and test sets:
> library(caret)
> trainset <- createDataPartition(y=data$status, p=0.5, list=F)
> data_train <- data[trainset,]
> data_test <- data[-trainset,]
```

Below a code example for fitting RRTs, CITs and SCTs.

 # Relative risk trees

The `rpart` function uses Poisson trees for building a RRT. The required arguments are: (i) a survival formula, i.e. a formula with a `Surv` object (in the `survival` package (Therneau and Lumley, 2015)) as dependent variable; (ii) a training set; and (iii) usual control options (e.g. `minsplit`, minimum number of observations in a node; `xval`, number of cross-validations). The resulting tree can then be pruned with the usual cost-complexity algorithm used by CART (`prune` function) using a suitable value for the complexity parameter (`cp`), e.g. the one that leads to the lowest cross-validation error.

```
> library(survival)
> library(rpart)
> RRT <- rpart(formula=Surv(eventtime, status)~., data=data_train,
            control=rpart.control(usesurrogate=2, minsplit=20, xval=10))
# Pruning RRT
> minerr <- which.min(RRT$cptable[,"xerror"])
```

```
> bestcp <- RRT$cptable[minerr,"CP"]
> pruned_RRT <- prune(tree=RRT, cp=bestcp)
# Plotting RRT
> library(rpart.plot)
> rpart.plot(x=pruned_RRT, type=5, tweak=1.0, gap=0.02, branch.lty=2)
# An alternative way of plotting RRT
> library(partykit)
> plot(as.party(pruned_RRT))
```

 # Conditional Inference Trees with `ctree`

The `ctree` function in `partykit` allows to grow a CIT. It requires as arguments: (i) a survival formula, (ii) a training set and (iii) some control options as the test statistics for variable and split-point selection (`teststat` and `splitstat` respectively), the significance level for variable selection (`alpha`) and the minimum sum of weights in a node in order to be considered for splitting (`minsplit`). Many other options can be set as shown in the help of the function.

```
> library(partykit)
> CIT <- ctree(formula=Surv(eventtime, status)~., data=data_train,
               control=ctree_control(minsplit=20))
# Plotting CIT
> plot(CIT)
```

 # Conditional Inference Trees with `SurvCART`

SCTs can be grown using the `SurvCART` function in `LongCART` package. It requires that training data includes an `"id"` column and that categorical variables are numerically coded. Once the training dataset has been formatted following these requirements, trees can be grown. The input arguments are: (i) the data on which training the algorithm (`data`); (ii) the name of the "id" variable (`patid`); (iii) the name of the time variable (`timevar`); (iv) the name of the status variable (`censorvar`); (v) a vector including the names of the partitioning variables to use (`gvars`) and (vi) a vector indicating the type of each partitioning variable (`tgvars`). This argument assumes value 1 if the corresponding variable in `gvars` is continuous and 0 if the variable is categorical. Moreover, it is necessary to specify (vii) the assumed distribution for survival times (`time.dist`). By default censoring is considered homogeneous (as in this hypothesized setting) and the related argument, `censdist`, is set equal to NA. However, it is also possible to set a particular distribution for censoring when the interest is also in censoring heterogeneity. Possible assumptions for both the time-to-event and censoring distributions are `"exponential"`, `"weibull"`, `"lognormal"` and `"normal"`. Finally, control options can be set, as the significance level of the parameter instability test (`alpha`), and the minimum number of observations in a node (`minsplit`).

```
### Data pre-processing
> data_train_SCT <- data_train
```

```
> id <- 1:nrow(data_train_SCT)
> data_train_SCT <- cbind(data_train_SCT,id)
> data_train_SCT[,paste0("X",6:10)] <-
                  lapply(data_train_SCT[,paste0("X",6:10)], as.numeric)
> data_train_SCT[,paste0("X",6:10)] <-
                  data_train_SCT[, paste0("X",6:10)]-1

> library(LongCART)
> SCT <- SurvCART(data=data_train_SCT, patid="id", timevar="eventtime",
                  censorvar="status", gvars=paste0("X",1:10),
                  tgvars=c(1,1,1,1,1,0,0,0,0,0), time.dist="exponential",
                  cens.dist="NA", minsplit=20)
# Plotting SCT
> par(xpd=TRUE)
> plot(SCT, compress=TRUE)
> text(SCT, use.n=TRUE)
# Plotting Kaplan-Meier curves estimated in the terminal nodes
> KMPlot.SurvCART(x=SCT, type=1)
```

Performance evaluation

Once all the trees are fitted their performance can be evaluated. Beyond the above-mentioned performance measures (C-index, time-dependent AUC, BS and IBS), the size and the structure of fitted trees can be compared to those of the theoretical one (considering also equivalent structures). For doing that it is necessary to extract the needed information from the resulting objects. This means, for example, working with the `frame` matrix of `rpart`, or with the `Treeout` matrix of `SurvCART`. An example of *R* code has been provided at `https://bodai.unibs.it/ml4sd/` - Supplementary File 2.

Then, for evaluating classical performance measures in the test set, the following functions can be used. Only examples for RRTs and CITs are shown below (due to the impossibility of using `predict`, `predictSurvProb` and `pec` for SCTs).

 # C-index

Harrell's C-index can be evaluated through `rcorr.cens` and `Cindex` in `Hmisc` and `SurvMetrics` respectively. The first one requires as first object a vector of survival predictions (one for each observation) and a `Surv` object. The function returns a vector containing the C-index value and Somer's *d* `Dxy`. It also provides other information like the number of comparable (`"relevant"`), concordant (`"concordant"`) and uncertain (`"uncertain"`) pairs.

If it is not possible to obtain survival estimates (but only risk ones) a possible solution, as already hinted, is to replace risk predictions with a decreasing function of these values. This is due to the fact that concordance is a rank measure and so it is based only on the ordering of predicted values. To obtain an overall measure of concordance for RRTs, the `predict` function can be used. It returns for each observation the corresponding node outcome (measure of risk).

```
### Risk predictions for RRT
> pred_risk_RRT <- predict(object=pruned_RRT, newdata=data_test)
### Survival outcome in the test set
> ySurv_test <- Surv(data_test$eventtime, data_test$status)
> library(Hmisc)
> C_ind_RRT <- rcorr.cens(x= -pred_risk_RRT, S=ySurv_test)
```

For an overall C-index for `ctree` it is possible to estimate the median survival time for each observation and to provide it to the function. If Inf values are returned, a possible solution is to replace them with a value greater than all the other observed median survival times (valid only for evaluating ranking measures).

```
### Survival predictions for CIT
> pred_surv_CIT <- predict(object=CIT, newdata=data_test, type="response")
# Replacement of Inf values with a value greater than all the others:
> pred_surv_CIT[which(pred_surv_CIT==Inf)] <-
                          max(pred_surv_CIT[-which(pred_surv_CIT==Inf)])+3
> C_ind_CIT <- rcorr.cens(x=pred_surv_CIT, S=ySurv_test)
```

The `SurvMetrics::Cindex` requires the same arguments of `rcorr.cens`, i.e. a `Surv` object and a vector of predicted survival probabilities or survival times:

```
> library(SurvMetrics)
> C_ind_RRT <- Cindex(object=ySurv_test, predicted= -pred_risk_RRT)
> C_ind_CIT <- Cindex(object=ySurv_test, predicted=pred_surv_CIT)
```

If, instead, interest is on the C-index at given time points, a truncated C-index can be evaluated. The first thing to do is to evaluate a matrix of predicted survival probabilities at the different time points. These probabilities can be obtained with the `pec:::predictSurvProb` function. This last function returns a matrix in which for each observation (row) the survival probability is evaluated at given time points (columns). It is compatible with a `pec` object; then, the `rpart` and `ctree` objects have to be converted with `pecRpart` and `pecCtree`. Below an example for both `rpart` and `ctree` is provided; in this last case `pec` converts it in a `party` object, as already hinted.

```
> library(pec)
> RRT_pec <- pecRpart(formula=Surv(eventtime, status)~., data=data_train,
                      cp=bestcp)
### Survival predictions for RRT
> pred_RRT_pec <- predictSurvProb(object=RRT_pec, newdata=data_test,
                                  times=tvec)
> colnames(pred_RRT_pec) <- paste0("time=", tvec)

> CIT_pec <- pecCtree(formula=Surv(eventtime, status)~., data=data_train)
### Survival predictions for CIT
```

```
> pred_CIT_pec <- predictSurvProb(object=CIT_pec, newdata=data_test,
                                  times=tvec)
> colnames(pred_CIT_pec) <- paste0("time=", tvec)
```

Once the survival probabilities have been estimated, a single column of the survival probabilities matrix has to be provided to the *R* functions. The idea is to evaluate an index similar to the one used by `pec` (shown hereafter). In particular, for each time point the status indicator δ_i is set equal to 1 only for those subjects who experienced the event before the time point of interest. Then, the time interval is truncated; thus all times (`eventtime`) greater than the given time point (`tvec[i]`) are considered equal to this last one, used as upper bound of the time interval:

```
> C_ind_RRT_vec <- C_ind_RRT_vec1 <- rep(NA, length(tvec))
> C_ind_CIT_vec <- C_ind_CIT_vec1 <- rep(NA, length(tvec))
> for(i in 1:length(tvec))
>  {
>  eventtime <- data_test$eventtime; status <- data_test$status
>  status[eventtime>tvec[i]] <- 0
>  eventtime[eventtime>tvec[i]] <- tvec[i]
>  ySurv <- Surv(eventtime, status)
>  C_ind_RRT_vec[i] <- rcorr.cens(x=pred_RRT_pec[,i], S=ySurv)
>  C_ind_RRT_vec1[i] <- Cindex(object=ySurv, predicted=pred_RRT_pec[,i])
>  C_ind_CIT_vec[i] <- rcorr.cens(x=pred_CIT_pec[,i], S=ySurv)
>  C_ind_CIT_vec1[i] <- Cindex(object=ySurv, predicted=pred_CIT_pec[,i])
>   }
```

Uno's C-index can be evaluated through the `UnoC` function in the `survAUC` package. This function requires three arguments: (i) a `Surv` object containing the outcome of the training set; (ii) a `Surv` object containing the outcome of the test set and (iii) a vector of predicted risk values for the test set.

```
### Survival outcome in the training set
> ySurv_train <- Surv(data_train$eventtime, data_train$status)

> library(survAUC)
> UnoC_RRT <- UnoC(Surv.rsp=ySurv_train, Surv.rsp.new=ySurv_test,
                   lpnew=pred_risk_RRT)
> UnoC_CIT <- UnoC(Surv.rsp=ySurv_train, Surv.rsp.new=ySurv_test,
                   lpnew= -pred_surv_CIT)
```

Finally, an alternative function for evaluating the concordance index is `cindex` in the `pec` package, that as already hinted evaluates the time-truncated C-index (Gerds et al., 2013). The function requires as arguments: (i) a list of models for which evaluating the performance index; (ii) the formula used for growing the tree; (iii) the dataset on which evaluating the performance and (iv) a vector of times on which evaluating the

index. Different methods for estimating IPCWs are available; among these there are the Cox regression PH model (`"cox"`) and the K-M estimator (`"marginal"`) (default method). Below an example of code for evaluating performance of both RRTs and CITs at different time points. The procedure used by `cindex` in `pec` is similar to that shown above with the `rcorr.cens` and `Cindex` functions.

```
> library(pec)
> TruncC_RRT_CIT <- cindex(object=list("RRT"=RRT_pec, "CIT"=CIT_pec),
                       formula=Surv(eventtime, status)~., data=data_test,
                       eval.times=tvec)
```

Computational issues could occur when there are some NAs in the matrix of predicted survival probabilities (see the example in the Supplementary File 1).

 # Time-dependent AUC

Time-dependent AUCs can be evaluated through `Score` and `AUC.hc` functions in `riskRegression` and `survAUC` packages respectively. The first function requires (i) a list of models to evaluate (`object`); (ii) the metrics; (iii) the used formula; (iv) the data; (v) the censoring model to use; (vi) a vector of times on which evaluating the measure of interest. Unfortunately, `Score` provides an error related to the inner function `predictRisk` when evaluating a `ctree` object. Firstly, it is important to point out that the function requires a `party` object and not a `partykit` one. Secondly, instead of extracting a measure of risk, it seems that it extracts for each observation the size of the corresponding node. The problem can be solved modifying this function, extracting a measure of risk from the tree (e.g. K-M hazard estimate - see Supplementary File 1). Below an example for evaluating time-dependent AUCs for an `rpart` object with `Score` is provided.

```
> library(riskRegression)
> AUC_Score_RRT <- Score(object=list("RRT"=pruned_RRT), metrics="AUC",
                formula=Surv(eventtime, status)~X1+X2+X3+X4+X5+X6+X7+X8+
                X9+X10, data=data_test, cens.model="km", times=tvec)
```

Differently, `survAUC::AUC.hc` requires a `Surv` object containing the outcome of the training data; a `Surv` object containing the outcome of the test set and a vector of risk predictions. Attention has to be paid to this function. During some applications on real data it provided values greater than one!

```
> library(survAUC)
> AUC_hc_RRT <- AUC.hc(Surv.rsp=ySurv_train, Surv.rsp.new=ySurv_test,
                       lpnew=pred_risk_RRT, times=tvec)
> AUC_hc_CIT <- AUC.hc(Surv.rsp=ySurv_train, Surv.rsp.new=ySurv_test,
                       lpnew= -pred_surv_CIT, times=tvec)
```

 # Brier Score and Integrated Brier Score

Two of the available packages for evaluating BS and IBS are `SurvMetrics` and `pec`. The two functions in `SurvMetrics` require (i) a `Surv` object evaluated in the test set; (ii) a vector (for BS) or a matrix (for IBS) of survival probabilities of each observation in the time point(s) of interest and (iii) the time point (or vector of time points) at which evaluating the two indices. The package uses K-M IPCWs. An example is provided below for evaluating the BS at time 1 and the IBS for both RRT and CIT.

```
> library(SurvMetrics)
> brier_RRT_1 <- Brier(object=ySurv_test, pre_sp=pred_RRT_pec[,1],
                       t_star=1)
> ibs_RRT <- IBS(object=ySurv_test, sp_matrix=pred_RRT_pec,
                 IBSrange=tvec)
> brier_CIT_1 <- Brier(object=ySurv_test, pre_sp=pred_CIT_pec[,1],
                       t_star=1)
> ibs_CIT <- IBS(object=ySurv_test, sp_matrix=pred_CIT_pec,
                 IBSrange=tvec)
```

The two indices can also be evaluated through the `pec` function in the homonymous package. Differently from `SurvMetrics` it allows to specify a different censoring model than K-M (`"marginal"`). Other possibilities are `"cox"`, `"nonpar"` and `"aalen"` (Gerds, 2021). BS and IBS can be evaluated as follows (for the example I evaluated BS at time 1):

```
> library(pec)
> bs_1 <- pec(object=list("RRT"=RRT_pec, "CIT"=CIT_pec), data=data_test,
              formula=Surv(eventtime, status)~., times=1, exact=F,
              cens.model="marginal")

> ibs <- pec(object=list("RRT"=RRT_pec, "CIT"=CIT_pec), data=data_test,
             formula=Surv(eventtime, status)~., times=tvec,
             cens.model="marginal")
```

Issues emerged, as already said in Subsection 3.3, with these functions when NA occurred in the estimated survival probabilities (see the example in the Supplementary File 1).

5 Discussion

In conclusion, survival trees are a useful and interesting method for defining subgroups of subjects according to their survival experience. However, even if there are a lot of proposed methods, many of these present little documentation mainly concerning the use of *R* packages and functions for fitting survival trees and evaluating their performance. Moreover, there does not exist an harmonization of all these proposals, making the approach to this kind of new methods for survival analysis difficult. Furthermore, it

also emerged that evaluating performance for this kind of methods is quite complex due to little documentation and computational issues. This work aims to shed light on the topic and to provide a practical guide with some hints for simulating right-censored data, fitting recursive partitioning algorithms and evaluating their performance in *R*. Future research work is still needed to solve some of the identified drawbacks and it will involve, among the others, the possibility of assessing SCTs performance and the evaluation of performance indexes as the Brier score and its integrated version when NA values occur in the estimated survival probabilities.

Supplementary Materials

Supplementary Materials are available online at `https://bodai.unibs.it/ml4sd/`.

Acknowledgements

I would like to thank Prof. Paola Zuccolotto and Dr. Marco Sandri for their precious help in developing this research work. Thanks also to the two anonymous reviewers for their useful comments that have substantially improved the quality of the manuscript.

References

Aalen, O. (1978). Nonparametric inference for a family of counting processes. *The Annals of Statistics*, pages 701–726.

Bou-Hamad, I., Larocque, D., and Ben-Ameur, H. (2011). A review of survival trees. *Statistics surveys*, 5:44–71.

Bou-hamad, I., Larocque, D., Ben-Ameur, H., Mâsse, L. C., Vitaro, F., and Tremblay, R. E. (2009). Discrete-time survival trees. *Canadian Journal of Statistics*, 37(1):17–32.

Breiman, L. (2001). Statistical modeling: The two cultures (with comments and a rejoinder by the author). *Statistical Science*, 16(3):199 – 231.

Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984). *Classification and regression trees*. Routledge.

Brier, G. W. et al. (1950). Verification of forecasts expressed in terms of probability. *Monthly weather review*, 78(1):1–3.

Brilleman, S. L., Wolfe, R., Moreno-Betancur, M., and Crowther, M. J. (2021). Simulating survival data using the simsurv r package. *Journal of Statistical Software*, 97(1):1–27.

Ciampi, A., Chang, C.-H., Hogg, S., and McKinney, S. (1987). Recursive partition: A versatile method for exploratory-data analysis in biostatistics. In *Biostatistics*, pages 23–50. Springer.

Ciampi, A., Thiffault, J., Nakache, J.-P., and Asselain, B. (1986). Stratification by stepwise regression, correspondence analysis and recursive partition: a comparison of

three methods of analysis for survival data with covariates. *Computational statistics & data analysis*, 4(3):185–204.

Clarke, J. and West, M. (2008). Bayesian weibull tree models for survival analysis of clinico-genomic data. *Statistical methodology*, 5(3):238–262.

Collett, D. (2015). *Modelling survival data in medical research*. CRC press.

Cox, D. R. (1972). Regression models and life-tables. *Journal of the Royal Statistical Society. Series B (Methodological)*, 34(2):187–220.

Crowther, M. J. and Lambert, P. C. (2012). Simulating complex survival data. *The Stata Journal*, 12(4):674–687.

Davis, R. B. and Anderson, J. R. (1989). Exponential survival trees. *Statistics in medicine*, 8(8):947–961.

Fu, W., Jeffrey, S., and Wenbo, J. (2021). Package 'ltrctrees', version 1.1.1. *Available online: https://cran.r-project.org/web/packages/LTRCtrees/LTRCtrees.pdf*.

Fu, W. and Simonoff, J. S. (2017a). Survival trees for interval-censored survival data. *Statistics in medicine*, 36(30):4831–4842.

Fu, W. and Simonoff, J. S. (2017b). Survival trees for left-truncated and right-censored data, with application to time-varying covariate data. *Biostatistics*, 18(2):352–369.

Gerds, T. A. (2021). Package 'pec', version 2020.11.17.

Gerds, T. A., Kattan, M. W., Schumacher, M., and Yu, C. (2013). Estimating a time-dependent concordance index for survival prediction models with covariate dependent censoring. *Statistics in medicine*, 32(13):2173–2184.

Gerds, T. A., Scheike, T. H., and Gerds, M. T. A. (2015). Package 'riskregression', version 2020.12.08.

Gordon, L. and Olshen, R. A. (1985). Tree-structured survival analysis. *Cancer treatment reports*, 69(10):1065–1069.

Graf, E., Schmoor, C., Sauerbrei, W., and Schumacher, M. (1999). Assessment and comparison of prognostic classification schemes for survival data. *Statistics in medicine*, 18(17-18):2529–2545.

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

Harrell Jr, F. E. and Dupont, M. C. (2006). The hmisc package. *R package version*, 3(0–12):3.

Hothorn, T., Hornik, K., Strobl, C., Zeileis, A., and Hothorn, M. T. (2015a). Package 'party', version 1.3-7. *Package Reference Manual for Party Version 0.9-998*, 16:37.

Hothorn, T., Hornik, K., and Zeileis, A. (2006). Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical statistics*, 15(3):651–674.

Hothorn, T., Hornik, K., and Zeileis, A. (2015b). ctree: Conditional inference trees. *The comprehensive R archive network*, 8.

Hothorn, T. and Zeileis, A. (2015). partykit: A modular toolkit for recursive partytioning

in r. *The Journal of Machine Learning Research*, 16(1):3905–3909.

Jackson, C. H. (2016). flexsurv: a platform for parametric survival modeling in r. *Journal of statistical software*, 70.

Kaplan, E. l. and Paul, M. (1958). Nonparametric estimation from incomplete observations. *Journal of the American Statistical Association*, 53(282):457–481.

Kattan, M. W. and Gerds, T. A. (2018). The index of prediction accuracy: an intuitive measure useful for evaluating risk prediction models. *Diagnostic and prognostic research*, 2(1):1–7.

Kropko, J. and Jeffrey, J. H. (2019). coxed: An r package for computing duration-based quantities from the cox proportional hazards model. *R J.*, 11(2):38.

Kundu, M. G. (2021). Package 'longcart', version 3.1. *Available online: cran. r-project. org/web/packages/LongCART/LongCART. pdf.*

Kundu, M. G. and Ghosh, S. (2021). Survival trees based on heterogeneity in time-to-event and censoring distributions using parameter instability test. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 14(5):466–483.

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

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

Lostritto, K., Strawderman, R. L., and Molinaro, A. M. (2012). A partitioning deletion/substitution/addition algorithm for creating survival risk groups. *Biometrics*, 68(4):1146–1156.

Molinaro, A. M., Dudoit, S., and Van der Laan, M. J. (2004). Tree-based multivariate regression and density estimation with right-censored data. *Journal of Multivariate Analysis*, 90(1):154–177.

Moriña, D., Navarro, A., et al. (2014). The r package survsim for the simulation of simple and complex survival data. *Journal of Statistical Software*, 59(2):1–20.

Nelson, W. (1969). Hazard plotting for incomplete failure data. *Journal of Quality Technology*, 1(1):27–52.

Nelson, W. (1972). Theory and applications of hazard plotting for censored failure data. *Technometrics*, 14(4):945–966.

Park, S. Y., Park, J. E., Kim, H., and Park, S. H. (2021). Review of statistical methods for evaluating the performance of survival or other time-to-event prediction models (from conventional to deep learning approaches). *Korean Journal of Radiology*, 22(10):1697.

Perrigot, R., Cliquet, G., and Mesbah, M. (2004). Possible applications of survival analysis in franchising research. *The International Review of Retail, Distribution and Consumer Research*, 14(1):129–143.

Peto, R. and Peto, J. (1972). Asymptotically efficient rank invariant test procedures. *Journal of the Royal Statistical Society: Series A (General)*, 135(2):185–198.

Potapov, S., Adler, W., Schmid, M., and Potapov, M. S. (2012). Package 'survauc'.

Statistics in Medicine, 25:3474–3486.

R Core Team (2021). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Rahman, M. S., Ambler, G., Choodari-Oskooei, B., and Omar, R. Z. (2017). Review and evaluation of performance measures for survival prediction models in external validation settings. *BMC medical research methodology*, 17(1):1–15.

Segal, M. R. (1988). Regression trees for censored data. *Biometrics*, pages 35–47.

Strasser, H. and Weber, C. (1999). On the asymptotic theory of permutation statistics.

Sun, Y., Chiou, S. H., and Wang, M.-C. (2020a). Roc-guided survival trees and ensembles. *Biometrics*, 76(4):1177–1189.

Sun, Y., Wang, M.-C., and Chiou, S. H. (2020b). Package 'roctree', version 1.1.1. *Available online: https://cran.r-project.org/web/packages/rocTree/rocTree.pdf*.

Therneau, T., Atkinson, B., and Ripley, B. (2015). Package 'rpart', version 4.1-15. *Available online: cran. ma. ic. ac. uk/web/packages/rpart/rpart. pdf (accessed on 20 April 2016)*.

Therneau, T. and Atkinson, E. (2020). 1 the concordance statistic.

Therneau, T. M. and Lumley, T. (2015). Package 'survival', version 3.2-10. *R Top Doc*, 128(10):28–33.

Uno, H., Cai, T., Pencina, M. J., D'Agostino, R. B., and Wei, L.-J. (2011). On the c-statistics for evaluating overall adequacy of risk prediction procedures with censored survival data. *Statistics in medicine*, 30(10):1105–1117.

Wang, P., Li, Y., and Reddy, C. K. (2019). Machine learning for survival analysis: A survey. *ACM Computing Surveys (CSUR)*, 51(6):1–36.

Zeileis, A. and Hothorn, T. (2015). Parties, models, mobsters: A new implementation of model-based recursive partitioning in r.

Zeileis, A., Hothorn, T., and Hornik, K. (2008). Model-based recursive partitioning. *Journal of Computational and Graphical Statistics*, 17(2):492–514.

Zeileis, A., Leisch, F., Hornik, K., Kleiber, C., Hansen, B., Merkle, E. C., and Zeileis, M. A. (2015). Package 'strucchange', version 1.5-2. *Journal of Statistical Software*.

Zhou, H., Cheng, X., Wang, S., Zou, Y., Wang, H., Zhou, M. H., Error, I. S., Error, I. A., and Error, M. A. (2022). Package 'survmetrics', version 0.4.0.