UNIVERSITÀ DEGLI STUDI DI BRESCIA

DOTTORATO DI RICERCA IN

Modelli e metodi per l'Economia e il Management
(Analytics for Economics and Management - AEM)

Settore Scientifico Disciplinare
SSD SECS-S/01 - Statistica

CICLO XXXV

Statistical Models and Machine Learning for Survival Data Analysis

Ambra Macis

Relatore

Prof. Paola Zuccolotto, Università degli studi di Brescia

Correlatore

Dr. Marco Sandri, BODaI-Lab, Università degli studi di Brescia

Secondo Correlatore

Prof. Marica Manisera, Università degli studi di Brescia

Acknowledgements

An enormous thank you to my supervisors and my co-supervisor Paola Zuccolotto, Marco Sandri and Marica Manisera. During these three years they guided me through the difficulties encountered and provided me unlimited support. They have always been there for me professionally, but also and above all from a human point of view. Thank you for having made this journey easier by filling it with good times.

A sincere thanks also to Eugenio Brentari, Maurizio Carpita and Silvia Golia who shared this journey with me.

Thanks to my family for their constant encouragement and their continuous support.

Last but not least thanks to Enrico for believing in me. Thanks for the infinite patience and for having always been there in the most difficult moments.

Abstract

The main topic of this thesis is survival analysis, a collection of methods used in longitudinal studies in which the interest is not only in the occurrence (or not) of a particular event, but also in the time needed for observing it.

Over the years, firstly statistical models and then machine learning methods have been proposed to address studies of survival analysis. The first part of the work provides an introduction to the basic concepts of survival analysis and an extensive review of the existing literature. In particular, the focus has been set on the main statistical models (nonparametric, semiparametric and parametric) and, among machine learning methods, on survival trees and random survival forests. For these methods the main proposals introduced during the last decades have been described.

In the second part of the thesis, instead, my research contributions have been reported. These works mainly focused on two aims: (1) the rationalization into a unified protocol of the computational approach, which nowadays is based on several existing packages with few documentation, several still obscure points and also some bugs, and (2) the application of survival data analysis methods in an unusual context where, to our best knowledge, this approach had never been used.

In particular, the first contribution consisted in the writing of a tutorial aimed to enable the interested users to approach these methods, making order among the many existing algorithms and packages and providing solutions to the several related computational issues. It dealt with the main steps to follow when a simulation study is carried out, paying attention to: (i) survival data simulation, (ii) model fitting and (iii) performance assessment.

The second contribution was based on the application of survival analysis methods, both statistical models and machine learning algorithms, for analyzing the offensive performance of the National Basketball Association (NBA) players. In particular, variable selection has been performed for determining the main variables associated to the probability of exceeding a given amount of scored points during the post All-Stars game season segment and the time needed for doing it.

Concluding, this thesis proposes to lay the ground for the development of a unified framework able to harmonize the existing fragmented approaches and without computational issues. Moreover, the findings of this thesis suggest that a survival analysis approach can be extended also to new contexts.

Abstract (Italian)

L'argomento principale di questa tesi è l'analisi della sopravvivenza, un insieme di metodi utilizzati negli studi longitudinali in cui l'interesse non è solo nel verificarsi (o meno) di un particolare evento, ma anche nel tempo necessario per osservarlo.

Negli anni sono stati inizialmente proposti dei modelli statistici e, in seguito, sono stati introdotti anche metodi di machine learning per affrontare studi di analisi di sopravvivenza. La prima parte del lavoro fornisce un'introduzione ai concetti di base dell'analisi di sopravvivenza e un'ampia rassegna della letteratura esistente. Nello specifico, particolare attenzione è stata posta sui principali modelli statistici (non parametrici, semiparametrici e parametrici) e, tra i metodi di machine learning, sugli alberi e sulle random forests di sopravvivenza. Per questi metodi sono state descritte le principali proposte introdotte negli ultimi decenni.

Nella seconda parte della tesi sono invece stati riportati i miei contributi di ricerca. Questi lavori si sono concentrati principalmente su due obiettivi: (1) la razionalizzazione in un protocollo unificato dell'approccio computazionale, che ad oggi è basato su diversi pacchetti esistenti con poca documentazione, molti punti ancora oscuri e anche alcuni bug, e (2) l'applicazione di metodi di analisi dei dati di sopravvivenza in un contesto insolito in cui, per quanto ne sappiamo, questo approccio non era mai stato utilizzato.

Nello specifico il primo contributo è consistito nella scrittura di un tutorial volto a permettere a coloro che sono interessati di utilizzare questi metodi, facendo ordine tra i molti pacchetti esistenti e risolvendo i molteplici problemi computazionali presenti. Esso affronta i principali passi da seguire quando si vuole condurre uno studio di simulazione, con particolare attenzione a: (i) simulazione dei dati di sopravvivenza, (ii) adattamento del modello e (iii) valutazione della performance.

Il secondo contributo è invece basato sull'applicazione di metodi di analisi di sopravvivenza, sia modelli statistici che algoritmi di machine learning, per analizzare le prestazioni offensive dei giocatori della National Basketball Association (NBA). In particolare, è stata effettuata una procedura di selezione delle variabili per determinare le principali variabili associate alla probabilità di superare un determinato numero di punti fatti durante la parte di stagione successiva all'All-Stars game e il tempo necessario per farlo.

Concludendo, questa tesi si propone di porre le basi per lo sviluppo di un framework unificato in grado di armonizzare gli approcci frammentati esistenti e privo di errori computazionali. Inoltre, i risultati di questa tesi suggeriscono che un approccio di analisi di sopravvivenza può essere esteso anche a nuovi contesti.

Contents

List of Figures

List of Tables

Introduction

In longitudinal studies the analysis of time-to-event data is usually of interest. To this extent survival analysis aims to analyze the occurrence of a given event and the time needed to experience it. This kind of analysis was firstly introduced for applications in medicine (for studying, for example, disease recurrence or death); however, its use has widespread to several fields, as economics (e.g. for studying the exit from unemployment), finance (e.g. for predicting bankruptcy) and sport analytics (e.g. for studying injuries or drop-out).
The main characteristic of survival data is *censoring*, which occurs when the event of interest has not been observed for a given subject during the observation time (follow-up) for many reasons (e.g. end of the study or loss to follow-up). So, for these subjects only partial data is available and the only known thing is the last timepoint on which they were observed.

In the literature, statistical models for analyzing censored data were firstly introduced; then, over the years machine learning methods have also been proposed.

The first part of this work provides an extensive review of the existing literature, devoting also particular attention to the alignment of the notation used in the different papers. The second part of this work, instead, concerns my research contributions, namely:

1. The rationalization of the computational approach into a unified protocol. The need of such an harmonization is due to the existence of several packages with few documentation, several still obscure points and also some bugs.

2. The investigation of some empirical case studies in a context where, to our best knowledge, a survival data analysis approach had never been used.

In details, the thesis begins with an introduction of the basic concepts related to survival analysis. Following, Chapter 2 reports the main methodological details of statistical models (nonparametric, semiparametric and parametric) proposed in this context. Among these, the Kaplan-Meier and Cox regression are two of the most used models when no assumption on survival times is made; on the other side, if a particular distribution can be assumed, parametric models as the Accelerated Failure Time model are preferable.
Then, Chapter 3 introduces many of the proposed algorithms for survival trees and random survival forests, an extension of the classical frameworks of Classification and Regression Trees (CART) [Breiman et al., 1984] and random forests (RF) [Breiman, 2001a] to censored data. In more details, for doing that, the splitting criterion (and consequently the pruning algorithm - for trees) had to be changed involving, for example, more specific measures for survival data, as the log-rank test [Peto and

Peto, 1972]. Among the noteworthy proposals for survival trees, there are relative risk trees [LeBlanc and Crowley, 1992], conditional inference trees [Hothorn et al., 2006b, Kundu and Ghosh, 2021] and model-based algorithms [Zeileis et al., 2008]. Among random survival forests the most commonly used are the ones proposed by Hothorn et al. [2006a] and Ishwaran et al. [2008]. In Chapter 3 an introduction to the original algorithms (CART and RF) is also supplied.
Finally, the first part of the thesis ends with Chapter 4 which presents the main measures used to evaluate models' performance.

Despite several proposals about survival trees and random survival forests have been presented over the years, none of these is recognized as the best one. For this reason further simulation studies should be done for better understanding which algorithm is better to use in different situations. Moreover, approaching these methods - in particular survival trees - it resulted straightforward that there are still a lot of unsolved problems concerning several aspects. Among these, the most problematic was the performance assessment: even if there are many proposed measures, their usefulness is still not clear [Zeileis et al., 2008]. Therefore, it is still difficult to compare the performance of different methods. In addition, during these research years, when using these methods from a practical point of view, several computational issues and bugs emerged. These issues were often not easily solvable also because little documentation concerning the corresponding *R* [R Core Team, 2021] functions was available.
Thus, the approach to these methods, still not harmonized due to the many existing proposals, is nowadays difficult. This overall lack of clarity has been also confirmed by the absence, to the best of our knowledge, both in literature and on the web, of guides and suggestions about how to deal with these methods in *R*. For this reason, one of the main aims of this work was to shed light on the topic, making order among the existing algorithms and *R* functions. To this extent, most of the effort was devoted to the fix of computational problems existing among the packages dealing with the fitting of survival trees and their performance evaluation, in order to make the available methods usable.

A list of the main problems has been firstly presented to the 51st Scientific Meeting of the Italian Statistical Society (SIS) and published in the conference proceedings (Macis [2022b], reported in Chapter 5, Section 5.1). As already hinted, to the best of our knowledge, it is very difficult to find suggestions or guides about the practical use of these methods. For this reason, a practical tutorial dealing with the use of these methods was thought to be a valuable reference point for all the users who want to approach this kind of analysis. Therefore, I wrote a paper [Macis, 2022a] aimed to provide a tutorial on how (i) simulating censored data; (ii) fitting survival trees considering three different algorithms (implemented with `rpart`, `ctree` and `SurvCART` functions); and (iii) evaluating trees' performance through many indexes. In particular, the outline of the paper included, beyond a theoretical introduction to the analyzed methods, a detailed description of the main aspects one should consider in these three steps. Moreover a review of the main existing packages has been provided. Then, the main issues and bugs faced in these three steps (mostly in the performance assessment) have been described in detail, together with the possible solutions I propose in my thesis. Finally, to enable the users to implement these methods and interpret the obtained results an example has been reported.

This work is reported in Section 5.2. In addition, a web page (`https://bodai.unibs.it/ml4sd/`) has been created with the aim of providing supplementary materials and updates about the topic. Also, we are still working in the construction of a new *R* package implementing the solutions found (some of them already presented in the tutorial paper). Updates on its state of art will be provided in GitHub and in the above-mentioned web page.

Finally, the last part of the work focused on the application of statistical models and machine learning algorithms for survival analysis in basketball analytics, carried out within the BDsports project (`https://bodai.unibs.it/bdsports/`). The novelty of this contribution stands in the examined aim; indeed, although survival analysis is widely used in sport analytics for studying injuries or drop-out, up to now it has not been used, to the best of our knowledge, for evaluating athletes' performance. These contributions are shown in Chapters 6 and 7.
More in detail, the aim was to evaluate the offensive performance of the National Basketball Association (NBA) players and identify the main achievements associated to the probability of exceeding a given amount of points during the second part of the 2020-2021 regular season. The works presented in the former chapter [Macis et al., 2022a,b] are based on the use of statistical models, in particular Cox regression and LASSO Cox. These works have been presented at the AUEB Sports Analytics Workshop 2022 (in Athens) and at the 13th World Congress of Performance Analysis of Sport 2022 & 13th International Symposium on Computer Science in Sport 2022 (in Wien). Finally, the work shown in the latter chapter is, instead, a work in progress (still not published) based on the application of survival trees and random survival forests for taking interactions and nonlinear effects into account. This work is accepted for presentation at the CMStatistics 2022 in London in the invited organized session *Sport analytics*.

Concluding, this thesis proposes to lay the ground for the development of a unified framework able to overcome and harmonize the existing fragmented approaches and without all the computational issues that trouble the existing *R* packages. For this reason, the writing of a tutorial for users who want to approach these methods, in particular survival trees, is very useful to guide them into this fragmented field. Moreover, applications in unusual contexts have been carried-out for extending the use of these methods in several contexts. The findings of this thesis show that survival analysis is a field in which a variety of research directions can be explored and novel methodological proposals could be provided.

PART I

Literature Review

1 Basic Concepts

Survival analysis was firstly developed in the XVII century [Camilleri, 2019]. It refers to the analysis of the occurrence of a particular event of interest (or endpoint) during a given observed period of time. Therefore, survival analysis is performed when, in a longitudinal study, a set of subjects is monitored to see if and when an event is going to occur. A particular feature of survival data is *censoring*. Censoring is related to the fact that for a certain individual the endpoint of interest cannot be observed during the observation time for many reasons, as the end of the study or the loss of the subject to follow-up (due, for example, to his/her transfer to another country). Therefore, the only known thing about censored subjects is the information we have about his/her survival until a certain timepoint. Then, only partial data about these subjects is available: the only attainable information on the survival experience of these subjects is the last timepoint on which they were observed; beyond that point nothing is known about the event of interest. In these situations, the individual who entered the study at time t_0 will have experienced the event at $t_0 + t$, where t is unknown. Supposing the individual was last known to have not experienced the event of interest at time $t_0 + c$, the time c is called a *censored survival time*. Different kinds of censoring can be identified: right, left and interval. An observation is said *right censored* if his/her censored survival time is less than the actual unknown survival time; on the other hand, an observation is *left censored* when the actual survival time is less than that observed (e.g. during a medical examination a cancer is detected; the time to start of the cancer is less than the time of the examination). Finally, an observation is said *interval censored* if individuals are known to have experienced the event within an interval of time (e.g. considering again the example of cancer identification, it could also be known that three months before the medical examination - t_0 - the patient was healthy and that at t_1 the cancer was identified; then the actual time is comprised between t_0 and t_1).

An example of a longitudinal study is represented in Figure 1. The figure represents a set of $N = 10$ subjects observed for 8 years. Blue crosses represent the event of interest and the red circles represent censored observations. More in detail, it can be seen that subjects 1, 5 and 6 exited the study after about 4, 2 and 5 years respectively without experiencing the event of interest at the last observed timepoints. All these subjects are censored. Finally, the remaining subjects (2, 3, 4, 7, 8, 9, 10) experienced the event of interest after about 7, 2, 6, 6, 8, 4 and 8 years respectively.

Treating this problem as a classification one, with the aim of identifying whether the subject is going to have the event ("positive") or not ("negative"), is not recommended for two reasons. The first one is that in this way the time to event of interest cannot be predicted because information about the time of occurrence of the event is not considered; the second one is that treating a censored event as a negative one is

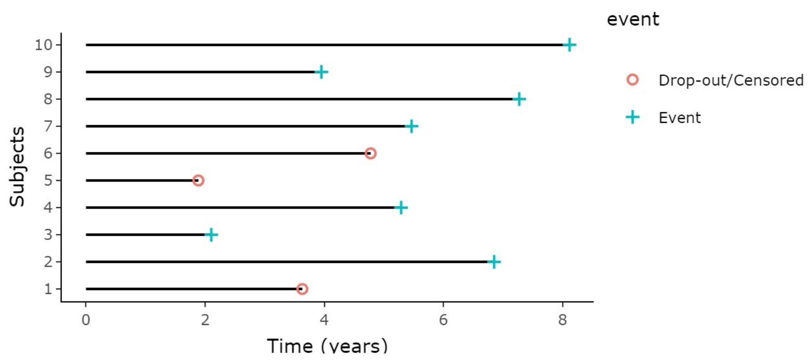


Figure 1: Example of a longitudinal study with censored and uncensored observations

misleading and induces bias in the model (it could happen, for example, that an event occurs for a censored subject few days after he/she left the study). On the other hand, classical regression methods cannot be performed because a regression model requires that the dependent variable (in this case time) has to be observed. Consequently, these methods cannot be used for censored data. Combining classification and regression techniques allows to use the advantages of both methods in order to account for censored data. A survival model (censored regression model), therefore, can be seen as a strengthening of the regression model in which also censored cases provide information.

1.1 Problem Statement and basics of survival analysis

The goal of survival analysis, as already hinted, is to estimate when an event occurs or, in other terms, the survival time of a subject, possibly based on a set of predictors (hereafter also called explanatory variables, features or covariates). The actual survival time t of an individual can be regarded as the observed value of a random variable T that can take any non-negative value. In this setting, a given subject i can be represented by a triplet $(\boldsymbol{x}_i, \tau_i, \delta_i)$. The first element $\boldsymbol{x}_i$ is the vector of observed covariates; τ_i is the observed time for the subject, that corresponds to the survival time for an uncensored subject and to the censor time if the subject is censored; finally, δ_i is a binary event indicator that assumes value 1 if the subject is uncensored and 0 otherwise. In a mathematical form:

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

It follows that for censored subjects the survival time T is latent.

An important assumption that is made is that the actual survival time t_i of an individual, i.e. the period time from the time origin to the occurrence of the event, is independent of any censoring mechanism that causes that individual's survival time is censored at time c_i, where $c_i < t_i$ [Collett, 2015].

Under the assumption that the random variable T has probability density function $f(t)$, the survival function can be defined. The *survival function* $S(t)$ represents the probability that an individual survives beyond a certain time t and can be defined as

$$S(t) = P(T \geq t) = \int_t^{\infty} f(u)du \; .$$

The survival function can also be defined in terms of cumulative density function $F(t)$, that represents the probability that the event of interest occurs earlier than t. In detail,

$$S(t) = 1 - F(t) = 1 - P(T < t) = 1 - \int_0^t f(u)du \; .$$

The density function (Figure 2) can be expressed in terms of survival function as:

$$f(t) = \frac{d}{dt}F(t) = -\frac{d}{dt}S(t) \; .$$

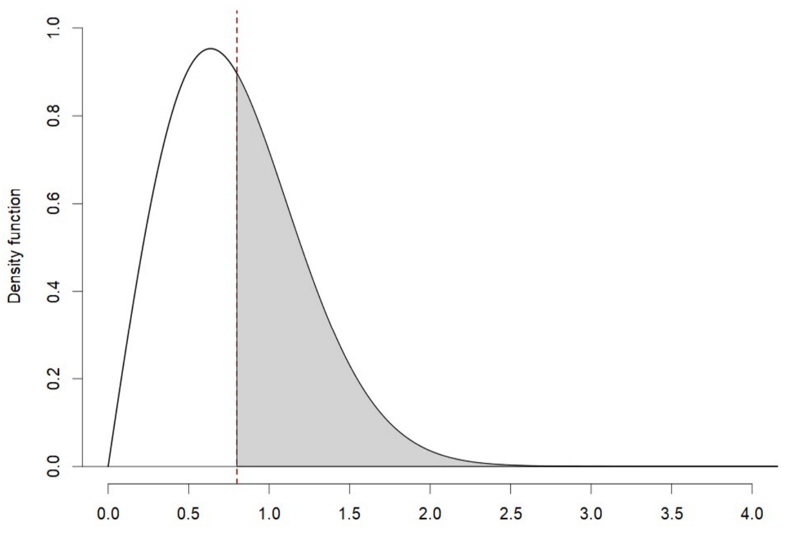


Figure 2: Density function of a random variable T

Another key element of survival analysis is the *hazard function*, that is used to express the risk or hazard that, given survival until time t, the event occurs in the next instant $t + \Delta t$ (with $\Delta t \to 0$). It is obtained from the probability that an individual has the event of interest at time t conditional on the fact that the event has not occurred until that time. Formally, the hazard function is defined as

$$h(t) = \lim_{\Delta t \to 0} \frac{P(t \leq T < t + \Delta t | T \geq t)}{\Delta t} \; ,$$

but can also be expressed in terms of survival function [Collett, 2015] as follows:

$$h(t) = \frac{f(t)}{S(t)} = -\frac{d}{dt} \ln S(t) \; .$$

The hazard function is also known as *hazard rate* or *intensity rate*.

Finally, another important quantity is the *cumulative hazard function*, that is the sum of the risks that the event occurs in the period $[0, t]$:

$$H(t) = \int_0^t h(u)du = -\ln S(t) \ .$$

The survival function and the cumulative hazard function are related by the following expression:

$$S(t) = e^{-H(t)} \ .$$

As reported by Breiman [2001b], there exist two cultures in the use of statistical modeling. Looking at data as generated by a black box that associates predictor variables with the response variables, we can distinguish a data modeling and an algorithmic modeling culture. The first one assumes a stochastic data model for the inside of the black box, and then estimates the parameters and uses the model to obtain information about the studied phenomenon or make a prediction. On the other hand, the second one considers the inside of the box as unknown and the aim is to find a function that allows to predict the outcome variable through the use of the explanatory variables. In this last case the interest is therefore on the predictive accuracy of the model and not on the data modeling [Breiman, 2001b]. On the basis of this distinction, also in survival analysis two main kinds of methods can be distinguished: statistical methods and machine learning methods. The main methods belonging to the two fields are shown in Figure 3.

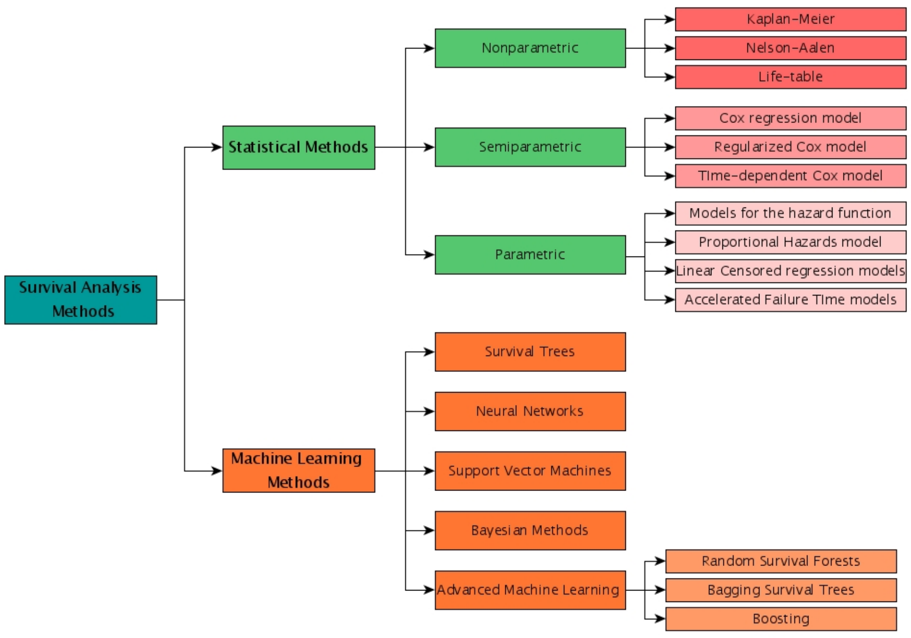


Figure 3: Our elaboration of the taxonomy of Survival Analysis Methods presented in Wang et al. [2019]

2 Statistical Models for Survival Analysis

An initial step in survival analysis is to present numerical and/or graphical summaries of the survival times for individuals in a particular group. These summaries can be provided through estimates of the survival and the hazard functions. The most used methods for this purpose are the *nonparametric models*. Then, in many studies a second step consists in exploring the associations between survival times and other features that are used as covariates. To explore the relationship between the survival experience of an observation and its related features *semiparametric models* are a good choice that extends and unifies the nonparametric procedures. Both these two families of models do not require assumptions on the distribution of survival times. However, in some cases it is of interest to assume a particular probability distribution for survival times. In this case *parametric models* are the most appropriate to use [Collett, 2015].

2.1 Nonparametric methods

Nonparametric methods are the simplest methods used for survival analysis. They are called nonparametric or distribution-free because no specific assumptions about the underlying distribution of the survival times are required. For this reason, they are efficient when no theoretical distributions are known. On the other hand, one of the main disadvantages is that these methods cannot take into account any covariates. They are usually used when the aim is to compare survival times among groups. The three main nonparametric estimation methods are the following:

- Kaplan-Meier estimator;
- Nelson-Aalen estimator;
- Life-table estimator.

These three estimators allow to obtain an estimate of the survival function. Summaries of this function can be then used to describe the survival time of a particular group. Examples are the mean or the median survival time.

2.1.1 Kaplan-Meier estimator

The Kaplan-Meier (K-M) estimator [Kaplan and Paul, 1958] is based on a set of intervals. The main assumptions are three: (i) each interval includes at least one event timepoint; (ii) the event occurs at the beginning of the interval; (iii) in the case of coincidence of an event time and a censored survival time, the last one is assumed to occur immediately after the event time. An example of the intervals considered for the evaluation of this estimator has been reported in Figure 4.
Supposing to have a sample of N individuals with observed survival times $t_1, t_2, \dots, t_N$ and that the event of interest occurs for r subjects ($r \leq N$), the K-M estimator

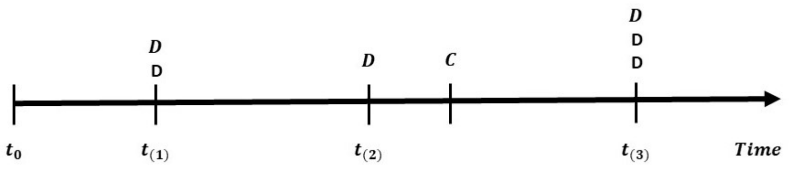


Figure 4: Construction of intervals to obtain the Kaplan-Meier estimate. In particular, $t_{(1)}$, $t_{(2)}$ and $t_{(3)}$ are three timepoints in which an event occurred. "D" represents an event, while "C" is a censored survival time. More in detail, it can be seen that two individuals had the event at $t_{(1)}$, one at $t_{(2)}$ and three a $t_{(3)}$. Finally, a censored subject was observed between $t_{(2)}$ and $t_{(3)}$. Source: Collett [2015]

is evaluated considering the ordered event times $t_{(1)}, t_{(2)}, \ldots, t_{(r)}$. In addition, the following quantities are used to obtain the estimator of interest:

- d_j, that is the number of subjects who experienced the event at $t_{(j)}$;
- n_j, that denotes the *risk set* at $t_{(j)}$, i.e. the set of subjects that just before $t_{(j)}$ had still not experienced the event (including the d_j ones). Due to the assumption that any censored survival time c_j occurring at time $t_{(j)}$ is assumed to occur immediately after the event time, the number of subjects at risk at time $t_{(j)}$ is evaluated as $n_j = n_{j-1} - d_{j-1} - c_{j-1}$.

Assuming that the events of the individuals in the sample occur independently of one another, the estimated survival function at any time t is given by:

$$\hat{S}_{K-M}(t) = \prod_{j=1}^{k} \frac{n_j - d_j}{n_j} \text{ for } t_{(k)} \leq t < t_{(k+1)},\ k = 1, 2, ..., r\ ,$$

with $\hat{S}_{K-M}(t) = 1$ at the beginning of the study (for $t < t_{(1)}$) and where $t_{(r+1)}$ is taken to be equal to ∞. Moreover, if the largest observation t^* is censored, $\hat{S}_{K-M}(t)$ is undefined for $t > t^*$; on the other hand, if the largest observed survival time $t_{(r)}$ is uncensored, then $n_r = d_r$ and $\hat{S}_{K-M}(t) = 0$ for $t \geq t_{(r)}$.
The graphical representation of the K-M estimate of the survival function (Figure 5) is a step function in which the estimated survival probabilities are constant between adjacent event times and decrease at each event time. The more events occur at one timepoint, the higher the step.
It can be shown that the standard error of this survival estimate, known as *Greenwood's formula* [Collett, 2015], is:

$$se(\hat{S}_{K-M}(t)) \approx \hat{S}_{K-M}(t) \left\{ \sum_{j=1}^{k} \frac{d_j}{n_j(n_j - d_j)} \right\}^{1/2} \text{ for } t_{(k)} \leq t < t_{(k+1)},\ k = 1, 2, ..., r\ .$$

Furthermore, the Kaplan-Meier method can also be used to obtain an estimate of the hazard function. A simple estimate is given by the ratio of the number of events at

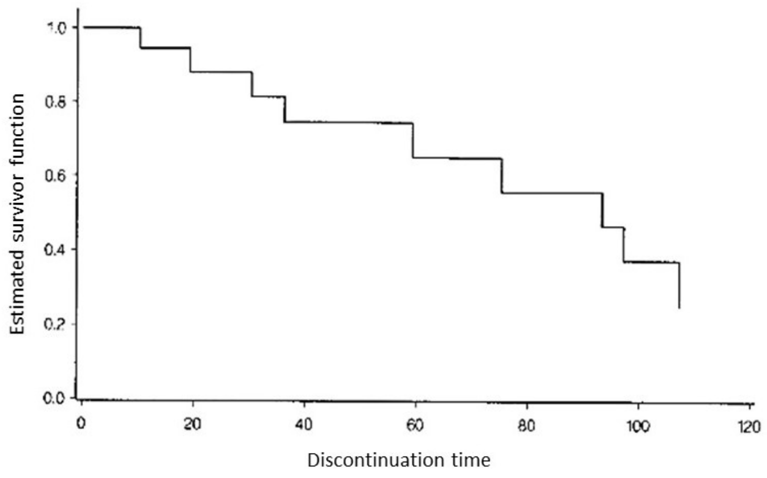


Figure 5: Example of plot of the Kaplan-Meier estimate of the survival function. Source: Collett [2015]

a given timepoint to the number of individuals at risk at that time. Moreover, if the hazard function is assumed to be constant between successive event times, the hazard function can be found by further dividing by the time interval $\gamma_j = t_{(j+1)} - t_{(j)}$. In this case, the hazard function in the interval from $t_{(j)}$ to $t_{(j+1)}$ can be estimated with the following expression:

$$\hat{h}_{K-M}(t) = \frac{d_j}{n_j \gamma_j} \text{ for } t_{(j)} \leq t < t_{(j+1)},\ j = 1, 2, ..., r\ .$$

The standard error of this estimate is given by

$$se(\hat{h}_{K-M}(t)) = \hat{h}_{K-M}(t) \sqrt{\frac{n_j - d_j}{n_j d_j}} \text{ for } t_{(j)} \leq t < t_{(j+1)},\ j = 1, 2, ..., r\ .$$

Finally, the cumulative hazard function can be defined using the K-M estimate of the survival function:

$$\hat{H}_{K-M}(t) = -\ln \hat{S}_{K-M}(t) = -\sum_{j=1}^{k} \ln\left(\frac{n_j - d_j}{n_j}\right) \text{ for } t_{(k)} \leq t < t_{(k+1)},\ k = 1, 2, ..., r\ .$$

A simple example to better understand how the K-M estimator is obtained is shown in Table 1.

2.1.2 Nelson-Aalen estimator

The Nelson-Aalen (N-A) estimator is an alternative statistic that can be used to estimate the survival function on the basis of the individual event times [Nelson, 1969, Aalen, 1978]. It is obtained from the following estimate of the cumulative hazard function

$$\hat{H}_{N-A}(t) = \sum_{j=1}^{k} \frac{d_j}{n_j} \text{ for } t_{(k)} \leq t < t_{(k+1)},\ k = 1, , 2, ..., r\ ,$$

Table 1: Example of evaluation of the Kaplan-Meier estimator. The interval $t_k \vdash t_{k+1}$ begins at time $t_{(k)}$ and ends just before $t_{(k+1)}$, $k = 1, 2, ..., r$. Source: Collett [2015]

Time interval	γ_j	n_j	c_j	d_j	$\frac{n_j - d_j}{n_j}$	$\hat{S}_{K-M}(t)$	$\hat{h}_{K-M}(t)$	$\hat{H}_{K-M}(t)$
$0 \vdash 10$	10	18	0	0	1.0000	1.0000	0.0000	0.0000
$10 \vdash 19$	9	18	2	1	0.9444	0.9444	0.0062	0.0572
$19 \vdash 30$	11	15	1	1	0.9333	0.8815	0.0061	0.1261
$30 \vdash 36$	6	13	0	1	0.9231	0.8137	0.0128	0.2062
$36 \vdash 59$	23	12	3	1	0.9167	0.7459	0.0036	0.2932
$59 \vdash 75$	16	8	0	1	0.8750	0.6527	0.0078	0.4266
$75 \vdash 93$	18	7	0	1	0.8571	0.5594	0.0079	0.5809
$93 \vdash 97$	4	6	0	1	0.8333	0.4661	0.00417	0.7634
$97 \vdash 107$	10	5	1	1	0.8000	0.3729	0.0200	0.9864
$107 \vdash$	$---$	3	2	1	0.6667	0.2486	$---$	1.3919

that is the cumulative sum of the estimated probabilities of event from the first to the j^{th} time interval. The Nelson-Aalen estimate of the survival function is then

$$\hat{S}_{N-A}(t) = \prod_{j=1}^{k} e^{-\frac{d_j}{n_j}} \text{ for } t_{(k)} \leq t < t_{(k+1)},\ k = 1, 2, ..., r\ ,$$

and its standard error is the following:

$$se(\hat{S}_{N-A}(t)) \approx \hat{S}_{N-A}(t) \left\{ \sum_{j=1}^{k} \frac{d_j}{n_j^2} \right\}^{1/2} .$$

It can be shown that the N-A estimate of the survival function is always greater than the K-M estimate at any given time. Indeed, each term of the product of the Nelson-Aalen estimate can be written as

$$e^{-x} = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \dots\ ,$$

where $x = \frac{d_j}{n_j}$. Thus, this expression is greater or at least equal to $1 - x$ (that corresponds to a generic term of the product of the K-M estimate) for all values of x. However, the K-M estimate can be seen as an approximation of the N-A estimate when x is small, that is when d_j is small relative to n_j. Moreover, the two estimators are asymptotically equivalent and the differences between them are small, particularly at the earlier survival times [Wang et al., 2019].

Although the N-A estimate has been shown to perform better than the K-M estimate in small samples, the last one is usually preferred because it can be seen as a generalization of the empirical survival function [1] when censored data are considered [Collett, 2015].

An estimate of the hazard function can be obtained by evaluating the differences between adjacent values of the estimated cumulative hazard function and dividing these quantities by the time interval, that is

$$\hat{h}_{N-A}(t_{(j)}) = \frac{\hat{H}_{N-A}(t_{(j)}) - \hat{H}_{N-A}(t_{(j-1)})}{\gamma_j} = \frac{d_j}{n_j \gamma_j},\ j = 1, ..., r\ .$$

Thus, the Nelson-Aalen estimate of the hazard function coincides with the hazard function estimate obtained using the Kaplan-Meier estimator. In the table below an example of the evaluation of the N-A estimator has been provided.

Table 2: Example of evaluation of the Nelson-Aalen estimator. The interval $t_k \vdash t_{k+1}$ begins at time $t_{(k)}$ and ends just before $t_{(k+1)}$, $k = 1, 2, ..., r$. Source: Collett [2015]

Time interval	γ_j	n_j	c_j	d_j	$\frac{d_j}{n_j}$	$e^{-\frac{d_j}{n_j}}$	$\hat{S}_{N-A}(t)$	$\hat{h}_{N-A}(t)$	$\hat{H}_{N-A}(t)$
$0 \vdash 10$	10	18	0	0	0.0000	1.0000	1.0000	0.0000	0.0000
$10 \vdash 19$	9	18	2	1	0.0556	0.9459	0.9459	0.0556	0.0062
$19 \vdash 30$	11	15	1	1	0.0667	0.9355	0.8849	0.1223	0.0061
$30 \vdash 36$	6	13	0	1	0.0769	0.9260	0.8194	0.1992	0.0128
$36 \vdash 59$	23	12	3	1	0.0833	0.9201	0.7539	0.2825	0.0036
$59 \vdash 75$	16	8	0	1	0.1250	0.8825	0.6653	0.4075	0.0078
$75 \vdash 93$	18	7	0	1	0.1429	0.8668	0.5767	0.5504	0.0079
$93 \vdash 97$	4	6	0	1	0.1667	0.8465	0.4882	0.7171	0.0417
$97 \vdash 107$	10	5	1	1	0.2000	0.8187	0.3997	0.9171	0.0200
$107 \vdash$	–	3	2	1	0.3333	0.7166	0.2864	1.2504	–

2.1.3 Life-Table estimator

The life-table (L-T) estimator, also known as the actuarial estimator, is used when there are grouped survival data. This method is useful when there are many observations or when the actual event times are unknown and the only available information is the number of observed events and censored observations that occurred in a series of consecutive time intervals. The L-T estimator can be seen as an approximation of the K-M estimator: instead of taking into account every timepoint in which an event occurs, a given interval is considered (e.g. half a year). More in detail,

[1] When there are no censored observations the empirical survival function can be evaluated as

$$\hat{S}(t) = \frac{\text{N° of individuals with survival times } \geq t}{\text{N° of individuals in the sample}}$$

the entire observation time period is divided in m intervals and the censored survival times are assumed to occur uniformly throughout each interval. As a consequence, the average number of individuals who are at risk during the j^{th} interval is $n'_j = n_j - \frac{c_j}{2}$, where n_j is the number of subjects at risk at the start of the j^{th} interval and c_j is the number of censored survival times in the j^{th} interval, that extends from time t'_j to t'_{j+1}.

Denoting with d_j the number of events occurred in the j^{th} interval, the probability that the event of interest occurs can be estimated as $\frac{d_j}{n_j}$ and the corresponding survival probability is $\frac{n'_j - d_j}{n'_j}$.
Then, the probability that an individual survives beyond time t'_j, $j = 1, 2, ..., m$, i.e. until some time after the start of the j^{th} interval, can be evaluated as the product of the probabilities that an individual survives at each of the j preceding intervals. Then, the L-T estimate of the survival function can be obtained as:

$$\hat{S}_{L-T}(t) = \prod_{j=1}^{k} \left(\frac{n'_j - d_j}{n'_j} \right) \text{ for } t'_k \leq t \leq t'_{k+1} \; k = 1, 2, ..., m \; .$$

The estimated survival probability until the start of the first interval t'_1 is equal to 1, while the estimated probability of surviving beyond t'_{m+1} is equal to zero.
The standard error of the survival estimate can be evaluated as

$$se(\hat{S}_{L-T}(t)) = \hat{S}_{L-T}(t) \left\{ \sum_{j=1}^{k} \frac{d_j}{n'_j(n'_j - d_j)} \right\}^{\frac{1}{2}} .$$

The graphical representation of the survival function is then a step-function with constant values in each time-interval and whose shape is sensitive to the choice about the width of the intervals.
It can be seen that the K-M estimate is the limiting value of the L-T estimate as the number of intervals tends to infinity and their width tends to zero. For this reason, it is also known as the product-limit estimate of the survival function.

The life-table estimate of the hazard function in the j^{th} time interval is given by

$$\hat{h}_{L-T}(t) = \frac{d_j}{(n'_j - \frac{d_j}{2})\gamma_j} \text{ for } t'_j \leq t < t'_{j+1}, \; j = 1, 2, ..., m \; ,$$

that represents the average hazard per unit time over each interval. It corresponds to the observed number of events in that interval, divided by the average survived time in the j^{th} interval, assuming that the event rate is constant during the interval.
Its standard error is

$$se(\hat{h}_{L-T}(t)) = \frac{\hat{h}_{L-T}(t) \sqrt{1 - \left(\frac{\hat{h}_{L-T}(t)\gamma_j}{2} \right)^2}}{\sqrt{d_j}} .$$

Finally, the estimated cumulative hazard function can be obtained as

$$\hat{H}_{L-T}(t) = -\ln \hat{S}_{L-T}(t) = -\sum_{j=1}^{k} \ln\left(\frac{n'_j - d_j}{n'_j}\right) \text{ for } t'_k \le t < t'_{k+1},\ k = 1, 2, ..., m\ .$$

An example has been provided in Table 3.

Table 3: Example of evaluation of the life-table estimator. The interval $t_k \vdash t_{k+1}$ begins at time $t_{(k)}$ and ends just before $t_{(k+1)}$, $k = 1, 2, ..., r$. Source: Collett [2015]

Time period	γ_j	n_j	c_j	d_j	n'_j	$\frac{n'_j - d_j}{n'_j}$	$\hat{S}_{L-T}(t)$	$\hat{h}_{L-T}(t)$	$\hat{H}_{L-T}(t)$
$0 \vdash 12$	12	48	4	16	46.0	0.6522	0.6522	0.0351	0.4274
$12 \vdash 24$	12	28	4	10	26.0	0.6154	0.4014	0.0397	0.9128
$24 \vdash 36$	12	14	0	1	14.0	0.9286	0.3727	0.0062	0.9870
$36 \vdash 48$	12	13	1	3	12.5	0.7600	0.2833	0.0227	1.2612
$48 \vdash 60$	12	9	2	2	8.0	0.7500	0.2124	0.0238	1.5493
$60 \vdash 96$	36	5	1	4	4.5	0.1111	0.0236	0.0444	3.7465

2.1.4 Comparison of groups of survival data

Nonparametric methods, as already said previously, are usually used to compare survival times of two or more groups. There are many methods that allow to make these comparisons. The most commonly used are reported below.

2.1.4.1 Comparison of two groups

The most used comparison method for two groups is the *log-rank test* [Peto and Peto, 1972]. It is a test statistic used to verify the null hypothesis of no difference in the survival experiences of the individuals in the two groups. It is based on the difference between the observed and expected number of events under the null hypothesis in the two groups at each timepoint. Complete information about these differences is then obtained combining all the event times, providing an overall measure of the deviation of the observed events from the respective expected values. More in detail, the test statistic is obtained by combining the information retrieved from the individual 2×2 tables for each of the r timepoints (see Table 4).

Considering the marginal totals as fixed, and assuming that the survival is independent of the two groups, the four entries of the table are solely determined by the value of d_{1j}. The number of events in group 1 at time j can then be seen as a random variable with hypergeometric distribution. Thus, the expected number of individuals who experience the event at time $t_{(j)}$ in group 1, can be obtained by considering the mean

Table 4: Example of contingency table for the evaluation of the log-rank test. Contingency table between Group and Status (survived or event) at $t_{(j)}$

Group	**Events at** $t_{(j)}$	**Surviving beyond** $t_{(j)}$	**At risk just before** $t_{(j)}$
1	d_{1j}	$n_{1j} - d_{1j}$	n_{1j}
2	d_{2j}	$n_{2j} - d_{2j}$	n_{2j}
Total	d_j	$n_j - d_j$	n_j

of the hypergeometric random variable, that is:

$$e_{1j} = \frac{n_{1j} d_j}{n_j} \ ,$$

where n_{1j} and n_j are respectively the number of subjects at risk in group 1 and in all the sample (both in group 1 and group 2) at time $t_{(j)}$.
The log-rank test statistic is then evaluated as

$$W_L = \frac{U_L^2}{V_L} \sim \chi_1^2 \ ,$$

where U_L is the difference between the total observed and expected number of events in group 1

$$U_L = \sum_{j=1}^{r} (d_{1j} - e_{1j})$$

and V_L is the variance of U_L, that is the sum of variances of the d_{1j}:

$$V_L = \sum_{j=1}^{r} v_{1j} = \sum_{j=1}^{r} \frac{n_{1j} n_{2j} d_j (n_j - d_j)}{n_j^2 (n_j - 1)} \ .$$

The larger the value of the statistic W_L, the greater the evidence against the null hypothesis.

Another popular test statistic to compare the survival times of two groups is the *Wilcoxon test*, also known as the *Breslow test* [Peto and Peto, 1972]. The null hypothesis is the same used for the log-rank test, that is the absence of differences in the survival functions of the two groups. The difference with the log-rank test is that the Wilcoxon test weights the difference between the observed and expected number of events with the correspondent size of the risk set. In this way less weight is given to the difference between d_{1j} and e_{1j} at the timepoints in which the total number of individuals at risk is small, i.e. at the longest survival times. Thus, the Wilcoxon test is less sensitive to deviations of the observed events from the expected ones in the tail of the distribution of survival times [Collett, 2015]. More in detail, it is evaluated as follows:

$$W_W = \frac{U_W^2}{V_W} \sim \chi_1^2 \ ,$$

where

$$U_W = \sum_{j=1}^{r} n_j (d_{1j} - e_{1j}) ,$$

and

$$V_W = \sum_{j=1}^{r} n_j^2 v_{1j} = \sum_{j=1}^{r} n_j^2 \frac{n_{1j} n_{2j} d_j (n_j - d_j)}{n_j^2 (n_j - 1)} .$$

The log-rank test is preferred to the Wilcoxon test when the alternative hypothesis is the proportional hazards assumption, i.e. that the hazard of experiencing the event at any given time for an individual in one group is proportional to the hazard at that time for a similar individual in the other group. In all the other cases the Wilcoxon test is more appropriate.

2.1.4.2 General case: comparison of g groups

The log-rank and the Wilcoxon tests can also be extended to the case of three or more groups. In order to compare the survival times of g groups ($g \geq 2$), the following U-statistics for the log-rank and the Wilcoxon test respectively can be evaluated, considering the difference between observed number of events in the groups $1, 2, ..., g-1$ and their expected values:

$$U_{Lk} = \sum_{j=1}^{r} \left(d_{kj} - \frac{n_{kj} d_j}{n_j} \right) \text{ for } k = 1, 2, ..., g-1 ;$$

$$U_{Wk} = \sum_{j=1}^{r} n_j \left(d_{kj} - \frac{n_{kj} d_j}{n_j} \right) \text{ for } k = 1, 2, ..., g-1 .$$

These quantities can then expressed in the form of a vector with $g-1$ components, $\boldsymbol{U_L}$ and $\boldsymbol{U_W}$. Finally, the variance-covariance matrix is needed. The variance-covariance matrix for $\boldsymbol{U_L}$ is then

$$\begin{pmatrix}
V_{L11} & V_{L12} & \dots & V_{L1k} & V_{L1k'} & \dots & V_{L1(g-1)} \\
V_{L21} & V_{L22} & \dots & V_{L2k} & V_{L2k'} & \dots & V_{L2(g-1)} \\
\vdots & \dots & \ddots & \vdots & \dots & \dots & \vdots \\
V_{Lk1} & V_{Lk2} & \dots & V_{Lkk} & V_{Lkk'} & \dots & V_{Lk(g-1)} \\
V_{Lk'1} & V_{Lk'2} & \dots & V_{Lk'k} & V_{Lk'k'} & \dots & V_{Lk'(g-1)} \\
\vdots & \dots & \ddots & \vdots & \dots & \dots & \vdots \\
V_{L(g-1)1} & V_{L(g-1)2} & \dots & V_{L(g-1)k} & V_{L(g-1)k'} & \dots & V_{L(g-1)(g-1)}
\end{pmatrix}$$

where the generic elements V_{Lkk} and $V_{Lkk'}$ are respectively the variance of U_{Lk} and the covariance of U_{Lk} and $U_{Lk'}$.
For the log-rank test the covariance between U_{Lk} and $U_{Lk'}$ is given by

$$V_{Lkk'} = \sum_{j=1}^{r} \frac{n_{kj} d_j (n_j - d_j)}{n_j (n_j - 1)} \left(\delta_{kk'} - \frac{n_{k'j}}{n_j} \right) \text{ for } k, k' = 1, 2, ..., g-1 ,$$

where $\delta_{kk'} = 1$ if $k = k'$ and 0 otherwise.

In the same way the variance-covariance matrix for $\boldsymbol{U_W}$ can be obtained. The generic element $V_{Wkk'}$ of the matrix for the Wilcoxon test is given by:

$$V_{Lkk'} = \sum_{j=1}^{r} n_j^2 \frac{n_{kj} d_j (n_j - d_j)}{n_j(n_j - 1)} \left(\delta_{kk'} - \frac{n_{k'j}}{n_j} \right) \text{ for } k, k' = 1, 2, ..., g - 1 \; .$$

Finally, in order to test the null hypothesis of no group differences the test statistics for the log-rank test and the Wilcoxon test are respectively

$$\boldsymbol{U'_L V_L^{-1} U_L}$$

and

$$\boldsymbol{U'_W V_W^{-1} U_W} \; .$$

Under the null hypothesis both the two statistics have a Chi-squared distribution with $g - 1$ degrees of freedom.

2.2 Semiparametric models

As already hinted, it is often interesting to study the relationship between survival and one or more explanatory variables; in this case, the focus will be on the risk/hazard that the event occurs at any time after the time origin of the study. For this purpose, semiparametric models are a good choice that allows to directly model the hazard function. In this context the main aims are two: (i) to determine which combination of explanatory variables influences the hazard function; (ii) to obtain an estimate of the hazard function of an individual with a particular set of features. The most popular semiparametric model is the *Cox regression model* introduced by Sir David Cox in 1972 [Cox, 1972]. Over the years, different extensions as the *regularized Cox models* and the *time-dependent Cox model* have also been introduced. In particular, the first ones have been proposed to analyze sparse data; differently, the second one has been introduced to take into account explanatory variables that change over time (time-dependent variables).

2.2.1 Cox regression model

The Cox regression model allows to estimate, on the basis of a vector of explanatory variables $\boldsymbol{X}$, the hazard function, i.e. the conditional probability that a subject has an event at a particular timepoint given that he/she has survived until now. The main assumption of the model is the proportionality of hazards (PH). Supposing to have two groups with hazard functions at time t equal to $h_1(t)$ and $h_2(t)$, the proportional hazards (PH) assumption can be expressed as

$$\psi = \frac{h_1(t)}{h_2(t)} \; \forall t \geq 0 \; ,$$

or analogously as

$$h_1(t) = \psi h_2(t) \ \forall t \geq 0 \ .$$

The constant ψ, called *relative hazard* or *hazard ratio*, is a measure that allows to compare two individuals belonging to different groups in terms of hazard of event. If $\psi < 1$ the hazard of event at t is smaller for an individual in the first group, relative to an individual in the second one; if $\psi > 1$ the vice-versa holds. Finally, if $\psi = 1$ the hazard of event at t is equal for the two groups. As a consequence of the PH assumption, the hazard ratio for two given groups is constant over time, or, in other words, it is independent of time. This assumption implies that the true survival functions for the two groups do not cross each other. A graphical approach that can be used to check the validity of the PH assumption is to plot the log-cumulative hazard curves $\ln(-\ln \hat{S}(t))$ for the two groups against the logarithm of survival time and to compare them. If the two curves are parallel, the proportional hazards assumption is valid [Collett, 2015]. More generally, the PH assumption can be generalized to the case in which the hazard of the event depends on the values $\boldsymbol{x} = (x_1, x_2, ..., x_p)'$ of p explanatory variables $\boldsymbol{X} = (X_1, X_2, ..., X_p)'$. In this case, and in particular for the Cox regression model, the hazard ratio can be expressed as a function of $\boldsymbol{x_i}$ the vector of observed features for the i^{th} individual:

$$\psi(\boldsymbol{x}_i) = e^{\eta_i} = e^{\beta_1 x_{1i} + ... + \beta_p x_{pi}} = e^{\sum_{k=1}^{p} \beta_k x_{ki}} \ .$$

The term η_i is the linear component of the model and it is also known as *risk score* or *prognostic index* for the i^{th} individual.
The Cox regression model is then formulated in terms of hazard function:

$$h_i(t) = h_0(t) e^{\eta_i} = h_0(t) e^{\beta_1 x_{1i} + ... + \beta_p x_{pi}} = h_0(t) e^{\sum_{k=1}^{p} \beta_k x_{ki}} \ .$$

The above expression represents the hazard of experiencing the event of interest at a particular time for the i^{th} individual characterized by a set of p explanatory variables $\boldsymbol{x}_i = (x_{1i}, x_{2i}, ..., x_{pi})$. The term $h_0(t)$ is the *baseline hazard function*, that can be any non-negative function of time. It corresponds to the hazard function for an individual whose values of all the independent variables are equal to zero. Therefore, the term $\psi(\boldsymbol{x}_i) = e^{\sum_{k=1}^{p} \beta_k x_{ki}}$ can be interpreted as the hazard at time t for the i^{th} subject relative to the baseline hazard. Finally, each β coefficient provides a measure of the effect of the respective covariate on the hazard of event. These parameters correspond to the logarithm of the hazard ratio. More in detail, if X_k is a continuous explanatory variable, the related coefficient β_k represents the estimated change in the logarithm of the hazard ratio as the value of X_k is increased by one unit. On the other hand, when X_k is categorical, β_k is the estimated logarithm of the relative hazard for an individual in group k relative to an individual in the reference group. Usually, for easiness of interpretation, the e^{β_k} coefficients, that correspond to the hazard ratios, are considered. When a model contains more than one variable, the parameter estimate associated with a particular effect is said to be adjusted for the other variables in the model, and so the estimates are adjusted for the other terms in the model.

Cox regression model is referred as semiparametric because although based on the proportional hazards assumption, no assumptions are made on the distribution of survival times. It can be seen that it combines a nonparametric and a parametric component. The nonparametric term is the baseline hazard function, because no particular assumption for the probability distribution of the survival times is made; the parametric component is instead related to the features vector.
If the main aim of the study is to compare subjects in terms of risk, then only the parametric component can be considered and it is not needed to estimate the baseline hazard function (that is common to all subjects). If, instead, the interest is on estimating the probability of occurrence of the event at a given timepoint, the baseline hazard function has to be estimated too.

2.2.1.1 Estimation of the linear component of the model (β parameters)

The first step of model fitting is to estimate the parameters $\boldsymbol{\beta} = (\beta_1, \beta_2, ..., \beta_p)'$, the second one is to build an estimate of the baseline hazard function $h_0(t)$ using the parameters estimated previously. The model parameters are estimated by Maximum Likelihood.
Assuming to have a sample of N subjects and to observe r events ($r \leq N$) during the entire observation period, the partial likelihood can be considered:

$$L(\boldsymbol{\beta}) = \prod_{j=1}^{r} \frac{e^{\boldsymbol{\beta}'\boldsymbol{x}_{(j)}}}{\sum_{l \in R_j} e^{\boldsymbol{\beta}'\boldsymbol{x}_l}} ,$$

where $\boldsymbol{x}_{(j)}$ is the vector of the observed covariates for the observation who experienced the event at $t_{(j)}$ and R_j is the risk set at $t_{(j)}$.
The partial likelihood function can also be expressed as

$$L(\boldsymbol{\beta}) = \prod_{i=1}^{N} \left[\frac{e^{\boldsymbol{\beta}'\boldsymbol{x}_i}}{\sum_{l \in R_j} e^{\boldsymbol{\beta}'\boldsymbol{x}_l}} \right]^{\delta_i} ,$$

where $\boldsymbol{x}_i$ is the observed covariate vector for the i^{th} subject who experienced the event at $t_{(j)}$. As usual, δ_i is the event indicator; it can be noticed that only uncensored subjects ($\delta_i = 1$) have an effect on the above product. On the contrary, censored observations do not directly contribute to the likelihood (when $\delta_i = 0$ the corresponding term is equal to 1). However, these observations take part in the likelihood function indirectly through the denominator, that includes all the subjects in the risk set.
The log-likelihood function

$$\ln L(\boldsymbol{\beta}) = l(\beta) = \sum_{i=1}^{N} \delta_i \left[\boldsymbol{\beta}'\boldsymbol{x}_i - \ln \left(\sum_{l \in R_j} e^{\boldsymbol{\beta}'\boldsymbol{x}_l} \right) \right]$$

is then maximized using iterative methods as the Newton-Raphson algorithm.
It is important to notice that these likelihood and log-likelihood functions cannot be used in the presence of ties because, given the continuity of the hazard function, tied survival times are not admissible. Many proposals have been made to approximate the likelihood function in this case [Collett, 2015]. Among these there is that of Breslow [Breslow, 1974] who suggested to use the following approximation based on the assumption of distinct events that occur sequentially:

$$\prod_{j=1}^{r} \frac{e^{\boldsymbol{\beta}' \boldsymbol{s}_j}}{[\sum_{l \in R_j} e^{\boldsymbol{\beta}' \boldsymbol{x}_l}]^{d_j}} ,$$

where $\boldsymbol{s_j}$ is the vector of sums of each covariate for all the d_j individuals who experienced the event at $t_{(j)}$. The generic element of this vector is $s_{kj} = \sum_{e=1}^{d_j} x_{kje}$, with x_{kje} the value of the k^{th} explanatory variable for the e^{th} of the d_j subjects.
This approximation is adequate when the number of tied observations is not too large.

2.2.1.2 Choice of the best model

Another important step in a modeling approach is to determine the best model. Usually, different models with different sets of explanatory variables are fitted, in order to find the best one in terms of fit. When comparing nested models, i.e. models in which one of the two contains a subset of the terms included in the other, the log-likelihood ratio statistic can be used:

$$-2 \ln \frac{\hat{L_1}}{\hat{L_2}} ,$$

where $\hat{L_1}$ and $\hat{L_2}$ are the values of the likelihood function when the parameters are replaced by their maximum likelihood estimates. This test statistic has an asymptotic chi-squared distribution under the null hypothesis of likelihood ratio equal to 1, with number of degrees of freedom equal to the difference between the number of independent β parameters being fitted under the two models. Therefore, the aim of this test is to determine whether the additional terms in the larger model significantly improve its explanatory power.
An alternative that can also be used to compare models which are not nested, is the Akaike Information Criterion (AIC)

$$AIC = -2 \ln \hat{L} + 2q ,$$

where q is the number of unknown parameters in the model. The smaller the value the better the model.
These two methods can be used when the number of potential explanatory variables is not too large and it might be feasible to fit all possible combination of terms. When

this is not possible, some automatic routines for variable selection can be used, based on forward selection, backward elimination or on a combination of the two, known as stepwise procedures. Other details about the variables selection procedures can be found in Collett [2015].

2.2.1.3 Estimation of the hazard and survival functions

The estimation of the β parameters is all that is needed when the aim is drawing inferences about the effect of explanatory variables on the hazard function. Once a suitable model has been identified (and therefore the β parameters are estimated), the baseline hazard function can also be estimated and consequently the hazard function and the corresponding cumulative hazard and survival functions can be obtained. In particular, the estimated hazard function is

$$\hat{h}_i(t) = e^{\hat{\boldsymbol{\beta}}'\boldsymbol{x}_i}\hat{h}_0(t) \ ,$$

where $\hat{\boldsymbol{\beta}}'$ is the vector of estimated coefficients and $\hat{h}_0(t)$ is the estimated baseline hazard function. An estimate of the baseline hazard function was derived by Kalbfleisch and Prentice [Kalbfleisch and Prentice, 1973] using an approach based on maximum likelihood estimation. More in detail, the estimated baseline hazard function at time $t_{(j)}$ is given by

$$\hat{h}_0(t_{(j)}) = 1 - \hat{\xi}_j \ ,$$

where $\hat{\xi}_j$ is the solution of the equation

$$\sum_{l \in D_j} \frac{e^{\hat{\boldsymbol{\beta}}'\boldsymbol{x}_l}}{1 - \hat{\xi}_j^{exp\{\hat{\boldsymbol{\beta}}'\boldsymbol{x}_l\}}} = \sum_{l \in R_j} e^{\hat{\boldsymbol{\beta}}'\boldsymbol{x}_l} \text{ for } j = 1, 2, ..., r \ ,$$

where D_j is the set of all d_j individuals who experienced the event at $t_{(j)}$.
When there are tied observations (i.e. more than one event has been observed at $t_{(j)}$, so that $d_j > 1$), the equation cannot be solved explicitly and iterative algorithms are needed. This can be avoided by approximating $\hat{\xi}_j$ with $\tilde{\xi}_j$. The method to do that is to write $\hat{\xi}_j^{exp\{\hat{\boldsymbol{\beta}}'\boldsymbol{x}_l\}}$ as:

$$\hat{\xi}_j^{exp\{\hat{\boldsymbol{\beta}}'\boldsymbol{x}_l\}} = exp\{e^{\hat{\boldsymbol{\beta}}'\boldsymbol{x}_l} \ln \hat{\xi}_j\} \approx 1 + e^{\hat{\boldsymbol{\beta}}'\boldsymbol{x}_l} \ln \tilde{\boldsymbol{\xi}}_j \ .$$

Using the approximation above, the equation becomes

$$-\sum_{l \in D_j} \frac{1}{\ln \tilde{\xi}_j} = \sum_{l \in R_j} e^{\hat{\boldsymbol{\beta}}'\boldsymbol{x}_l}$$

that can also be written, by solving the first summation, as

$$-\frac{d_j}{\ln \tilde{\xi}_j} = \sum_{l \in R_j} e^{\hat{\boldsymbol{\beta}}'\boldsymbol{x}_l} \ .$$

Then, $\tilde{\xi}_j$ can be found:

$$\tilde{\xi}_j = exp\left\{ - \frac{d_j}{\sum_{l \in R_j} e^{\hat{\boldsymbol{\beta}}'\boldsymbol{x}_l}} \right\} .$$

Thus, assuming that the hazard of event is constant between two adjacent event times, the estimated baseline hazard function is

$$\hat{h}_0(t) = \frac{1 - \hat{\xi}_j}{\gamma_j} \approx \frac{1 - e^{-\frac{d_j}{\sum_{l \in R_j} e^{\hat{\boldsymbol{\beta}}'\boldsymbol{x}_l}}}}{\gamma_j} \text{ for } t_{(j)} \leq t < t_{(j+1)} \text{ , } j = 1, 2, ..., r-1 \text{ ,}$$

where $\gamma_j = t_{(j+1)} - t_{(j)}$.
The quantity $\hat{\xi}_j$ (as well as its approximation $\tilde{\xi}_j$) can be seen as an estimate of the probability that an individual survives through the interval beginning at $t_{(j)}$ and ending just before $t_{(j+1)}$. So, the baseline survival function can be obtained as

$$\hat{S}_0(t) = \prod_{j=1}^{k} \hat{\xi}_j \approx \prod_{j=1}^{k} e^{-\frac{d_j}{\sum_{l \in R_j} e^{\hat{\boldsymbol{\beta}}'\boldsymbol{x}_l}}} \text{ for } t_{(k)} \leq t < t_{(k+1)} \text{ , } k = 1, 2, ..., r-1 \text{ .}$$

It is worth noting that the baseline survival function above coincides with the Nelson-Aalen estimate of the survival function when there are no covariates.
Then, the baseline cumulative hazard function, known as the *Nelson-Aalen estimate* or the *Breslow estimate* can be obtained as:

$$\hat{H}_0(t) = -\ln \hat{S}_0(t) = -\sum_{j=1}^{k} \ln \hat{\xi}_j \approx \sum_{j=1}^{k} \frac{d_j}{\sum_{l \in R_j} e^{\hat{\boldsymbol{\beta}}'\boldsymbol{x}_l}} \text{ for } t_{(k)} \leq t < t_{(k+1)} \text{ , } k = 1, 2, ..., r-1 \text{ .}$$

Finally, the cumulative hazard and the survival functions for the i^{th} subject can be respectively obtained as

$$\hat{H}_i(t) = e^{\hat{\boldsymbol{\beta}}'\boldsymbol{x}_i} \hat{H}_0(t) \text{ ,}$$

and

$$\hat{S}_i(t) = \left\{ \hat{S}_0(t) \right\}^{e^{\hat{\boldsymbol{\beta}}'\boldsymbol{x}_i}} .$$

A further approximation can be obtained by considering that the exponent of $\tilde{\xi}_j$ will tend to be small, unless there are a large amount of ties at particular timepoints. Then, in its turn, $\tilde{\xi}_j$ can be approximated by $\bar{\xi}_j$ as shown in the following:

$$\tilde{\xi}_j = e^{-\frac{d_j}{\sum_{l \in R_j} e^{\hat{\boldsymbol{\beta}}'\boldsymbol{x}_l}}} \approx \bar{\xi}_j = 1 - \frac{d_j}{\sum_{l \in R_j} e^{\hat{\boldsymbol{\beta}}'\boldsymbol{x}_l}} .$$

Using this approximation

$$\hat{h}_0(t) = \frac{1 - \hat{\xi}_j}{\gamma_j} \approx \frac{1 - \tilde{\xi}_j}{\gamma_j} \approx \frac{1 - \bar{\xi}_j}{\gamma_j} = \frac{d_j}{\gamma_j \sum_{l \in R_j} e^{\hat{\boldsymbol{\beta}}'\boldsymbol{x}_l}} \text{ ;}$$

$$\hat{S}_0(t) \approx \prod_{j=1}^{k} \left(1 - \frac{d_j}{\sum_{l \in R_j} e^{\hat{\boldsymbol{\beta}}' \boldsymbol{x}_l}}\right) ;$$

$$\hat{H}_0(t) = -\ln \hat{S}_0(t) .$$

Concluding, it will be computationally advantageous to use one of the two proposed approximations; when the number of tied survival times is small these estimates will tend to be very similar [Collett, 2015].

2.2.2 Extensions of Cox regression model

Different extensions to classical Cox proportional hazards model have been proposed in order to consider some possible drawbacks. In the following paragraphs two particular extensions have been shown (i) to solve the problem of sparse data and (ii) to consider explanatory variables that change over time.

2.2.2.1 Regularized Cox Models

In the last decades high-dimensional data (both in the number of observations and in the number of variables) are usually collected. In some cases, the number of collected variables is greater than the number of observations, leading to a sparse model. Therefore, it has become challenging to build a good model that takes into account all the available features and that simultaneously avoids overfitting. For this reason, research on the regularization of the parameters has been developed in order to select the main explanatory variables in high-dimensional contexts under the assumption that most of them are not significant [Wang et al., 2019]. The use of sparsity norms has been therefore encouraged mainly to perform variable selection when the number of explanatory variables is much bigger than the number of observations ($p > N$) and to avoid overfitting.

As for classical regression models, also for Cox regression model different regularizations have been proposed. In general, regularized parameters can be obtained thanks to the introduction of a penalty in the minimization of a loss function:

$$\beta = argmin_{\beta} l(\beta) + \lambda P(\beta)$$

where $P(\beta)$ is a sparsity norm and λ is the regularization/penalty parameter used to avoid overfitting. The most used penalty terms are the following:

- Lasso;
- Ridge;
- Elastic Net;

- Adaptive Elastic Net;
- OSCAR.

Lasso regularization for Cox proportional hazards model was firstly proposed by Tibshirani [Tibshirani, 1997] and it is useful when the model is too sparse. More in detail, Lasso regularization is based on the use of a ℓ_1-norm penalty, that allows to obtain a well-defined solution with few nonzero coefficients β_k [Simon et al., 2011]. Thus, it is useful because simultaneously it performs feature selection among all the covariates and estimates the regression coefficients. The Lasso penalty term is:

$$P(\beta) = \sum_{k=1}^{p} |\beta_k| \ .$$

Ridge regression was firstly used in the context of Cox regression in 1994 [Verweij and Van Houwelingen, 1994]. This method uses a ℓ_2-norm regularizer to select correlated variables and to shrink their values toward each other [Wang et al., 2019]. More in detail, the penalty term used is:

$$P(\beta) = \sum_{k=1}^{p} \beta_k^2 \ .$$

More recently Zou and Hastie [Zou and Hastie, 2005] proposed to use the *Elastic Net* for linear regression. Then, in 2011 Simon and co-authors [Simon et al., 2011] proposed to use the Elastic Net for Cox regression model. This method is useful to both perform automatic variable selection and continuous shrinkage and to select groups of correlated variables. The elastic net penalty term, that combines linearly the ℓ_1 and squared ℓ_2-penalties, is:

$$P(\beta) = \alpha \sum_{k=1}^{p} |\beta_k| + (1-\alpha) \sum_{k=1}^{p} \beta_k^2 \ .$$

An alternative approach is based on the use of the *Adaptive-Elastic Net*, known also as *Kernel Elastic Net* algorithm [Vinzamuri and Reddy, 2013]. This method builds a kernel similarity matrix for the features space, that is a sort of correlation matrix for the covariates. The used regularizer term is:

$$P(\beta) = \lambda\alpha \sum_{k=1}^{p} |\beta_k| + \lambda(1-\alpha)\boldsymbol{\beta}'\boldsymbol{K}\boldsymbol{\beta} \ ,$$

where $\boldsymbol{K}$ is the symmetric kernel matrix whose generic element K_{ij} is

$$K_{ij} = e^{-\frac{\sum_{i,j=1}^{p}(x_i-x_j)^2}{2\sigma^2}} \quad i,j = 1,2,...,p \ .$$

Finally, another alternative to perform variable selection for highly correlated covariates is the *modified graph Octagonal Shrinkage and Clustering Algorithm for*

Regression (OSCAR) regularizer [Vinzamuri and Reddy, 2013]. With this method equal coefficients are obtained for the explanatory variables which are associated to the outcome is a similar way. More in detail, the OSCAR COX regression model uses as regularization term:

$$P(\beta) = \lambda_1||\beta||_1 + \lambda_2||\boldsymbol{E}\beta||_1 ,$$

where $\boldsymbol{E}$ is the sparse symmetric edge set matrix obtained from a graph that considers each feature as an individual node and the feature similarity as the edge between the two nodes [Wang et al., 2019].

2.2.2.2 Time Dependent Cox model

Although in survival studies individuals are monitored through time, the classical Cox regression model takes into account only explanatory variables collected at the beginning of the study. However, given the longitudinal nature of these studies, it can also be interesting to consider explanatory variables recorded more than once; taking into account the time evolution of the covariates can be useful to obtain a more satisfactory model for the hazard of the event of interest. Therefore, an extension to Cox regression model has been introduced to handle time-dependent covariates, i.e. variables whose values change over time. Two kinds of time-dependent variables may be distinguished: internal and external variables. The first ones are related to a particular individual in the study and come from repeated measurements made on a subject over time. The second ones can also be related to the individual but their change is known in advance at any future time (e.g. the age of a subject) or exist independently of any particular individual, e.g. the air temperature.

Assuming to have a set of covariates that includes both time-dependent (p_1) and time-independent features (p_2), i.e. $\boldsymbol{X(t)} = (X_{1.}(t), X_{2.}(t), ..., X_{p_1.}(t), X_{1.}, X_{2.}, ..., X_{p_2.})$, the time-dependent Cox model can be written as

$$h_i(t) = h_0(t)e^{\sum_{k=1}^{p1} \alpha_k x_{ki}(t) + \sum_{k=1}^{p2} \beta_k x_{ki}} .$$

Hereafter, for simplicity, the following formulation [Collett, 2015] will be used

$$h_i(t) = h_0(t)e^{\sum_{k=1}^{p} \beta_k x_{ki}(t)} ,$$

where $x_{ki}(t)$ is the value of the k^{th} explanatory variable at time t in the i^{th} individual. Alike to the Cox proportional hazards model, the baseline hazard function represents the hazard function for an individual whose values of all the variables are equal to zero at the time origin and remain at this same value through time. Moreover, each coefficient β_k can be interpreted as the log-hazard ratio for two subjects for whom the value of the k^{th} explanatory variable at any time t differs by one unit, while keeping constant all the other $p-1$ variables. Indeed, the ratio of the hazard functions at time

t for two subjects q and s is:

$$\frac{h_q(t)}{h_s(t)} = e^{\beta_1[x_{1q}(t)-x_{1s}(t)]+\ldots+\beta_p[x_{pq}(t)-x_{ps}(t)]} \ .$$

Finally, it is important to notice that this model does not satisfy the proportional hazards assumption because the hazard ratio is time-dependent.
To obtain the estimates of the β parameters the partial log-likelihood to maximize is:

$$\ln L(\beta) = l(\beta) = \sum_{i=1}^{N} \delta_i \left\{ \sum_{k=1}^{p} \beta_k x_{ki}(t_i) - \ln \sum_{l \in R_{t_i}} e^{\sum_{k=1}^{p} \beta_k x_{kl}(t_i)} \right\}$$

with R_{t_i} the risk set at t_i, that is the event time of the i^{th} individual in the study.
The maximization process works if the values of the variables in the model are known at each event time for all individuals in the risk set. This may be a problem mainly for internal variables and for the external variables that cannot be known in advance, i.e. those that exist independently of the individuals. A possible solution is to use the last recorded value before the specific event time at which the variable is needed. Another alternative for variables that have been collected before and after the time of interest is to use the value closest to that time. Finally, if the variable is continuous, linear interpolation between consecutive values could also be used. In these cases, it could be interesting to perform a sensitivity analysis; indeed, if inference is sensitive to the method used, it is important to be careful when interpreting the results because the value of the time-dependent variable could be subject to measurement error, substantial inherent variation or the values have not been recorded sufficiently regularly [Collett, 2015].
After the model has been fitted, the baseline hazard and survival functions can be estimated. When dealing with time-dependent covariates, the Nelson-Aalen estimate of the baseline cumulative hazard function becomes

$$H_0(t) = -\ln S_0(t) = \sum_{j=1}^{k} \frac{d_j}{\sum_{l \in R_j} e^{\hat{\boldsymbol{\beta}}' \boldsymbol{x}_l(t)}} \text{ for } t_{(k)} \leq t < t_{(k+1)} \ , k = 1, 2, \ldots, r-1 \ ,$$

where $\boldsymbol{x}_l(t)$ is the vector of values of the explanatory variables for the l^{th} individual at time t and, coherently with the notation introduced in the previous sections, d_j is the number of events at the j^{th} ordered event time $t_{(j)}$, $j = 1, 2, \ldots, r$.
Having obtained an estimate of the cumulative hazard function, the corresponding baseline hazard and survival functions can be estimated using the following equations

$$h_0(t) = \frac{d_j}{\gamma_j \sum_{l \in R_j} e^{\hat{\boldsymbol{\beta}}' \boldsymbol{x}_l(t)}} \ ,$$

$$S_0(t) = e^{-H_0(t)} \ .$$

The survival function for a particular subject is, instead, much more difficult to estimate because the result obtained for Cox proportional model, i.e.

$$S_i(t) = \{S_0(t)\}^{e^{\hat{\boldsymbol{\beta}}'\boldsymbol{x}_i}}$$

no longer holds. More in detail, for the time-dependent Cox model the survival function for the i^{th} individual can be obtained as

$$S_i(t) = e^{-\int_0^t e^{\sum_{k=1}^p \beta_k x_{ki}(u)} h_0(u)du} .$$

This survival function therefore depends on the baseline hazard function and on the values of the variables over the time interval $[0, t]$. Approximate conditional probabilities of surviving a certain time interval can be found by considering the conditional probability that an individual survives over an interval $P(T_i \geq t+h|T_i \geq t)$. As usual, T_i is the random variable associated to the survival time of the i^{th} individual. This conditional probability can be evaluated as

$$P_i(t, t+h) = P(T_i \geq t+h|T_i \geq t) = \frac{P(T_i \geq t+h)}{P(T_i \geq t)} = \frac{S_i(t+h)}{S_i(t)} .$$

With the additional assumption that any time-dependent variable remains constant through the interval of interest, the approximate conditional probability can be written as

$$P_i(t, t+h) = \frac{\exp\{-e^{\sum_{k=1}^p \beta_k x_{ki}(t)} \int_0^{t+h} h_0(u)du\}}{\exp\{-e^{\sum_{k=1}^p \beta_k x_{ki}(t)} \int_0^{t} h_0(u)du\}} .$$

An estimate of this approximate conditional probability $\hat{P}_i(t, t+h)$ can be obtained by replacing the estimated baseline cumulative hazard function and the estimated β coefficients. Consequently, an estimate of the expected number of events in the interval is $1 - \hat{P}_i(t, t+h)$. Comparing these values with the observed number of events in each interval provides an informal assessment of model adequacy [Collett, 2015].
Finally, model selection for survival data that include time-dependent variables can be performed in the same way as Cox PH model. Therefore, the log-likelihood ratio test statistic for nested models or comparisons between AIC of different models can be performed.

2.3 Parametric models

Nonparametric methods and Cox regression are useful when there is no need to assume a particular form of probability distribution for survival times. As a result, these models have flexibility and widespread applicability. On the other hand, if a particular probability distribution for survival times can be assumed, parametric models can be a better choice. The main advantage of these models is that, if the probability assumption is valid, inference will be more precise [Collett, 2015]. However, as a direct consequence, their performance strongly relies on the choice of the distribution, that is usually a difficult process in many situations. There are different types of parametric models. The simplest ones do not take into account covariates and are simply based on a distributional assumption on survival times (usually the assumption is made on the hazard function). On the other hand, there exist other parametric models that allow to study the effect of a set of explanatory variables on survival times. Among these, there are (i) the parametric proportional hazards models, that extend the Cox regression model when a particular distribution can be assumed for the baseline hazard function (e.g. the Weibull distribution), (ii) linear censored regression models that are based on assumptions on the survival times and (iii) accelerated failure time models, based on distributional assumptions on the logarithm of the survival times.

2.3.1 Models for the hazard function

Alike to nonparametric methods, parametric models allow to directly predict survival times. More in detail, once a distributional assumption has been specified in terms of a probability density function, the corresponding survival and hazard functions can be obtained from the relations already seen in Section 1.1 and recalled below:

$$S(t) = P(T \geq t) = \int_t^{\infty} f(u)du$$

and

$$h(t) = \frac{f(t)}{S(t)} = -\frac{d}{dt} \ln S(t) \ .$$

An alternative to assume a probability density function for survival times is to specify a functional form for the hazard function and then to obtain the cumulative hazard function, the survival function and the probability density function through the following relationships:

$$H(t) = \int_0^t h(u)d(u) \ ,$$

$$S(t) = e^{-H(t)} \ ,$$

and

$$f(t) = h(t)S(t) = -\frac{d}{dt}S(t) \ .$$

2.3.1.1 Exponential model

The simplest parametric model is based on the assumption of constant hazard rate over time, that is

$$h(t) = \lambda \text{ for } t \geq 0$$

with λ a positive constant that is estimated by fitting the model to observed data. Thus, the cumulative hazard, survival and density functions are:

$$H(t) = \int_0^t \lambda du = \lambda t \ ,$$

$$S(t) = e^{-\lambda t} \, ,$$

and

$$f(t) = \lambda e^{-\lambda t} \, .$$

This probability density function corresponds to that of a random variable T with exponential distribution $T \sim Exp(\lambda)$.

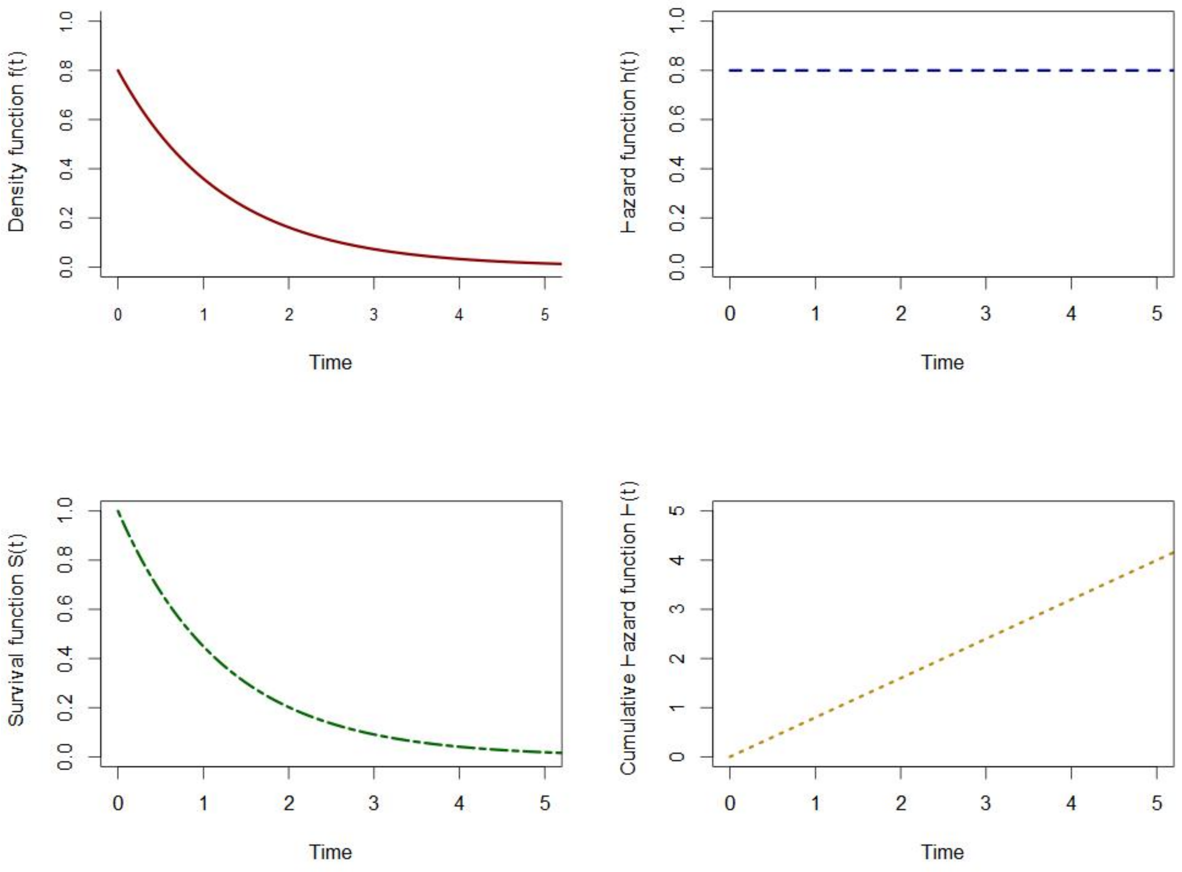


Figure 6: Graphical representation of the probability density function (top-left), hazard function (top-right), survival function (bottom-left) and cumulative hazard function (bottom-right) under the assumption that survival times follow an Exponential distribution $T \sim Exp(0.8)$

Two useful summaries are the mean and the median survival time, that are respectively:

$$\mathbb{E} = \frac{1}{\lambda}$$

and

$$Me[T] = \frac{1}{\lambda} \ln 2$$

In practice, however, the assumption of a constant hazard function is rarely plausible. Therefore, other distributions should be considered.

2.3.1.2 Weibull distribution

An alternative to the Exponential distribution is to assume that survival times follow a Weibull distribution, $T \sim W(\lambda, \nu)$ with λ, $\nu > 0$ the *scale* and the *shape* parameter respectively. The Weibull probability density function is:

$$f(t) = \lambda \nu t^{\nu-1} e^{-\lambda t \nu} \text{ for } t \geq 0 \ .$$

The corresponding survival, hazard and cumulative hazard functions are

$$S(t) = e^{-\lambda t^{\nu}} \ ,$$

$$h(t) = \lambda \nu t^{\nu-1} \ ,$$

and

$$H(t) = \lambda t^{\nu} \ .$$

When survival times follow a Weibull distribution, the hazard function is monotonic and its shape depends strongly on the value of the shape parameter ν.

The mean survival time is

$$\mathbb{E}[T] = \lambda^{-\frac{1}{\nu}} \Gamma(\nu^{-1} + 1) \ ,$$

where $\Gamma(\cdot)$ is the Gamma function, that is

$$\Gamma(s) = \int_0^{\infty} u^{s-1} e^{-u} du = (s-1)! \ .$$

However, since the Weibull distribution is a positively skewed distribution, a more appropriate summary of the location of the distribution is the median survival time:

$$Me[T] = \left[\frac{1}{\lambda} \ln 2 \right]^{\frac{1}{\nu}} \ .$$

Since the Weibull hazard function can take different forms depending on the values of the shape and scale parameters, this distribution is widely used in the parametric analysis of survival data.

2.3.1.3 Suitability of a parametric model

Before fitting a model based on an assumed probability distribution, a preliminary study of the validity of this assumption should be carried out. A first simple approach is to graphically represent the hazard function obtained using nonparametric methods: if it is constant, the exponential distribution would be an appropriate model for the data; on the other hand, if the hazard function increases or decreases monotonically with time, a Weibull distribution would be more suitable. Another way of checking the validity of a particular distributional assumption for survival times is to compare

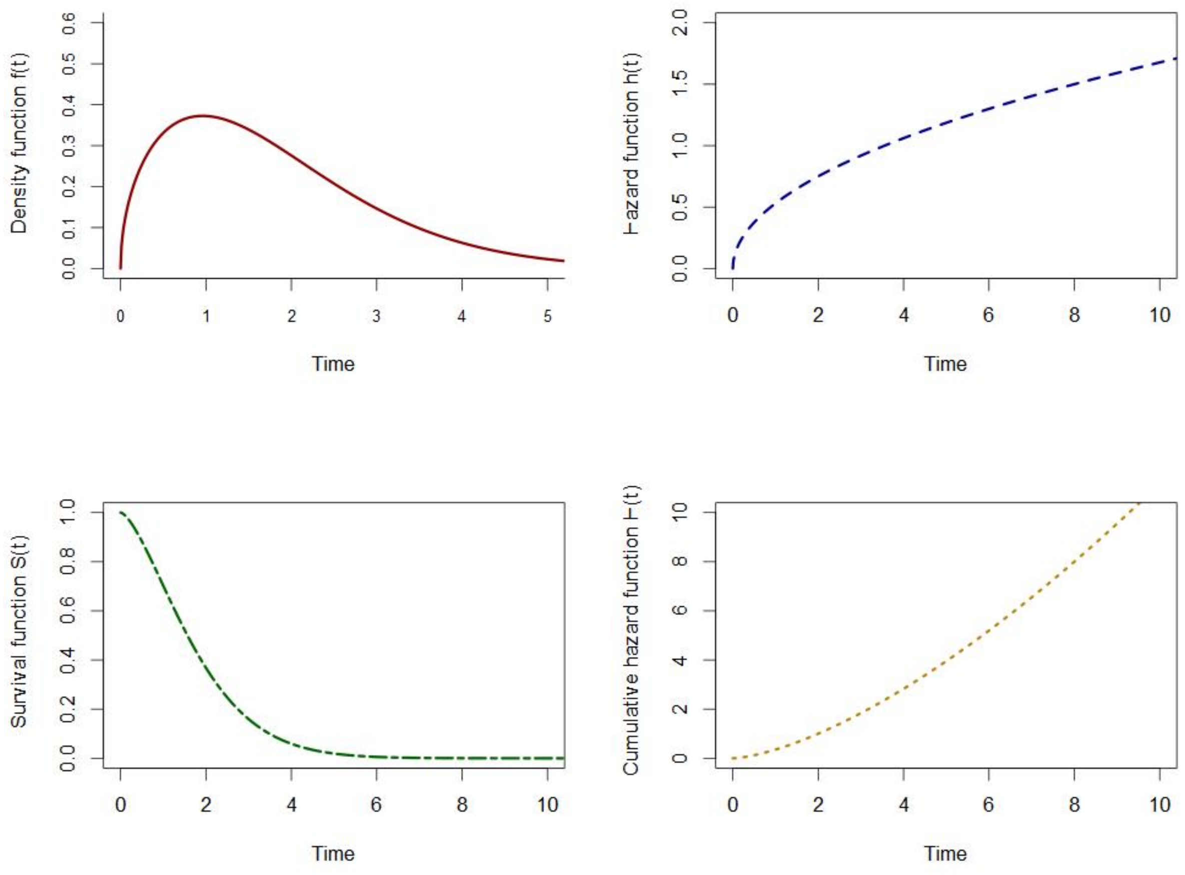


Figure 7: Graphical representation of the probability density function (top-left), hazard function (top-right), survival function (bottom-left) and cumulative hazard function (bottom-right) under the assumption that survival times follow a Weibull distribution $T \sim W(1, 1.5)$

the survival function with that of the theoretical model that has been chosen. This procedure can be carried out by transforming the survival function in such a way that its graphical representation is a straight line if the assumed model is appropriate [Collett, 2015]. For example, supposing to have a sample of survival data and assuming a Weibull distribution for survival times, i.e.

$$S(t) = e^{-\lambda t^{\nu}} ,$$

the following transformation, that corresponds to the log-cumulative hazard, can be used:

$$\ln[-\ln S(t)] = \ln \lambda + \nu \ln t .$$

Then, the Kaplan-Meier estimate of the survival function $\hat{S}(t)$ can be considered and the estimated log-cumulative hazard can be plotted against $\ln(t)$. If the Weibull assumption is plausible, $\hat{S}(t)$ will be close to $S(t)$ and the plot will be approximately a straight line. If the assumption is valid, the log-cumulative hazard plot can also be used to provide a rough estimate (not too accurate) of the two parameters. Indeed, the slope and the exponential of the intercept provides respectively an estimate of the shape parameter ν and of the scale parameter λ. Furthermore, if the slope of the straight line is close to unity, the survival times could have an exponential distribution.

2.3.1.4 Fitting of a parametric model

Parametric models can be fitted to an observed set of survival data using maximum likelihood. Supposing that there are r event times $t_1, t_2, \dots, t_r$ and $N-r$ right censored observations, the total likelihood function can be evaluated as

$$L(\boldsymbol{\theta}) = \prod_{i=1}^{N} [f(t_i)]^{\delta_i} [S(t_i)]^{1-\delta_i} ,$$

where $\boldsymbol{\theta}$ is the vector of all the parameters to estimate.
The first term corresponds to the joint probability for uncensored observations, that is the product of the probability of occurrence of the event. The second term represents the joint probability for censored observations; in this case, the survival probability is considered, because the only known information is that the event did not occur until the end of the observation period.
Alternatively, the likelihood function can be expressed as follows:

$$L(\boldsymbol{\theta}) = \prod_{i=1}^{N} \left[\frac{f(t_i)}{S(t_i)} \right]^{\delta_i} [S(t_i)] = \prod_{i=1}^{N} [h(t_i)]^{\delta_i} [S(t_i)] ,$$

Then, the likelihood (or the log-likelihood) function can be maximized with respect to the unknown parameters in both the density and survival functions.
Once the parameter estimates have been obtained, they can be replaced in the expressions of the hazard, survival and density functions to obtain the respective estimates.

2.3.2 Parametric proportional hazards model

When the interest is on evaluating the effect of a set of covariates on the hazard of event, a parametric version of the Cox regression model can be obtained imposing a Weibull distribution to the baseline hazard function. Thus, the main difference with Cox regression model is that in Cox regression the shape of $h_0(t)$ is unspecified and only determined by data. Differently, in the parametric version a specific parametric form on $h_0(t)$ is imposed. This model provides a more suitable basis for modeling the data when the assumption of a specific distribution is appropriate for the observed data.

2.3.2.1 Weibull proportional hazards model

The Weibull distribution is usually chosen and preferred to other ones because it has the *proportional hazards property.* More in details, the Weibull PH model is obtained imposing a Weibull distribution to the baseline hazard function (see Subsection 2.3.1.2) in the Cox regression model. The resulting model is then:

$$h_i(t) = e^{\boldsymbol{\beta}'\boldsymbol{x}_i}\lambda\nu t^{\nu-1} \ .$$

The corresponding survival function is

$$S_i(t) = e^{-e^{\boldsymbol{\beta}'\boldsymbol{x}_i}\lambda t^{\nu}} \ .$$

The Exponential PH model can be easily obtained exploiting the relationship between the Exponential and Weibull distributions, assuming the shape parameter ν equal to 1.

A simple example to show the validity of the PH property for the Weibull distribution is provided below. Considering the simpler case of a Cox PH model with only a dichotomous variable x and assuming that survival times have a Weibull distribution, the corresponding Cox regression model is

$$h_i(t) = e^{\beta x_i}\lambda\nu t^{\nu-1} \ .$$

Thus, if $x_i = 1$

$$h_i(t) = e^{\beta}\lambda\nu t^{\nu-1} \ ,$$

and if $x_i = 0$

$$h_0(t) = \lambda\nu t^{\nu-1} \ .$$

Then, it can be easily noticed that the hazard function for a subject with $x = 1$ corresponds to the hazard of a Weibull distribution with scale parameter $e^{\beta}\lambda$ and shape parameter ν; while for a subject with $x = 0$ it corresponds to the hazard of a Weibull distribution with scale parameter λ and shape parameter ν. Thus, the hazard functions of the two groups are proportional and the proportional hazards property is satisfied.

The Weibull PH model can then be fitted by maximizing the likelihood function of the N observations with respect to the unknown parameters $\boldsymbol{\beta} = (\beta_1, \beta_2, ..., \beta_p)$, λ and ν. The likelihood function and the corresponding log-likelihood function are:

$$L(\boldsymbol{\theta}) = L(\boldsymbol{\beta}, \lambda, \nu) = \prod_{i=1}^{N}[h_i(t_i)]^{\delta_i} S_i(t_i) = \prod_{i=1}^{N}[e^{\boldsymbol{\beta}'\boldsymbol{x}_i}\lambda\nu t^{\nu-1}]^{\delta_i} e^{-e^{\boldsymbol{\beta}'\boldsymbol{x}_i}\lambda t^{\nu}} \ ;$$

$$
\begin{aligned}
\ln L(\boldsymbol{\theta}) = l(\boldsymbol{\theta}) &= \sum_{i=1}^{N} \{\delta_i \ln h_i(t_i) + \ln S_i(t_i)\} \\
&= \sum_{i=1}^{N} \{\delta_i [\boldsymbol{\beta}' \boldsymbol{x}_i + \ln \lambda\nu + (k-1)\ln t_i] - e^{\boldsymbol{\beta}'\boldsymbol{x}_i} \lambda t^{\nu}\} \\
&= \sum_{i=1}^{N} \{\delta_i [\boldsymbol{\beta}' \boldsymbol{x}_i + \ln \lambda\nu + \nu \ln t_i] - e^{\boldsymbol{\beta}'\boldsymbol{x}_i} \lambda t^{\nu}\} - \sum_{i=1}^{N} \delta_i \ln t_i \ .
\end{aligned}
$$

The last term can be omitted from the log-likelihood function because it does not contain any of the unknown parameters of interest.

2.3.3 Accelerated failure time models

The PH models are widely used in survival analysis; however, there are relatively few distributions that can be used with this family of models [Collett, 2015]. Indeed, beyond Exponential and Weibull distributions, many other distributions can be assumed for survival times; examples are the log-logistic, the log-normal and the Gamma distributions. To this extent *accelerated failure time models* are noteworthy. The main difference between the proportional hazard model and the accelerated failure time (AFT) model is that in the first one the covariates act multiplicatively on the hazard, while in the second one the explanatory variables are assumed to act multiplicatively on the time-scale, affecting the rate at which an individual proceeds along the time axis [Collett, 2015].
According to the general accelerated failure time model, the hazard distribution can also be written as

$$
h_i(t) = e^{-\eta_i} h_0\left(\frac{t}{e^{\eta_i}}\right) ,
$$

where $\eta_i = \boldsymbol{\alpha}'\boldsymbol{x}_i = \alpha_1 x_{1i} + \alpha_2 x_{2i} + \cdots + \alpha_p x_{pi}$ is the linear component of the model and $h_0(\cdot)$ is the baseline hazard function. The term $e^{-\eta_i}$ is known as *acceleration factor*.
The corresponding survival function is then

$$
S_i(t) = S_0\left(\frac{t}{e^{\eta_i}}\right) ,
$$

where $S_0(\cdot)$ is the baseline survival function, i.e. the survival function of an individual for whom $\boldsymbol{x} = \mathbf{0}$.

AFT models can also be represented as log-linear models for the survival time T as follows:

$$
\ln T_i = \mu + \alpha_1 x_{1i} + \alpha_2 x_{2i} + ... + \alpha_p x_{pi} + \sigma\epsilon_i \ .
$$

In this model α_1, ..., α_p are the unknown coefficients related to the p explanatory variables and μ and σ are two further parameters, known as the intercept and scale parameter respectively. Finally, ϵ_i is a random variable assumed to have a specific probability distribution. In this formulation, the α parameters represent the effect that each explanatory variable has on the survival times: positive values suggest that survival time increases with increasing values of the explanatory variable, and vice versa [Collett, 2015].

The relationship between the two formulations of the AFT model can be proved as follows. Starting from the log-linear representation of the AFT, the survival function can be expressed as

$$S_i(t) = P(T_i \geq t) = P\Big(e^{\mu+\alpha_1 x_{1i}+\ldots+\alpha_p x_{pi}+\sigma\epsilon_i} \geq t\Big) = P\Big(e^{\mu+\sigma\epsilon_i} \geq te^{-\boldsymbol{\alpha}'\boldsymbol{x}_i}\Big) \ ,$$

and consequently, the baseline survival function can be written as

$$S_0(t) = P\Big(e^{\mu+\sigma\epsilon_i} \geq t\Big) \ .$$

Then, the general form of the survival function for the i^{th} individual in an accelerated failure time model can be expressed in terms of the baseline survival function:

$$S_i(t) = S_0\left(\frac{t}{e^{\boldsymbol{\alpha}'\boldsymbol{x}_i}}\right) \ .$$

Finally, recalling that $h_i(t) = -\frac{d}{dt}\ln S_i(t)$, the resulting hazard function is

$$h_i(t) = e^{-\boldsymbol{\alpha}'\boldsymbol{x}_i} h_0\left(\frac{t}{e^{\boldsymbol{\alpha}'\boldsymbol{x}_i}}\right) \ ,$$

which is exactly the formulation of the AFT model already seen above.

The log-linear expression of the model can also be used to express the survival function for the i^{th} individual in a general form:

$$\begin{aligned} S_i(t) &= P(T_i \geq t) = P(\ln T_i \geq \ln t) \\ &= P(\mu + \alpha_1 x_{1i} + \ldots + \alpha_p x_{pi} + \sigma\epsilon_i \geq \ln t) \\ &= P\left(\epsilon_i \geq \frac{\ln t - \mu - \alpha_1 x_{1i} - \ldots - \alpha_p x_{pi}}{\sigma}\right) \ . \end{aligned}$$

Defining with $S_{\epsilon_i}(\epsilon)$ the survival function of the random variable ϵ_i in the log-linear model, the survival function of the i^{th} individual can then be expressed as

$$S_i(t) = S_{\epsilon_i}\left(\frac{\ln t - \mu - \alpha_1 x_{1i} - \ldots - \alpha_p x_{pi}}{\sigma}\right) \ . \tag{1}$$

This shows how the survival function for T_i can be obtained from the survival function of the distribution of ϵ_i. Moreover, this result also shows that an AFT model can be derived under different assumptions on the probability distributions of ϵ_i.

Similarly, the cumulative hazard function of the distribution of T_i can be obtained as follows:

$$\begin{aligned} H_i(t) &= -\ln S_i(t) \\ &= -\ln S_{\epsilon_i}\left(\frac{\ln t - \mu - \alpha_1 x_{1i} - \ldots - \alpha_p x_{pi}}{\sigma}\right) \\ &= H_{\epsilon_i}\left(\frac{\ln t - \mu - \alpha_1 x_{1i} - \ldots - \alpha_p x_{pi}}{\sigma}\right) \ , \end{aligned} \tag{2}$$

where $H_{\epsilon_i}(\epsilon) = -\ln S_{\epsilon_i}(\epsilon)$ is the cumulative hazard function of ϵ_i.
Finally, the corresponding hazard function can be evaluated differentiating $H_i(t)$ with respect to t:

$$h_i(t) = \frac{1}{\sigma t} h_{\epsilon_i}\left(\frac{\ln t - \mu - \alpha_1 x_{1i} - ... - \alpha_p x_{pi}}{\sigma}\right) . \tag{3}$$

Different choices for the distribution of ϵ_i in the log-linear formulation lead to different distributions for T_i, the random variable associated with the survival time of the i^{th} individual. In particular, the survival and hazard function of ϵ_i lead to the AFT models for survival times. Among the most used distributions there are the Weibull, the log-logistic and the log-normal distributions.

2.3.3.1 Weibull accelerated failure time model

Supposing that survival times are assumed to follow a Weibull distribution $W(\lambda, \nu)$ with scale and shape parameters λ and ν respectively, the baseline hazard function is $h_0(t) = \lambda\nu t^{\nu-1}$ (see Subsections 2.3.1.2 and 2.3.2.1).
Under an AFT model, then, the hazard function for the i^{th} observation is given by

$$h_i(t) = e^{-\eta_i} h_0\left(\frac{t}{e^{\eta_i}}\right) = e^{-\eta_i}\lambda\nu(te^{-\eta_i})^{\nu-1} = e^{-\eta_i\nu}\lambda\nu t^{\nu-1} .$$

So, it follows that the survival times of this observation have a $W(\lambda e^{-\eta_i\nu}, \nu)$ distribution. For this reason the Weibull distribution is said to have the *accelerated failure time property.* It is the only distribution that satisfies both the proportional hazards (see Subsection 2.3.2.1) and the accelerated failure time properties. Summarizing, if the baseline hazard function is the hazard function of a $W(\lambda, \nu)$, survival times have a $W(\lambda e^{\boldsymbol{\beta}'\boldsymbol{x}_i}, \nu)$ distribution under a proportional hazards model and a $W(\lambda e^{-\nu\boldsymbol{\alpha}'\boldsymbol{x}_i}, \nu)$ distribution under an accelerated failure time model. Then, it can be noticed that multiplying the α coefficients in the linear component of the AFT model by $-\nu$, the corresponding β coefficients in the proportional hazards model are obtained [Collett, 2015].

In terms of log-linear representation of the model, if T_i has a Weibull distribution then ϵ_i follows the Gumbel distribution, an asymmetric distribution whose survival function is

$$S_{\epsilon_i}(\epsilon) = e^{-e^{\epsilon}} \text{ for } -\infty < \epsilon < \infty .$$

The corresponding cumulative hazard and hazard functions of this distribution are

$$H_{\epsilon_i}(\epsilon) = e^{\epsilon}$$

and

$$h_{\epsilon_i} = e^{\epsilon} .$$

It can be proved that the corresponding random variable $T_i = e^{\mu+\boldsymbol{\alpha}'\boldsymbol{x}_i+\sigma\epsilon_i}$ has a Weibull distribution by using the relationship in (1):

$$S_i(t) = S_{\epsilon_i}\left(\frac{\ln t - \mu - \alpha_1 x_{1i} - ... - \alpha_p x_{pi}}{\sigma}\right) = e^{-e^{\frac{\ln t - \mu - \alpha_1 x_{1i} - ... - \alpha_p x_{pi}}{\sigma}}} ,$$

that can be expressed in the reduced form

$$S_i(t) = e^{-\lambda_i t^{1/\sigma}} ,$$

with $\lambda_i = e^{-(\mu+\alpha_1 x_{1i}+...+\alpha_p x_{pi})/\sigma}$.
It can be seen that the above survival function is that of a Weibull distribution with scale parameter λ_i and shape parameter σ^{-1}. Therefore, it corresponds to the accelerated failure time representation of the survival function of the Weibull model. It can also be proved that a direct correspondence exists between the survival functions of the Weibull PH model and of the Weibull AFT model. Recalling that under the Weibull proportional hazards model the survival function for the i^{th} individual is

$$S_i(t) = e^{-e^{\beta_1 x_{1i}+\beta_2 x_{2i}+...+\beta_p x_{pi}} \lambda t^{\nu}}$$

where λ and ν are the parameters of the Weibull baseline hazard function (see Subsection 2.3.2.1), it can be seen that $\lambda = e^{-\frac{\mu}{\sigma}}$, $\nu = \sigma^{-1}$ and $\beta_k = -\frac{\alpha_k}{\sigma}$ for $k = 1, 2, ..., p$. Thus, the following log-linear model in which ϵ_i has a Gumbel distribution:

$$\ln T_i = \frac{1}{\nu}[-\ln \lambda - \beta_1 x_{1i} - \beta_2 x_{2i} - ... - \beta_p x_{pi} + \epsilon_i] .$$

provides an alternative representation of the Weibull proportional hazards model.

2.3.3.2 Log-logistic accelerated failure time model

Another example of AFT can be provided assuming that survival times have a log-logistic distribution with parameters θ and κ. The corresponding baseline hazard function is

$$h_0(t) = \frac{e^{\theta}\kappa t^{\kappa-1}}{1+e^{\theta}t^{\kappa}} .$$

Under the accelerated failure time model, the hazard at time t for the i^{th} individual is

$$h_i(t) = e^{-\eta_i} h_0(e^{-\eta_i}t) = \frac{e^{-\eta_i}e^{\theta}\kappa(e^{-\eta_i}t)^{\kappa-1}}{1+e^{\theta}(e^{-\eta_i}t)^{\kappa}} = \frac{e^{\theta-k\eta_i}\kappa t^{\kappa-1}}{1+e^{\theta-\kappa\eta_i}t^{\kappa}} .$$

It then follows that the survival time for the i^{th} individual has also a log-logistic distribution with parameters $\theta - \kappa\eta_i$ and κ. Therefore, as the Weibull distribution, also the log-logistic distribution has the accelerated failure time property. Then, it can be shown that the log-linear form of the AFT also provides a representation of the log-logistic distribution [Collett, 2015]. Supposing that ϵ_i is assumed to have a logistic distribution with zero mean and variance $\pi^2/3$, so that its survival function is

$$S_{\epsilon_i}(\epsilon) = \frac{1}{1+e^{\epsilon}} ,$$

the survival function of T_i can then be expressed as

$$S_i(t) = \frac{1}{1 + e^{\frac{\ln t - \mu - \alpha_1 x_{1i} - \ldots - \alpha_p x_{pi}}{\sigma}}} .$$

Comparing this survival function with that of a random variable with log-logistic distribution with parameters $\theta - \kappa\eta_i$ and κ, i.e.

$$S_i(t) = \frac{1}{1 + e^{\theta - \kappa\eta_i} t^{\kappa}} ,$$

it can be derived that the parameters θ and κ can be expressed in terms of μ and σ as $\theta = -\frac{\mu}{\sigma}$ and $\kappa = \sigma^{-1}$. Therefore, it has been shown that the accelerated failure time model with log-logistic survival times can be formulated in terms of a log-linear model.

Finally, the cumulative hazard and hazard functions of the distribution of ϵ_i are such that

$$H_{\epsilon_i}(\epsilon) = \ln(1 + e^{\epsilon})$$

and

$$h_{\epsilon_i}(\epsilon) = \frac{1}{1 + e^{-\epsilon}} .$$

Using these two expressions, $H_i(t)$ and $h_i(t)$ can be found using the expressions shown in Subsection 2.3.3 (Equations (2),(3)).

2.3.3.3 The log-normal accelerated failure time model

Another possible distribution for the survival times is the log-normal distribution with parameters μ and σ. In this case, the baseline survival function is equal to

$$S_0(t) = 1 - \Phi\left(\frac{\ln t - \mu}{\sigma}\right) .$$

Under the AFT model, the survival function can then be written as

$$S_i(t) = S_0(e^{-\eta_i} t) = 1 - \Phi\left(\frac{\ln t - \eta_i - \mu}{\sigma}\right) ,$$

that corresponds to the survival function for an individual whose survival times follow a log-normal distribution with parameters $\mu + \eta_i$ and σ. Therefore, also the log-normal distribution has the accelerated failure time property. In the log-linear formulation, the random variable associated with the survival time of the i^{th} individual has a log-normal distribution if its logarithm is normally distributed. Therefore ϵ_i is assumed to have a standard normal distribution whose survival, cumulative hazard and hazard functions are respectively

$$S_{\epsilon_i}(\epsilon) = 1 - \Phi(\epsilon) ;$$

$$H_{\epsilon_i}(\epsilon) = -\ln\left(1 - \Phi(\epsilon)\right) ;$$

$$h_{\epsilon_i}(\epsilon) = \frac{f_{\epsilon_i}(\epsilon)}{S_{\epsilon_i}(\epsilon)} = \frac{\frac{1}{\sqrt{2\pi}}e^{-\frac{\epsilon^2}{2}}}{1-\Phi(\epsilon)} \ .$$

These expressions can then be used to evaluate the survival, hazard and cumulative hazard functions of T_i (through the relationships shown in Subsection 2.3.3, Equations (1), (3) and (2)).

2.3.3.4 Fitting the accelerated failure time model

The estimates of the unknown parameters are obtained, as usual, through maximum likelihood estimation. The likelihood function

$$L(\boldsymbol{\alpha}, \mu, \sigma) = \prod_{i=1}^{N} [f_i(t_i)]^{\delta_i} [S_i(t_i)]^{1-\delta_i} \ .$$

can be derived from the log-linear representation of the AFT models using the expression of the survival function in terms of S_{ϵ_i}. Indeed, the survival function $S_i(t_i)$ can be written as $S_{\epsilon_i}(\frac{\ln t_i - \mu - \alpha_1 x_{1i} - \cdots - \alpha_p x_{pi}}{\sigma}) = S_{\epsilon_i}(z_i)$. Then, the density function can be obtained by differentiating the survival function with respect to t:

$$f_i(t_i) = \frac{1}{\sigma t_i} f_{\epsilon_i}(z_i) \ .$$

Therefore, the likelihood function can be written as

$$L(\boldsymbol{\alpha}, \mu, \sigma) = \prod_{i=1}^{N} (\sigma t_i)^{-\delta_i} [f_{\epsilon_i}(z_i)]^{\delta_i} [S_{\epsilon_i}(z_i)]^{1-\delta_i} \ ,$$

and the corresponding log-likelihood function to be maximized is

$$\ln L(\boldsymbol{\alpha}, \mu, \sigma) = l(\boldsymbol{\alpha}, \mu, \sigma) = \sum_{i=1}^{N} -\delta_i \ln \sigma t_i + \delta_i \ln f_{\epsilon_i}(z_i) + (1-\delta_i) \ln S_{\epsilon}(z_i) \ .$$

After having fitted the model and having obtained the maximum likelihood estimates of the unknown parameters, nested models can be compared through the evaluation of the statistic $-2\ln \hat{L}$. Then, once a suitable model has been identified, the estimates of the survival and hazard function can be obtained. The estimated value of the acceleration factor $e^{-\hat{\eta}_i}$ for specific individuals, or the median and other percentiles of the distribution of survival times, can be used to comment the attained results. In particular, the p^{th} percentile can be obtained as

$$\hat{t}_i(p) = e^{\hat{\sigma}\epsilon_i(p) + \hat{\mu} + \hat{\alpha}_1 x_{1i} + \cdots + \hat{\alpha}_p x_{pi}} \ ,$$

where $\hat{\sigma}$, $\hat{\mu}$, $\hat{\alpha}_1, \ldots, \hat{\alpha}_p$ are the maximum likelihood estimates of the unknown parameters, and $\epsilon_i(p)$ is the p^{th} percentile of the distribution of ϵ_i.

Concluding, as already hinted, other parametric censored regression models exist. Among these there are the Tobit model [Tobin, 1958], that was one of the earliest attempts to extend a linear regression model for censored data, the Buckley-James regression model [Buckley and James, 1979] and the proportional odds model [Bennett, 1983a,b].

3 Machine Learning Methods for Survival Analysis

As shown by Breiman [2001b], in many situations machine learning methods can perform better than classical statistical models. It is therefore interesting to study and to consider machine learning methods in the context of survival analysis. Among these (see Figure 3), I will mainly focus on survival trees and random survival forests.

3.1 Survival Trees

3.1.1 Rationale and preliminary concepts

In classical survival analysis the Cox PH regression model and parametric censored regression models are very useful because of the simple interpretation of the covariate effects on the hazard function. However, they also have some limits because a specific link between the covariates and the response variable is assumed, and because they do not take into account interactions between covariates, unless specified. To this extent, survival trees are a popular nonparametric alternative to classical survival models. They offer great flexibility and can automatically detect certain types of interactions without the need to specify them beforehand. For these reasons interest in extending tree-based methods to survival analysis increased, and the development of survival trees grew from the mid-1980s up to the mid-1990s [Bou-Hamad et al., 2011b].

Survival trees were born as an extension of Classification and Regression Trees (CART) [Breiman et al., 1984]. This algorithm can be used to define interpretable classification rules on the basis of data and it is very attractive for many fields, including medicine, economics and finance. Dealing with survival data this algorithm can be useful, for example, in medical research for both understanding the prognostic structure of data and designing future clinical trials [LeBlanc and Crowley, 1993]; in economics, it can be instead used, for example, for studying the exit from unemployment [Bieszk-Stolorz and Dmytrów, 2017]. Finally, in finance this algorithm can be used, as an example, for predicting bankruptcy [Bou-Hamad et al., 2011a].

Therefore, survival trees, as CART, can be a useful tool for defining homogeneous groups according to their survival behavior and, consequently, permit to specify a rule based on the covariates for uniquely assigning a potential new subject to one of the obtained groups. The partition of the covariate space can be obtained using one-sample tools for censored survival data, such as the Kaplan-Meyer estimator or other simple statistics such as the quantiles, to compare the outcome between the nodes in the tree. Usually, for prediction goals, the K-M estimate of the survival function or the median survival time in the terminal nodes are used.
The basic setup that leads to the development of survival trees is the same shown in the previous chapters and, for the sake of clarity, the main notation is briefly recalled hereafter. The true survival time and the true censoring time are denoted respectively

by T and C. The observed data for each subject can be represented by the triplet $(\boldsymbol{x}_i, \tau_i, \delta_i)$, with $i = 1, 2, \ldots, N$. The elements of the triplet are:

- $\tau_i = \min(t_i, c_i)$, the time until either the event occurs or the subject is censored;
- $\delta_i = I_{t_i \leq c_i}$, an indicator variable that takes a value of 1 if the true time-to-event is observed and 0 if the subject is censored;
- $\boldsymbol{x} = (x_1, x_2, \ldots, x_p)$ a vector of p observed covariates.

Finally, an additional assumption is that T and C are independent given $\boldsymbol{x}$.
Before entering into the details of survival trees, a brief introduction to the CART algorithm is reported in the subsection below.

3.1.1.1 Classification and Regression Trees (CART)

CART are statistical tools that can be used to predict a categorical or a continuous outcome (classification and regression trees, respectively), using a set of explanatory variables. Roughly speaking, the basic idea is to divide the population into homogeneous subgroups with respect to the outcome of interest. In order to obtain these groups, the CART algorithm involves the recursive partitioning of the predictor space χ, i.e. the set of all the possible values for the explanatory variables X_1, X_2, $\ldots$, X_p, into distinct and non-overlapping regions (terminal nodes), which induce a corresponding partition of the sample into groups of subjects that are similar according to the outcome of interest. Trees are typically grown upside down, starting with a root node and ending with the leaves at the bottom of the tree. The subjects are repeatedly split into two parts according to the criterion of achieving maximum homogeneity within the new parts or, correspondingly, to impress the maximum heterogeneity reduction of the outcome through the split. This procedure returns a binary tree like that represented in Figure 8. The circles represent *internal nodes*, i.e. nodes that have direct descendants (*children nodes*), or, in other words, subsets that undergo further splitting. Nodes that are no longer partitioned are known as *terminal nodes/leaves* of the tree (represented by rectangles in the figure). These nodes correspond to the final partition defined by the tree; each observation belongs to only one of the terminal nodes, on the basis of its covariates.
Other two definitions useful to understand well the methods shown in this work are the following. A *branch* $\mathcal{T}_j$ of a tree $\mathcal{T}$ consists of the node R_j and all its descendants in $\mathcal{T}$ (highlighted by the green dotted line in Figure 8). A tree $\mathcal{T}_1$ is a *subtree* of $\mathcal{T}$ if $\mathcal{T}_1$ is a tree with the same root node of the original tree $\mathcal{T}$, and if $\forall$ node R_j in $\mathcal{T}_1$, then R_j is also in $\mathcal{T}$ (dashed blue line in Figure 8) [LeBlanc and Crowley, 1993].
A split partitions a node into a left and a right node. The most used splits are the univariate splits, i.e. splits on the single covariate. If the predictor X_k is categorical, the set of candidate splits is obtained from the answer to questions as "Does $x_k \in A$?",

where A is a subset of different categories of the variable. If, instead, the variable is continuous, the splits are made on the basis of questions like "Is $x_k \leq s$?", with s, namely the threshold, any real number that can be observed for the variable [De Rose and Pallara, 1997].

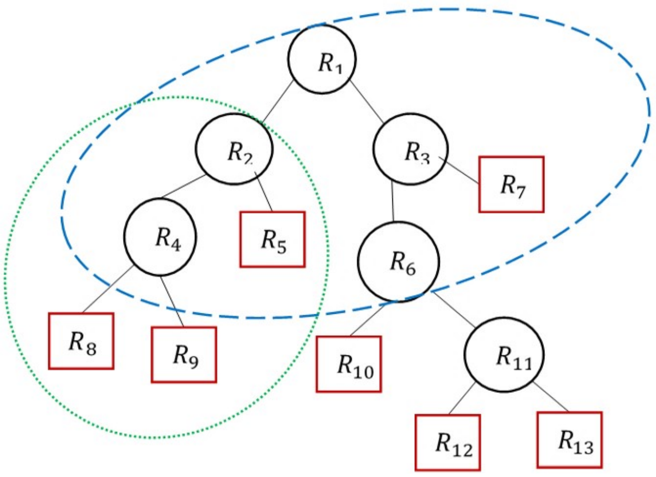


Figure 8: Graphical representation of a decision tree with 13 nodes. R_1 is the root node; the nodes represented by circles are internal nodes (R_2, R_3, R_4, R_6 and R_{11}). The nodes represented by rectangles are the terminal nodes (R_5, R_7, R_8, R_9, R_{10}, R_{12} and R_{13}). The dashed blue line indicates a subtree of the original tree. The dotted green line represents the branch $\mathcal{T}_2$, a tree with root node R_2 and including all its descendants.

Building the tree - Recursive partitioning

The construction of the tree is performed using a top-down greedy approach, known as *recursive binary splitting*, that is usually adopted to find the best partitions of the predictor space. The recursive approach is top-down because, as already said, it starts at the root node of the tree and then successively splits the predictor space. Moreover, it is defined greedy because at each step of the tree-building process the best split is made at that particular step, rather than looking ahead and picking a split that will lead to a better tree in some future step [James et al., 2013].
Formally, the algorithm performs a thorough search through all the potential binary splits with the covariates and selects the best one according to a *splitting criterion*, as an impurity measure. When a final tree has been obtained, node summaries are usually computed at the terminal nodes to obtain predictions for each defined subgroup. If the outcome is continuous (regression tree) the predicted value $\hat{y}_j$ is usually the mean evaluated for the training observations in each terminal node; if the outcome is categorical (classification trees) the most commonly occurring category in the corresponding terminal node is usually used. This means using the category m^* such that the proportion $\hat{p}_{jm^*}$ of training observations belonging to that class in the given node is maximum. Let $J_{\tilde{\mathcal{T}}}$ be the index set of the terminal nodes of the tree

$\mathcal{T}$. For regression trees the aim is to find the regions R_j ($j \in J_{\tilde{\mathcal{T}}}$) that minimize the residual sum of squares (RSS):

$$RSS = \sum_{j \in J_{\tilde{\mathcal{T}}}} \sum_{i \in R_j} (y_i - \hat{y}_j)^2 \ ,$$

where $\hat{y}_j$ is, as already said, the mean response for the training observations within the terminal node R_j ($j \in J_{\tilde{\mathcal{T}}}$).
Hereafter, for the sake of simplicity, I will refer only to quantitative predictors. The extension to categorical explanatory variables is straightforward. In details, when a node R_j is split, for any X_k and any threshold s, the following subsets of the training set are defined

$$R_{jl}(k,s) = i | i \in R_j \wedge x_{ki} < s \text{ and } R_{jr}(k,s) = i | i \in R_j \wedge x_{ki} \geq s \ ,$$

where x_{ki} is the value of X_k for the i^{th} observation. The values of k and s that minimize the following expression are searched:

$$\sum_{i \in R_{jl}(k,s)} (y_i - \hat{y}_{jl})^2 + \sum_{i \in R_{jr}(k,s)} (y_i - \hat{y}_{jr})^2 \ ,$$

where $\hat{y}_{jl}$ and $\hat{y}_{jr}$ are respectively the mean response for the training observations within the nodes $R_{jl}(k,s)$ and $R_{jr}(k,s)$.

The process is repeated, searching for the best predictor and split point in order to further partition the data so as to minimize the RSS within each of the resulting regions. After the first step (that partitions the root node) only one of the two previously identified regions is divided. The process continues until a stopping criterion is reached (usually until a minimum node size is attained, or the variance decrease due to a further split is below a given threshold).
In a classification setting RSS cannot be used as a splitting criterion, and three possible alternatives exist. The first one is the *classification error rate*. Supposing that the outcome is a categorical variable with M categories, the classification error rate can be defined as the portion of the training observations in that region that do not belong to the most occurring class (that corresponds to the predicted class for the observations in that region):

$$E = 1 - \max_m \hat{p}_{jm}$$

where $\hat{p}_{jm}$ is the proportion of training observations belonging to class m in region R_j.
The second possible splitting criterion is the *Gini index*, that is a measure of total heterogeneity across the M classes:

$$G = \sum_{m=1}^{M} \hat{p}_{jm}(1 - \hat{p}_{jm}) \ .$$

This index assumes small values if all the $\hat{p}_{jm}$'s are close to zero or one ($\sum_{m=1}^{M} \hat{p}_{jm} = 1$). For this reason, it is referred to as a measure of node purity: a small value indicates that a node contains predominantly observations from a specific class.
The last alternative measure is the *entropy*, given by:

$$D = -\sum_{m=1}^{M} \hat{p}_{jm} \ln \hat{p}_{jm} \ .$$

As the Gini index, if the node R_j is pure this measure will assume small values, with the $\hat{p}_{jm}$'s all near zero or one.
Usually, either the Gini index or the entropy are used because they are more sensitive to node purity than the classification error rate [James et al., 2013].
After the splitting procedure has been ended and the terminal regions have been created, the prediction for a given test observation can be made, as already described above.

Problems of overfitting - Pruning the tree
This process of recursive binary splitting may produce good predictions on the training set, but is likely to overfit the data, leading to poor test set performances. Overfitting is a typical feature of statistical learning that occurs when the algorithm may be catching some patterns observed in the training set that are just caused by random noise rather than by true properties of the studied phenomenon. In order to avoid this, a bias-variance trade-off should be faced. Rather than using a large tree, a smaller tree with fewer splits might be chosen, leading to lower variance and better interpretation at the cost of a little bias [James et al., 2013]. The most used method to prevent overfitting with trees is *pruning*. The idea is to grow a very large (*full-grown*) tree and then prune it in order to obtain a sequence of nested subtrees. Then, the smallest optimal pruned tree will be chosen as the final one.
For regression trees pruning is made through the minimization of cost-complexity, that is defined as:

$$\sum_{j \in J_{\tilde{\mathcal{T}}}} \sum_{i \in R_j} (y_i - \hat{y}_j)^2 + \alpha |J_{\tilde{\mathcal{T}}}| \ ,$$

where $|J_{\tilde{\mathcal{T}}}|$ is the number of terminal nodes of the tree $\mathcal{T}$. The tuning parameter α (≥ 0) controls a trade-off between the subtree's complexity (given by the number of terminal nodes) and its fit to the training data (given by the impurity measure, in this case the RSS). As α increases, there is a price to pay for having a tree with many terminal nodes, and so the cost-complexity will tend to be minimized for a smaller subtree. Breiman et al. [1984] proved that for any value of α there exists a unique smallest subtree of the full-grown tree that minimizes the cost-complexity. It turns out that as α increases, branches get pruned from the tree in a nested way; therefore, obtaining the whole sequence of optimally pruned subtrees as a function of α is easy. Usually, the value of α is determined using a validation set or cross-validation. Then, returning to the full data set, the subtree corresponding to the resulting α is obtained. An alternative to the minimization of the cross-validation error for choosing the final model is the use of the "*one standard error rule*", that consists in choosing as final tree the smallest tree within one standard error of the minimum cross-validation error.

Similarly to regression trees, the classification error rate, the Gini index or the entropy might be used as impurity measures when pruning a classification tree. Among these, the classification error rate is preferable when the main aim is the prediction accuracy of the final pruned tree [James et al., 2013].

Finally, a possible alternative to pruning is called *early stopping*. It consists in trying to stop the tree-building process early, before it produces leaves with very small samples. At each stage of splitting the tree, the cross-validation error is checked and if it does not decrease significantly enough then the algorithm is stopped. The disadvantage of this approach is that it may underfit data by stopping too early; this because a seemingly worthless split early in the tree might be followed by a split that leads to a large reduction in the impurity measure later on.

Missing data

In CART missing data are usually handled with the use of *surrogate variables*. Given s the best split for x, and s^* the best split for x^*, x^* is the surrogate variable for x if s and s^* are closest to one another in terms of predictive association [Ishwaran et al., 2008]. Therefore, a surrogate split s^* is the one that most accurately predicts the action of s [Breiman et al., 1984]. So, if an observation has a missing value for the variable used to split a node, the CART algorithm uses the best surrogate variable among those that are not missing for the observation. In this way every observation can be classified independently of the presence of missing values.
More in detail, the impurity of the node is evaluated over the observations which have not missing values for the examined predictor and, once a splitting variable and a split point have been chosen, the missing values are imputed using surrogate variables. These ones are found by re-applying the partitioning algorithm without recursion to predict the two categories defined by the chosen split ($x \leq s$ and $x > s$) using the other independent variables. For each explanatory variable an optimal split point and a misclassification error are computed; the blind rule, that assigns the observation to the majority direction, is also evaluated. Then, surrogates are ranked and those variables which do no better than the blind rule are discarded from the list. At the end, any observation for whom the split variable is missing is classified using the first surrogate variable or, if missing, the second one, and so on. If an observation is missing all the surrogates, the blind rule is used [Therneau et al., 2022].

Advantages and disadvantages of CART
The main advantages of CART are the following:

1. The prediction of the outcome variable is greatly simplified. Once the tree is grown, the correct terminal node (and the corresponding prediction) for a new observation is determined by answering to a small number of dichotomous questions about the values of specific covariates.

2. Its simple structure provides easier interpretation of results, as one can quickly determine those variables that are the most important for prediction of the outcome.

3. They identify effects of covariates within subgroups, in contrast to regression methods that examine effects across the entire learning sample. The result is that trees have greater ability to detect interactions among the variables.

The main disadvantages are that trees generally do not have the same level of predictive accuracy as some of the other regression and classification approaches, and that they are instable, i.e. they can be very sensitive to small changes in the data.

In the following subsection a review of the main papers that reported proposals of how building survival trees has been made to provide a scenario of the development of this algorithm. Summarizing, the aim is the same for all the works presented below, i.e. to find a method that can allow to define homogeneous groups according to their survival behaviour and to define interpretable classification rules. As in CART, the algorithm is similar: a splitting criterion is chosen and used to partition the covariate space, in order to create homogeneous subgroups in terms of survival experience. After a large tree is grown, a pruning technique is used and a final tree is chosen. More in detail, in the papers presented here, both within and between node criteria were proposed as splitting criteria. Then, on the basis of the splitting measure used, different pruning algorithms were proposed. Some of the authors who used within node homogeneity managed to use the CART cost-complexity algorithms [Davis and Anderson, 1989, LeBlanc and Crowley, 1992, Molinaro et al., 2004, Lostritto et al., 2012]; the others proposed alternative methods, as the maximum split statistic used by Segal [1988] and the split-complexity algorithm [LeBlanc and Crowley, 1993].

3.1.2 Review of the literature on survival trees

The first authors who provided a proposal of algorithm for building survival trees were Gordon and Olshen in 1985 [Gordon and Olshen, 1985]. They proposed a splitting criterion based on the use of distances between estimated survival functions in the candidate left and right children nodes, with the aim to force each node to be more homogeneous in terms of survival than its predecessors. More in detail, they used the

Kaplan-Meier survival estimator and L^p Wassertein metrics. This metrics allows to measure the distance between two distribution functions F_Z and F_W and it is defined as:

$$L^p = min\left[\mathbb{E}\big(|Z-W|^p\big)\right]^{\frac{1}{p}} \text{ , with } p \geq 1 \text{ ,}$$

computed for all Z and W defined on the same probability space and distributed respectively as F_Z and F_W.
In particular, they measured the node impurity looking at the distance between the within-node K-M survival curve and a K-M curve for a homogeneous node. A node can be considered homogeneous if one of the following conditions occurs: (i) all the observations are censored; (ii) all the subjects experience the event of interest at the same time; (iii) all the events occur at the same time and then only censored times are observed [Zhang and Singer, 2010].
Then, given a node R_j of a tree and a K-M estimate $\hat{S}^j_{K-M}$ computed from data at that node, the split that maximizes the expression below is chosen to be the best one:

$$\hat{P}(j)d_2(\hat{S}^j_{K-M}, \delta_{\hat{S}^j_{K-M}}) - [\hat{P}(jl)d_2(\hat{S}^{jl}_{K-M}, \delta_{\hat{S}^{jl}_{K-M}}) + \hat{P}(jr)d_2(\hat{S}^{jr}_{K-M}, \delta_{\hat{S}^{jr}_{K-M}})] \text{ .}$$

The quantities that appear in the proposed splitting criterion are the following:

- $\hat{P}(j)$ is the empirical probability of the region of the predictor space corresponding to the node R_j, that is $\hat{P}(j) = \frac{n_j}{N}$, with n_j the number of observations in node R_j;
- jl and jr are the indexes of the left and right children nodes of R_j, i.e. $R_{jl}(k,s)$ and $R_{jr}(k,s)$;
- $d_2(\hat{S}^j_{K-M}, \delta_{\hat{S}^j_{K-M}})$ is the L^2 Wassertein metrics between $\hat{S}^j_{K-M}$ and $\delta_{\hat{S}^j_{K-M}}$, which is a survival function with mass on at most one finite point and that minimizes the Wassertein distance to $\hat{S}^j_{K-M}$. It represents a measure of the variability of $\hat{S}^j_{K-M}$.

So, this splitting criterion maximizes the difference between average variability in children nodes and variability in the parent node.
Then, for pruning the full-grown tree $\mathcal{T}$, the authors proposed the following measure, that is similar to cost-complexity:

$$\left[\sum_{j \in J_{\tilde{\mathcal{T}}}} \hat{P}(j)d_2(\hat{S}^j_{K-M}, \delta_{\hat{S}^j_{K-M}})\right] + \alpha|J_{\tilde{\mathcal{T}}}| \text{ ,}$$

where the first term is a measure of heterogeneity of the terminal nodes R_j, with $j \in J_{\tilde{\mathcal{T}}}$, and the second term is, as in CART, a penalty for lack of parsimony of the tree. The probability weights $\hat{P}(j)$ guarantee that more popular nodes have more importance in the overall assessment of the tree.

The method proposed by Gordon and Olshen did not gain much popularity but, nonetheless, they are those who laid ground for the following works [Bou-Hamad et al., 2011b]. Indeed, in their paper they also mentioned the possibility of using the log-rank statistic or a parametric likelihood ratio statistic to measure the distance between the two children nodes.

The ideas suggested by Gordon and Olshen were subsequently used for the first time by Ciampi et al. [1986]. In their work the authors proposed using the log-rank statistic to compare the groups determined by two children nodes. More in detail, they suggested to compute the log-rank test statistic for each covariate X_k for comparing the survival curves of the two groups defined by the split of the given variable and to choose, finally, the split with the largest significant test statistic value at a predetermined level α. The split procedure is then repeated for each subpopulation until no covariates produce a significant value of the test statistic. Thus, this method leads to a split which ensures the best separation of the median survival times in the two children nodes [Bou-Hamad et al., 2011b].
Furthermore, the authors pointed out that the survival curves of the terminal nodes may not be necessarily pairwise significantly different, so they suggested to recombine some of the branches. In order to do this, they presented the *amalgamation algorithm*, that consists in computing the log-rank statistic for each pair of terminal nodes and in merging together the two subpopulations with the smallest non-significant statistic (at the level α). The process is then repeated until further recombination is not possible. In this work the authors did not talk about the pruning of the tree.

The following year Ciampi et al. [1987] proposed an alternative method using the likelihood ratio statistic (LRS) - under an assumed model - as a dissimilarity measure between two disjoint populations (children nodes). More specifically, the authors discussed two possibilities: an Exponential model and a Cox PH model. Going into more detail, the LRS can be used to test the null hypothesis that the two populations obtained by the candidate split are equal; the larger the statistic the more dissimilar the two nodes. Then, the admissible split that maximizes the LRS is chosen. The concept of *split admissibility* is introduced to avoid nodes with too few observations. More in detail, the authors defined a split $\{R_{jl}, R_{jr}\}$ as admissible if (i) the two children nodes have at least a given number of observed events or, in alternative, of individuals, and (ii) if the LRS used for testing their difference is significant at a given significance level α.
Besides providing a measure of dissimilarity, the LRS also supplies a natural tool to define homogeneity: under regular conditions it would seem reasonable to consider a population homogeneous if for every possible split the LRS is no more statistically significant at some fixed level α.

So, the tree-growing algorithm proposed by Ciampi et al. [1987] can be summed up as follows. The process starts from the root node representing the whole population; then, the LRS is computed for every covariate and every split and the split admissibility is checked. Afterwards, the homogeneity is tested for all the possible nodes and if all of them are homogeneous the algorithm is stopped, else the node with largest dissimilarity is chosen and all the procedure is repeated.
After having built a large tree, the authors proposed a pruning method to find an "honest" tree, i.e. a tree that does not overfit data. They suggested to use resampling techniques as *jackknife and cross-validation*, *v-fold cross-validation* or *bootstrap*. More in detail, the pruning and tree selection algorithm can be described as follows. Firstly, a sequence of significance levels $\alpha_1 < \alpha_2 < ... < \alpha_n$ is chosen and for each α_h a tree $\mathcal{T}_h$ is built ($\mathcal{T}_1 \leq \mathcal{T}_2 \leq \cdots \leq \mathcal{T}_n$). Then, V samples $D^{(v)}$, with $v = 1, 2, \ldots, V$, are built from the original one by using an appropriate resampling technique. For each tree $\mathcal{T}_h$ the corresponding adequacy $A_h^{(v)}$ (measure of global dissimilarity obtained through the generalization of LRS to more than 2 populations) is evaluated for all the $D^{(v)}$ samples. Finally, the mean and standard deviation of $A_h^{(v)}$ over all the V samples are evaluated and the tree corresponding to the maximum average adequacy, or the simplest tree among those for which the mean of A_h falls within l SD's from the "best" tree (usually $l = 1, 2, 3$), is chosen. Ciampi et al. [1987] also provided an alternative to simplify, from a computational point of view, the above algorithm. In particular, they proposed to use the Akaike Information Criterion (AIC) instead of $mean(A_h^{(v)})$, selecting the tree minimizing the AIC (the number of parameters in the AIC expression corresponds to the number of terminal nodes of the tree).

In 1988, Segal [Segal, 1988] adopted a between–node separation approach using a dissimilarity measure, instead of a within-node homogeneity measure as in CART. His proposal was to use the Tarone–Ware class of two–sample statistics for censored data. This class of statistics derives from a sequence of 2×2 tables (as that represented in Table 4, Subsection 2.1.4.1) and it has the following form (using the same notation of Subsection 2.1.4.1):

$$TW = \frac{\sum_{j=1}^{r} w_j [d_{1j} - e_{1j}]}{[\sum_{j=1}^{r} w_j^2 v_{1j}]^{\frac{1}{2}}} .$$

The constants w_j are used to weight the respective table (one for each of the r uncensored observation). Different choices of weights lead to different test statistics; for example, if $w_j = 1$ the TW corresponds to the log-rank test, and if $w_j = n_j$ the Gehan statistic is given.
To prune the tree Segal proposed a different method from the cost-complexity algorithm used for CART, due to the absence of a within-node loss. After a large tree is grown, the proposed pruning algorithm works as follows:

(i) From the bottom, the tree is stepped up, and, to each internal node the *maximum* *split statistic* contained in the subtree of which the node under consideration is the root, is assigned;

(ii) All these maxima values are collected together and sorted in increasing order;

(iii) The first pruned tree of the sequence is obtained identifying the highest node in the tree with the smallest maximum and removing all its descendants;

(iv) The second tree of the sequence is then obtained by reapplying this process to the first tree and so on until only the root node remains.

Davis and Anderson [1989] based, instead, all their work on an Exponential model. To take into account the recursive partitioning structure of the trees the authors modified the usual Exponential model (constant hazard function over time) as follows:

$$h_i(t) = \lambda_j \ \ \forall i \in R_j \ .$$

As splitting criterion the authors proposed to use the exponential log-likelihood loss function, selecting among all the possible binary splits the one that minimizes this loss. More in details, the proposed loss function for the j^{th} node is:

$$Q(R_j) = -\hat{L}(R_j) = D_j - D_j \ln \frac{D_j}{T_j} \ ,$$

where $D_j = \sum_{i \in R_j} d_i$ is the number of complete observations (events) at the node R_j and $T_j = \sum_{i \in R_j} t_i$ is the total observation time. This splitting rule was equivalent to the LRS dissimilarity measure under the Exponential model suggested by Ciampi et al. [1987].
Then, the estimate of the hazard within each node, i.e. $\hat{\lambda}_j$, is used as predicted value at each given node. This measure is an easily interpretable statistic that is useful for predicting the outcome for other individuals with the same characteristics defined by the j^{th} node.
As a stopping criterion the authors suggested to use a minimum node size of at least ten, twenty subjects. Once a full-grown tree $\mathcal{T}$ is obtained, Davis and Anderson used the CART cost-complexity algorithm to prune it:

$$Q_\alpha(\mathcal{T}) = Q(\mathcal{T}) + \alpha |J_{\tilde{\mathcal{T}}}| \ ,$$

with $Q(\mathcal{T})$ the sum of the resubstitution loss over the terminal nodes of $\mathcal{T}$, α the complexity parameter and $|J_{\tilde{\mathcal{T}}}|$, as usual, the number of terminal nodes. Therefore, as in CART, a nested sequence of subtrees is obtained by minimizing the cost-complexity algorithm and cross-validation is used to estimate the loss associated to each tree in the sequence.

The authors pointed out that some problems could arise in the application of this method to survival data if a grown tree contains a node with only censored observations, and therefore the corresponding hazard estimate is set equal to zero. Indeed, supposing that the validation sample includes, for that node, an observation that experiences the event, the likelihood under $\hat{\lambda}_j = 0$ will be zero. This results in an infinite contribution to the cross-validation estimate of loss within the node, $Q^{cv}(R_j)$, and, as a consequence, all the trees including this node would be rejected. Even if at first sight this seems right (a tree that identifies as no at risk a subject that fails is rejected), in this context this may be a problem because, usually, the observation of a group with no events does not indicate a null hazard, but rather a lower hazard relative to the observation time. Therefore, assuming that a zero hazard is not admissible, Davis and Anderson proposed to use $\lambda^* = \frac{1}{2T}$ as adjustment for such nodes to avoid the possibility of zero likelihoods [Davis and Anderson, 1989].

Finally, for the choice of the final tree a comparison of the best tree, i.e. the tree with minimum cross-validated loss $\mathcal{T}_{min}$, and each smaller tree in the sequence is carried out: the most parsimonious tree with a predictive capability that is not substantially worse than the minimum loss tree is selected as the final tree. Differently from CART, the authors replaced the one standard error rule [Breiman et al., 1984] with an empiric rule based on the Chi-square distribution (distribution chosen empirically). Each tree $\mathcal{T}_h$ smaller than $\mathcal{T}_{min}$ is compared to $\mathcal{T}_{min}$ through a Chi-square test. The tree $\mathcal{T}_h$ is not considered an acceptable model if :

$$2[Q^{cv}(\mathcal{T}_h) - Q^{cv}(\mathcal{T}_{min})] > \chi^2_{(\gamma),0.95} \ ,$$

where $\chi^2_{(\gamma),0.95}$ is the 95th percentile of the Chi-square distribution with γ degrees of freedom, determined by the difference between the number of splits in $\mathcal{T}_h$ and $\mathcal{T}_{min}$. The choice of using this percentile is arbitrary and other percentiles could be considered. Concluding, the smallest tree not rejected by this procedure is selected as final tree.

After some years, in 1992, LeBlanc and Crowley [LeBlanc and Crowley, 1992] adopted another method that became one of the most popular, since it can be easily implemented in any recursive partitioning software for Poisson trees, such as the `rpart` package of *R* [Therneau et al., 2015]. The authors based their proposal on the PH model, exploiting the connection between the proportional hazards full-likelihood and the Poisson model likelihood.

They used as a splitting criterion the deviance, a within-node error measure that allows the use of the CART cost-complexity, differently from the algorithms of Ciampi et al. [1986] and Segal [1988]. In details, LeBlanc and Crowley suggested to use only the first step of the full-likelihood estimation procedure to grow and prune the tree, and to use the entire estimation procedure only for the final estimates.

This model is based on the proportional hazards model:

$$h_j(t) = h_0(t)\theta_j \ ,$$

holds, where θ_j is a non-negative parameter and $h_j(t)$ and $h_0(t)$ are the hazard and the baseline hazard functions respectively. The term θ_j corresponds to $e^{\boldsymbol{x\beta}}$ and is a single parameter because one covariate at time is used for splitting and the value of that covariate is treated as the same inside each children node [Zhang and Singer, 2010].

Under the PH assumption, the likelihood can be written as follows:

$$L = \prod_{j \in J_{\tilde{\mathfrak{T}}}} \prod_{i \in R_j} h_j(\tau_i)^{\delta_i} e^{-H_j(\tau_i)} = \prod_{j \in J_{\tilde{\mathfrak{T}}}} \prod_{i \in R_j} (\theta_j h_0(\tau_i))^{\delta_i} e^{-H_0(\tau_i)\theta_j} \ ,$$

where $J_{\tilde{\mathfrak{T}}}$ is the set of terminal nodes; R_j is the j^{th} node and (τ_i, δ_i) is the vector of observation time and event indicator for the i^{th} individual. Finally, H_j and $H_0(t)$ are respectively the cumulative hazard and baseline cumulative hazard function for the j^{th} node.
If the baseline cumulative hazard function was known, the maximum likelihood estimate of the relative risk θ_j ($j \in J_{\tilde{\mathfrak{T}}}$) in each terminal node would be easily obtained as

$$\tilde{\theta}_j = \frac{\sum_{i \in R_j} \delta_i}{\sum_{i \in R_j} H_0(\tau_i)} \ , \ \ j \in J_{\tilde{\mathfrak{T}}} \ . \tag{4}$$

However, in practice the cumulative hazard is usually unknown and an estimate has to be used. An estimator of the baseline cumulative hazard is the one proposed by Breslow [Breslow, 1972]. The full estimation iterative procedure is reported below. Firstly the Breslow cumulative hazard estimate at the u^{th} iteration is obtained using the current estimates $\hat{\theta}^u_j$ of θ_j:

$$\hat{H}^u_0(t) = \sum_{i:\tau_i \leq t} \frac{\delta_i}{\sum_{j \in J_{\tilde{\mathfrak{T}}}} \sum_{i:\tau_i \geq t;\ i \in R_j} \hat{\theta}^u_j} \ . \tag{5}$$

Next, the estimate of $\hat{\theta}^{u+1}_j$ is evaluated using the current estimate $\hat{H}^u_0(\tau_i)$ as

$$\hat{\theta}^{u+1}_j = \frac{\sum_{i \in R_j} \delta_i}{\sum_{i \in R_j} \hat{H}^u_0(\tau_i)} \ . \tag{6}$$

The two steps of algorithm are then iterated until convergence; at convergence only the ratios of the estimates $\{\theta_j : j \in J_{\tilde{\mathfrak{T}}}\}$ between nodes, and not the single estimates, are unique.
As already hinted, LeBlanc and Crowley [1992] used only the first iteration of this procedure to grow and prune the tree. In particular, the Nelson-Aalen cumulative hazard estimator, i.e. the Breslow estimator evaluated at $\{\hat{\theta}^u_j = 1 : j \in J_{\tilde{\mathfrak{T}}}\}$, is used. Then, the one-step estimate of θ_j is:

$$\hat{\theta}^1_j = \frac{\sum_{i \in R_j} \delta_i}{\sum_{i \in R_j} \hat{H}^1_0(\tau_i)} \ .$$

This quantity can be interpreted as the ratio between the observed number of events and the expected number of events in node R_j under the assumption of no structure in survival times.
To measure the goodness of fit of the tree the full-likelihood deviance is used. For the j^{th} node the deviance is:

$$Q(R_j) = 2L_j(saturated) - L_j(\tilde{\theta}_j) \ ,$$

where $L_j(saturated)$ is the log-likelihood for the saturated model (model that allows one parameter for each observation) and $L_j(\tilde{\theta}_j)$ is the maximized log-likelihood when the baseline cumulative hazard function $H_0(t)$ is known.
It can be seen that the deviance residual for the i^{th} observation in node j, which is

$$dev_i = 2\left[\delta_i \ln\left(\frac{\delta_i}{H_0(\tau_i)\hat{\theta}_j}\right) - (\delta_i - H_0(\tau_i)\hat{\theta}_j)\right] \ ,$$

is equivalent to the deviance residual based on the Poisson regression model with response δ_i and mean $\mu_i = H_0(\tau_i)\hat{\theta}_j$. Therefore, this equivalence explains the relationship between the proportional hazards full-likelihood and the Poisson model likelihood. In particular, the recursive partitioning procedure evaluates one-step deviance residuals with the Nelson-Aalen estimate for $H_0(t)$ and with the one-step parameter estimates $\{\hat{\theta}_j^1 : j \in J_{\tilde{\mathcal{T}}}\}$.
So, the proposed algorithm splits the covariate space into regions that maximize the reduction in one-step deviance realized by the split. The improvement for a split s at node R_j into children nodes R_{jl} and R_{jr} is

$$Q(s,j) = Q(R_j) - [Q(R_{jl}) + Q(R_{jr})] \ ,$$

with

$$Q(R_j) = \frac{1}{N}\sum_{i \in R_j}\left[\delta_i \ln\left(\frac{\delta_i}{\hat{H}_0^1(\tau_i)\hat{\theta}_j}\right) - (\delta_i - \hat{H}_0^1(\tau_i)\hat{\theta}_j)\right] .$$

As in CART, the binary splitting procedure continues until a full-grown tree is obtained and there are only a few observations in each node. Then, the tree is pruned using the CART cost-complexity algorithm

$$Q_\alpha(\mathcal{T}) = \sum_{j \in J_{\tilde{\mathcal{T}}}} Q(R_j) + \alpha|J_{\tilde{\mathcal{T}}}| \ ,$$

which efficiently yields trees that perform best in terms of residual error (deviance) given their size. The optimally pruned trees are, as in CART, those that minimize the cost-complexity algorithm for given α's, and the smallest optimal pruned tree is typically chosen. The estimation of the one step deviance for the pruned trees is performed by V-fold cross-validation; thus the complexity parameter α that minimizes the average cross-validated deviance residuals for trees over the V cross-validation subsamples is chosen. Even in this case, as in Davis and Anderson [1989], the same problem arose when evaluating the cross-validated estimates for censored data. To solve the issue LeBlanc and Crowley [1992] used the following adjustment for those nodes with only censored observations:

$$\hat{\theta}_j = \frac{1}{2\sum_{i\in R_j} \hat{H}_0^1(\tau_i)} \ .$$

Finally, after having selected the final tree, maximum likelihood estimates of the relative risk between nodes are obtained by iterating Equations (5) and (6).
The authors also proposed, as alternative to the full-likelihood deviance measure, the use of one-step weighted least squares. In particular, they suggested to use the adjusted dependent variable

$$y_i = \frac{\delta_i}{H_0(\tau_i)} \ ,$$

where $H_0(t)$ is replaced with the Nelson estimate $\hat{H}_0^1(t)$, and the weight

$$w_i = \frac{H_0(\tau_i)}{\theta_j} \ . \tag{7}$$

Then, the weighted least squares score for the j^{th} node is

$$Q(R_j) = \frac{1}{N}\sum_{i\in R_j} w_{0i}(y_i - \theta_j)^2 = \frac{1}{N}\sum_{i\in R_j} \frac{(\delta_i - \hat{H}_0^1(\tau_i))^2}{\hat{H}_0^1(\tau_i)} \ ,$$

with w_{0i} the weight function in (7) at $\theta_j = 1$. The value of θ_j that minimizes this quantity corresponds exactly to the maximum likelihood estimator of θ_j (Equation (4)) with $H_0(t)$ substituted for $\hat{H}_0^1(t)$.
Different weights could also be used; further details can be found in LeBlanc and Crowley [1992].

The following year LeBlanc and Crowley [LeBlanc and Crowley, 1993] proposed another method for growing survival trees. The novelty of the work is a new pruning algorithm called *split-complexity*, useful when a between-node measure is used. They built survival trees using the log-rank statistic to measure dissimilarity in survival between two children nodes, as other authors had already done [Ciampi et al., 1986, Segal, 1988]. The numerator of the log-rank test statistic can be expressed as a weighted difference between estimated hazard functions as:

$$V = \int_0^\infty w(u)\frac{N_1(u)N_2(u)}{N_1(u)+N_2(u)}(d\hat{H}_1(u) - d\hat{H}_2(u)) \ ,$$

where $w(\cdot) = 1$, $N_1(u)$ and $N_2(u)$ are the number of individuals at risk in the two examined group at time u, and $\hat{H}_1(u)$ and $\hat{H}_2(u)$ are the Nelson cumulative hazard estimators for the two groups. Other weights, however, could also be chosen to give greater sensitivity to early or late differences. The ratio of this statistic squared, divided by an estimate of its variance, is used as a splitting statistic and to define the tree structure. This statistic can also be represented as a linear function as follows:

$$V(s) = \sum_{i=1}^{N} I[X_{ki} \leq s](\delta_i - \hat{H}_0(\tau_i)) \ ,$$

where s is the split point, X_{ki} is the value of covariate X_k and τ_i is the observed time (survival or censoring) for the i^{th} observation.
Then, the split that maximizes the log-rank test is chosen and the splitting procedure is repeated until a full-grown tree with a small number of observations in each node is obtained.
As already said, however, the innovation of this paper is the method proposed to prune and select the final tree, known as split-complexity, and defined as follows:

$$V_\alpha(\mathcal{T}) = V(\mathcal{T}) - \alpha|J_{\ddot{\mathcal{T}}}| = \sum_{j \in J_{\ddot{\mathcal{T}}}} V(R_j) - \alpha|J_{\ddot{\mathcal{T}}}| \ ,$$

where $V(\mathcal{T})$, the sum over the standardized splitting statistics $V(R_j)$ in the tree $\mathcal{T}$, represents the goodness of split of the tree; and $J_{\ddot{\mathcal{T}}} = J_{\mathcal{T}} - J_{\tilde{\mathcal{T}}}$ is the set of internal nodes of the tree.The negative sign in front of α is due to the fact that the log-rank test is to be maximized, differently from the cost-complexity that is minimized [Zhang and Singer, 2010]. The complexity parameter α governs the trade-off between the size of the tree and the goodness-of-fit ot the model: as α increases the size of the optimal tree decreases. The optimal tree is the smallest subtree that maximizes the split-complexity with respect to α. In particular, as with the cost-complexity measure in CART, there exists a unique subtree that maximizes $V_\alpha(\mathcal{T})$.
This measure has the peculiar property that if the values of the splitting statistics decrease as one moves down any path in the tree, then the tree that maximizes $V_\alpha(\mathcal{T})$ will be the tree obtained by pruning off all branches that have splitting statistics less than α.
The pruning algorithm based on this measure uses the idea of weakest link cutting from the CART cost-complexity algorithm. As α increases, for any non-terminal node R_j of a tree $\mathcal{T}$ there will be a threshold for which the split-complexity for the branch $\mathcal{T}_j$ will become negative. This value is found by simply solving the following equation for α:

$$V_\alpha(\mathcal{T}_j) = V(\mathcal{T}_j) - \alpha|J_{\ddot{\mathcal{T}}_j}| = 0 \ ,$$

where $J_{\ddot{\mathcal{T}}_j} = J_{\mathcal{T}_j} - J_{\tilde{\mathcal{T}}_j}$ is the set of internal nodes of $\mathcal{T}_j$.
The weakest link $\bar{j}$ in $\mathcal{T}$ is the node for which

$$v(\bar{j}) = \min_{j \in \mathcal{T}} v(j) \ ,$$

with $v(j)$, $R_j \in \mathcal{T}$, the following function:

$$v(j) = \begin{cases} \frac{V(\mathcal{T}_j)}{|J_{\tilde{\mathcal{T}}_j}|} & \text{if } j \in J_{\ddot{\mathcal{T}}} = J_{\mathcal{T}} - J_{\tilde{\mathcal{T}}} \\ +\infty & \text{otherwise} \end{cases}.$$

Then, the nested sequence of subtrees $\mathcal{T}_m \prec \cdots \prec \mathcal{T}_2 \prec \mathcal{T}_1 \prec \mathcal{T}$, where $\mathcal{T}_m$ is the root node, and the corresponding sequence of complexity parameters $\infty > \alpha_m > \cdots > \alpha_2 > \alpha_1 > 0$ is obtained by iterating the minimization of $v(j)$ to find the weakest links and the corresponding pruned trees. Just as an example, α_1 is set equal to $v(\bar{j})$ and the corresponding tree $\mathcal{T}_1$ is obtained by pruning off branch $\mathcal{T}_{\bar{j}}$; similarly, $\alpha_2 = v(\bar{j}_1) = \min_{j \in \mathcal{T}_1} v(j)$ and $\mathcal{T}_2$ be the tree obtained by pruning off branch $\mathcal{T}_{\bar{j}_1}$, and so on.
The authors also showed that the split-complexity can be expressed as a linear function of cost-complexity based on the within node cost $Q(R_j)$:

$$Q(R_j) = \begin{cases} \sum_{s \in J_{\mathcal{T}_j} - J_{\tilde{\mathcal{T}}_j}} V(s) & \text{if } j \in J_{\mathcal{T}} - J_{\tilde{\mathcal{T}}} \\ 0 & \text{otherwise} \end{cases},$$

where $\mathcal{T}_j$ is a branch of $\mathcal{T}$ rooted at node R_j. Therefore,

$$V_\alpha(\mathcal{T}') = V(\mathcal{T}') - \alpha(|J_{\mathcal{T}'}| - |J_{\tilde{\mathcal{T}}'}|) = V(\mathcal{T}) - (Q(\mathcal{T}') + \alpha|J_{\tilde{\mathcal{T}}'}|) + \alpha$$

for subtrees $\mathcal{T}'$ of $\mathcal{T}$.
Due to this relationship, any properties of this pruning algorithm can be obtained from that of CART. In particular, it follows that with this algorithm, the best pruned subtree for any penalty α can be obtained.
Finally, the authors showed some techniques that could be used to select the tree size when between-node dissimilarity is used to measure tree performance. Indeed, as the splitting procedure is adaptively based on the survival, the split complexity $V_\alpha(\mathcal{T})$ is larger than it would be with previously chosen slit points and, consequently, the usual distributional results do not hold for the two-sample statistics. To solve this issue the authors proposed three solutions: the use of (i) a training and a test set; (ii) bootstrap; (iii) permutation sampling methods. In the first case, applicable when a large sample is available, the general idea is to use the learning sample to grow and prune the tree and then to use the test sample for estimating the model's performance. In particular, the test sample data is sent down each of the subtrees obtained with pruning and the tree that maximizes $V_{\alpha_c}(\mathcal{T})$ is selected. The α_c value is a penalty chosen for each split; the authors suggested using a value in the range $[2, 4]$ if the test statistic is approximately χ^2_1 distributed in the two-sample case under the null hypothesis (no difference between groups). Using a value of 2 is in the spirit of the *AIC* criterion, while a value of 4 corresponds roughly to use a 0.05 significance level. The second solution, instead, is based on the use of bootstrap samples for

evaluating the overoptimism due to split point optimization (ω) and then choosing the tree that maximizes a bias-corrected estimate of the split-complexity. However, in highly overfitted situations, the bootstrap estimate of ω can be biased. Finally, the last option allows to compare the observed splitting statistic to an estimate of its permutation distribution. This solution, however, suffers for multiple comparisons due to the number of nodes and splits in a tree. Further methodological details can be found in LeBlanc and Crowley [1993].

In 2004, Molinaro et al. argued that all the previous works on survival trees did not use the classical estimation procedure based on the minimization of the expected value of a loss function (the so called risk). Indeed, the full data loss function cannot be evaluated for censored data and, moreover, risk estimators based on only uncensored observations are biased. In the previous works the problem was bypassed by altering the splitting and pruning criteria with approaches specific to right-censored survival data. To this extent, the approach proposed in the paper is different from all the works shown up to now. Actually, Molinaro and coauthors proposed to use the *general estimating function methodology* of Van der Laan and Robins [2003] to replace the full uncensored data loss function with an observed censored data loss function having the same expected value (risk). The aim of the work is, therefore, the definition of an observed data loss function for node splitting, tree pruning and cross-validation performance assessment in the context of censored survival data.
More in detail, the suggested road map provides for (i) the choice of a loss function for the full data structure and the definition of the parameter of interest, that is the minimizer of the expected value of the loss function for the observed data; (ii) the mapping of the full uncensored data loss function into an observed censored one; (iii) the definition of a set of candidate estimators for the parameter of interest on the basis of a sieve of increasing dimension approximating the complete parameter space (e.g. recursive binary partitioning of the covariate space); (iv) the choice of the best estimator among the candidates through cross-validation (based on the unified cross-validation methodology of van der Laan et al. [2006]).

The general estimating function methodology allows to map the full data loss function $L(W, \psi)$, with ψ being the parameter of interest, into an observed data loss function $L(O, \psi|\eta_0)$ with the same risk as the full data loss function. The quantity η_0 denotes the nuisance parameters. This means that it satisfies the following equivalence:

$$\mathbb{E}_{P_0}[L(o, \psi|\eta_0)] = \int L(o, \psi|\eta_0) dP_0(o)) = \int L(w, \psi) dF_{W,0}(w) = \mathbb{E}_{F_{(W,0)}}[L(W, \psi)] \ .$$

Possible estimating functions are the *inverse probability censoring weighted loss function* (IPCW) and the *doubly robust inverse probability of censoring weighted loss function* (DR-IPCW). A third more parametric approach can also be used estimating the full data risk by maximum likelihood.

In this paper the authors mainly focused on the IPCW, that weights the full data function by the inverse of a censoring probability, i.e.

$$L(O, \psi|\eta_0) = L(W, \psi)\frac{\delta}{\bar{G}_0(T|W)} \; ,$$

with nuisance parameter $\eta_0 = G_0$, that is the conditional survival function for the censoring time C given full data W, $\bar{G}_0(c|W) = Pr(C > c|W)$. It can be proved that this function has the same risk of the full data loss function [Molinaro et al., 2004]. It can also be noticed that in absence of censoring, i.e. when $\delta = 1$ and $C = \infty$, the observed data loss function coincides with the full data loss function.
The corresponding risk estimator is the empirical mean, that is

$$\hat{\theta}_N = \frac{1}{N}\sum_{i=1}^{N} L(O_i, \psi|\eta_N) = \frac{1}{N}\sum_{i=1}^{N} L(W_i, \psi)\frac{\delta_i}{\bar{G}_N(T_i|W_i)} \; ,$$

where η_N represents $\bar{G}_N$, an estimator of the nuisance parameter $\bar{G}_0$ derived under the Coarsening At Random (CAR) assumption (for further details see [Molinaro et al., 2004]).
In the paper the authors examined three prediction problems on the basis of the outcome of interest: univariate outcome prediction, multivariate outcome prediction and density estimation. In the univariate outcome prediction problem the interest is in estimating the log-survival time Z on the basis of a vector of baseline time-independent covariates $\boldsymbol{X}$ (or a subset of it). Two possible related parameters of interest are the conditional expectation $\psi_0(\boldsymbol{X}) = \mathbb{E}_0[Z|\boldsymbol{X}]$ and the conditional median $\psi_0(\boldsymbol{X}) = Me_0[Z|\boldsymbol{X}]$ that can be obtained minimizing respectively the expected values of the following IPCW observed data loss functions:

$$L(O, \psi|\eta_N) = (Z - \psi(\boldsymbol{X}))^2\frac{\delta}{\bar{G}_N(T|W)} \text{ for the squared error full data loss function}$$

and

$$L(O, \psi|\eta_N) = |Z - \psi(\boldsymbol{X})|\frac{\delta}{\bar{G}_N(T|W)} \text{ for the absolute error full data loss function}$$

After having defined the observed loss function on the basis of the parameter of interest, tree-based estimators generate a sequence of candidates using a procedure based on forward selection (node splitting) followed by backward deletion (tree pruning).

For what concerns the splitting procedure, the split that among all possible binary splits maximizes the decrease in empirical risk is chosen as the best one and yields the new index set $\mathcal{T}'$. Using this procedure, a sequence of candidate tree estimators is built. The size of the maximal tree is typically determined by criteria such as minimal terminal node size for continuous outcomes. Then, cross-validation can be applied to estimate risk for each candidate using the observed data loss function and to select

an optimal estimator among these. Finally, once a maximal tree is grown, a minimal cost-complexity pruning algorithm is applied to yield a nested decreasing sequence of subtrees as candidate estimators and cross-validation is used to select the complexity parameter α which minimizes risk.
Finally, for each terminal node, the node coefficients representing the weighted means of the outcome in the terminal node R_j, $j \in J_{\tilde{\mathfrak{T}}}$, can be estimated as the minimizers of the empirical risk in that node.

An issue related to this proposal and underlined by the authors, is related to the choice of the appropriate loss function.

Authors pointed out that trees based on the IPCW loss function can be grown using the `rpart` package, by setting the method argument to `anova` and by providing the IPCW weights for individual observations through the `weights` argument. They also provided a full *R* code in the Supplementary Materials. However, the code doesn't work anymore with the latest versions of *R*.

Two years later Hothorn et al. [2006b] introduced a new proposal that could be applied to all kinds of outcomes, included censored data, to overcome the issues of other recursive partitioning algorithms of (i) overfitting and of (ii) selection bias towards covariates with many possible splits or missing values. More in details, the authors based their algorithm into the theory of permutation tests developed by Strasser and Weber [1999]. So, the focus is on regression models that describe the conditional distribution of the response variable Y (that can also be multivariate) given the set of p covariates by means of tree-structured recursive partitioning [Hothorn et al., 2006b]. So, the basic assumption is that the conditional distribution $D(Y|\boldsymbol{X})$ of the response Y given the covariates $\boldsymbol{X}$ depends on a function f of the covariates.
The recursive binary partitioning algorithm, defined below, is formulated on the basis of a vector of non-negative integer weights w_i indicating the membership of the i^{th} observation to a given node and is based on the idea of dividing the variable and split-point selection in two distinct steps. For notational convenience weights can be assumed to be equal to 1 if the observation belongs to the examined node and 0 otherwise.

Test of Heterogeneity and Variable Selection. For case weights $\boldsymbol{w} = w_1, ..., w_N$ the global null hypothesis of independence between any of the p covariates and the response variable Y is tested. The algorithm is stopped if the null hypothesis cannot be rejected. Otherwise, the covariate X_{k^*} with the strongest association to the outcome is selected.
Splitting Procedure. A subset of the covariate space $A^* \subset \chi_{k^*}$ is chosen in order to split it into two disjoint sets: A^* and $\chi_{k^*} \backslash A^*$ characterized by the weights $w_{left,i} = w_i I(X_{k^*i} \in A^*)$ and $w_{right,i} = w_i I(X_{k^*i} \in \chi_{k^*} \backslash A^*)$.

The two steps are then iterated with modified weights $w_{left,i}$ and $w_{right,i}$. The algorithm is then stopped when the global null hypothesis of independence between the outcome and any of the covariates cannot be rejected at prespecified nominal level α.

More in details, the global null hypothesis H_0 is seen as the combination of p partial null hypotheses:

$$H_0 = \cap_{k=1}^{p} H_0^k \ ,$$

where H_0^k is the k^{th} partial hypothesis testing the independence between Y and X_k, that is:

$$H_0^k : D(Y|X_k) = D(Y) \ .$$

If the global null hypothesis can be rejected, the association between the response variable Y and each of the p covariates is measured through the linear statistics of the form:

$$Z_k(\mathcal{L}_N, \boldsymbol{w}) = \mathsf{vec}\Big(\sum_{i=1}^{N} w_i a_k(X_{ki}) e(Y_i, (Y_1, ..., Y_N))^\top \Big) \ \in \mathbb{R}^{o_k q} \ ,$$

where $\mathcal{L}_N$ is the learning sample, $a_k : \chi_k \rightarrow \mathbb{R}^{o_k}$ is a nonrandom transformation of X_k, and $e : \mathcal{Y} \times \mathcal{Y}^N \rightarrow \mathbb{R}^q$ is the *influence function*, that depends on the responses $(Y_1, ..., Y_N)$ in a permutation symmetric way.
In survival analysis possible choices for the influence function $e(\cdot)$ are the log-rank score or Savage scoring. Then, possible choices for the $a(\cdot)$ function are (i) the identity transformation if the covariate is continuous or (ii) a unit vector with the m^{th} element equal to 1 if the observed covariate assumes the m^{th} level of the categorical variable [Hothorn et al., 2006b].

Afterwards, to proceed with unbiased variable selection procedure, the p-values can be considered. So, the covariate with the minimum p-value (P_k^*) is chosen as splitting variable, that is X_k^*, with $k^* = argmin_{k=1,...,p} P_k$. Once each partial null hypothesis is tested, a global test is required. It can be constructed through aggregation of the transformations a_k, $k = 1, \ldots, p$ [Hothorn et al., 2006b]. Alternatively, an approach based on multiple test procedures based on k p-values is possible (e.g. Bonferroni-adjusted p-values or a min p-value resampling approach). The global null hypothesis is rejected when the minimum adjusted p-value is less than the pre-specified nominal level α. In details p-values can be evaluated exploiting the discoveries of Strasser and Weber [1999] about the conditional distribution of linear statistics, that tends to a multivariate normal distribution as N and the sum of weights w_i goes to ∞ [Hothorn et al., 2006b].

Once a covariate is selected the split can be defined in several ways. One alternative is to use the same permutation test framework used in the first step. In particular, the goodness of a split is evaluated by two sample linear statistics measuring the discrepancy between the two candidate subsets $\{Y_i | w_i > 0 \text{ and } X_{ki} \in A : i = 1, ..., N\}$ and $\{Y_i | w_i > 0 \text{ and } X_{ki} \notin A : i = 1, ..., N\}$.

$$Z_{k^*}^A(\mathcal{L}_N, \boldsymbol{w}) = \mathsf{vec}\left(\sum_{i=1}^{N} w_i I(X_{k^*i} \in A) e(Y_i, (Y_1, ..., Y_N))^\top \right) \in \mathbb{R}^q .$$

Then, the split that maximizes the standardized test statistic is chosen. Finally, a Kaplan-Meier curve for each terminal node can be used for prediction.

For what concerns missing data Hothorn et al. [2006b] pointed out that if an observation has a missing value for the k^{th} covariate, the corresponding case weight is set equal to zero for the computation of the corresponding linear statistic(s) and, once a split is defined, surrogate splits are used.

The parameter α is of great importance in the model. It can be interpreted as a prespecified nominal level of the association tests or as a hyper-parameter linked to the tree size. In the first case it is important to consider that the test procedures have high power only for very specific directions of deviation from dependence, depending on the choice of $a(\cdot)$ and $e(\cdot)$ [Hothorn et al., 2006b]. Hence, an alternative strategy is to increase the value of α for growing a very large tree and then using a smaller value α' to control the size of the tree.

Results from a simulation study carried out by the authors showed that the proposed algorithm is unbiased, i.e. under the null hypothesis of independence between the covariates and the outcome variable, each covariate has the same probability to be selected. Furthermore, under this hypothesis, the proportion of incorrect decisions in the root node is limited by α. Moreover, it emerged that conditional inference trees have a predictive performance equivalent to that of optimally pruned trees, and thus represent a computationally efficient and intuitive solution also to the overfitting problem. Furthermore, the induced partitions are on average closer to the true data partition with respect to an exhaustive search procedure with pruning. Finally, the authors showed that the prediction accuracy of their algorithm is competitive with that of relative risk trees, despite the partitions induced by both algorithms differ structurally and, consequently, the interpretations obtained from the two algorithms cannot be assumed to be equivalent.

This algorithm is implemented in *R* in the `partykit` package [Hothorn and Zeileis, 2015] (oldest version: `party`) [Hothorn et al., 2015a] through the `ctree` function [Hothorn et al., 2015b].

Another interesting proposal, the model-based recursive partitioning algorithm (MOB), was then advanced in 2008 by Zeileis et al. [Zeileis et al., 2008]. Starting from the idea that in many situations it is unreasonable assuming that a single global model fits all observations well, the authors proposed to fit a segmented parametric model by computing a tree in which every leaf is associated with a fitted model. So, this algorithm incorporates parametric models into trees and can be used for many kinds of outcomes (e.g. continuous, categorical, censored). The idea is therefore to partition the observations with respect to a set of covariates such that a well-fitting model, with a node-specific parameter, can be found in each region of the partition. This algorithm is based on fluctuation tests for parameter instability for assessing whether or not splitting each node. More in detail the algorithm works as follows. The model is fitted once to all observations in the current node and the corresponding parameter is estimated by minimizing the model's objective function. Then, the parameter stability is tested for each partitioning variable and, if there is significant instability, the variable with the highest parameter instability is chosen for splitting the node. This very general framework for testing parameter stability is called M-fluctuation test [Zeileis and Hornik, 2007]. Then, the split point(s) that locally optimize the objective function are defined and the splitting procedure is iterated until no more significant instabilities are found. Finally, as already hinted, the final outcome is a tree in which each leaf is associated with a specific model. Further methodological details can be found in Zeileis et al. [2008].

The choice of the variables to consider as regressors or partitioning variables is usually made based on subject knowledge. An example of survival tree is shown in Figure 9. In more detail, for growing a survival tree the authors applied the MOB to a Weibull regression for modeling censored survival times. The model was estimated by maximum likelihood and the instability assessed using a Bonferroni-corrected significance level of $\alpha = 0.05$. The final output is a tree in which each leaf shows survival curves based on the fitted Weibull regression model (see Figure 9).

MOB, as conditional inference trees [Hothorn et al., 2006b], are unbiased because of the separation of variable and split point selection: if the model fitted in a certain node is stable, any of the partitioning variables has a probability of being selected almost equal to α because the overall test is constructed by adjusted Bonferroni p-values from the single parameter instability tests for each partitioning variable [Zeileis et al., 2008]. Finally, for what concerns pruning, the algorithm relies on a statistically motivated internal stopping criterion based on the significance level α (*pre-pruning*). However, if the focus is prediction, cross-validation-based *post-pruning* could be used. Alternatively the tree could be grown with a larger α and then pruned based on information criteria.

MOB are implemented in *R* in the `partykit` package [Hothorn and Zeileis, 2015] through the `mob` function [Zeileis and Hothorn, 2015].

Over the following years many other proposals have been introduced by several authors. Among these there are the the partitioning Deletion-Substitution-Addition algorithm [Lostritto et al., 2012] and the ROC-guided survival trees [Sun et al., 2020a]. The first work inherited the IPCW loss function used by Molinaro et al.

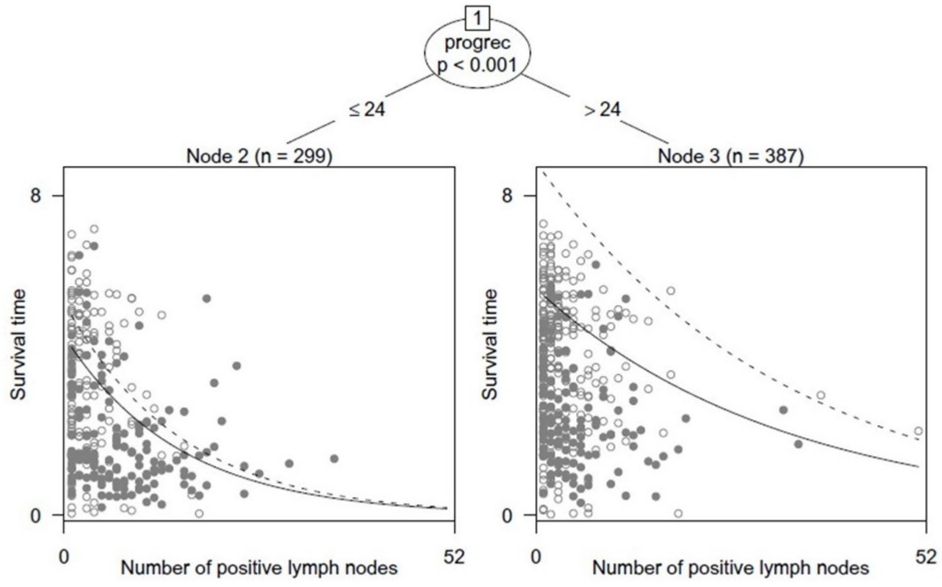


Figure 9: Example of a model-based survival tree. Weibull-regression based tree for the German Breast Cancer Study [Schumacher et al., 1994]. The plots in the leaves show censored (hollow) and uncensored (solid) survival time by number of positive lymph nodes along with fitted median survival for patients with (dashed line) and without (solid line) hormonal therapy. Source: Zeileis et al. [2008]

[2004] and extends CART growing a tree using both "and" and "or" conjunctions of predictors. Thus, distinct subpopulations (defined by the covariates) with the same outcome distribution would be represented using a single partition. The authors also introduced the problem of predicting a binary outcome of the form $I(T > t)$ involving a modification of the squared loss function that uses the Brier Score [Graf et al., 1999] as parameter of interest. The algorithm is implemented in R in the `partDSA` package through the homonymous function.

In the second work, instead, Sun et al. proposed to grow trees using Receiver Operating Characteristic (ROC) curves. More in details, the optimal split is chosen as the one maximizing the increment of the integrated Area Under the Curve (AUC) within a node. Then, for pruning the authors used a *concordance-complexity* measure based on the integrated AUC. ROC-guided survival trees can be grown by using the R function `rocTree` in the homonymous package [Sun et al., 2020b].

Notwithstanding the interesting proposed approaches, these algorithms were not computationally efficient and/or not well documented (see Macis [2022a] reported in Section 5.2).

Moreover, in 2017 Fu and Simonoff extended the relative risk trees [LeBlanc and Crowley, 1992] and conditional inference trees [Hothorn et al., 2006b] to left-truncated data [Fu and Simonoff, 2017a] and to interval-censored data [Fu and Simonoff, 2017b]. Since, our focus was on algorithms for right-censored data these proposals have not been deepened here.

Finally, very recently, Kundu and Ghosh [Kundu and Ghosh, 2021] introduced a new algorithm, the *SurvCART* algorithm, that stands in the same conditional inference framework of the *ctree* algorithm of Hothorn et al. [2006b], but that for some aspects is different from it. These differences concern (i) the parameter instability tests used for growing the tree; (ii) the possibility of also considering censoring heterogeneity. For what concerns the first point, while the algorithm of Hothorn et al. [2006b] uses a nonparametric test and thus it does not require any specification about the distribution of times, the *SurvCART* is based on different parameter instability tests that require a distributional assumption (test statistic of Hjort and Koning [Hjort and Koning, 2002] for continuous covariates and the chi-square test statistic for categorical predictors). With regard to the second point *SurvCART* also allows to consider possible heterogeneity in the censoring distribution. According to this, it could be useful when censoring is independent of survival only conditional on the covariates, i.e. under the assumption of *conditional independent censoring*. In practice, the algorithm works as follows:
Test of Heterogeneity and Variable Selection. For each covariate the parameter instability test is performed to assess heterogeneity in time-to-event and censoring distribution. If no covariates are significantly associated to the outcome the algorithm is stopped, otherwise the partitioning variable with the smallest significant p-value is chosen. If there are multiple partitioning variables the p-values are adjusted, controlling overall type I error at the node level.
Splitting Procedure. For the selected splitting variable all the possible cut-off values are considered. For each of these values the log-rank test (or another alternative test statistic) is evaluated. In particular, if the censoring distribution was found more heterogeneous, the log-rank test is evaluated considering censoring as an event. Then, the value that maximizes the log-rank test is selected as splitting point.

The steps are repeated until the parameter instability tests fail to reject the null hypothesis of homogeneity.

In absence of any prior knowledge the AIC can be used to compare survival trees obtained with different plausible distributions, for choosing the best ones for the time-to-event T and for the censoring time C.

Once a survival tree $\mathcal{T}$ is grown, the AIC can also be used to measure the related improvement:

$$AIC(\mathcal{T}) = 2\sum_{j=1}^{|J_{\tilde{\mathcal{T}}}|}\sum_{i=1}^{N} I(i \in R_j)\, l_i - 2|J_{\tilde{\mathcal{T}}}|dim(\boldsymbol{\theta}) \quad j \in J_{\tilde{\mathcal{T}}}.$$

When only heterogeneity in time-to-event distribution is considered the log-likelihood is $l_i = \delta_i \log f(t_i) + (1-\delta_i)\log S(t_i)$, with $f(\cdot)$ the density function of the event-time T and $S(\cdot)$ is the survival function.
If also heterogeneity in the censoring distribution is taken into account the log-likelihood becomes $l_i = \delta_i[\log f(t_i)e + \log H(t_i)] + (1-\delta_i)[\log S(t_i) + \log g(t_i)]$, where $g(\cdot)$ is the density function of the censoring time C and $H(\cdot)$ is the complementary of the cumulative distribution function of C. In the first case the dimension of θ depends only on the distribution of T, while in the second one it depends on the distribution of both T and C.

Then, the improvement due to $\mathcal{T}$ is measured as the difference between $AIC(\mathcal{T})$ and $AIC(\mathcal{T}_0)$, with $AIC(\mathcal{T}_0)$ being the AIC evaluated at the root node $\mathcal{T}_0$. Moreover, since the overall model fitted at the root node is nested within any survival tree, alternative measures of overall improvement due to $\mathcal{T}$ are the likelihood ratio test and the test for deviance.

Finally, pruning can be performed through the cost-adjusted AIC:

$$AIC_\alpha(\mathcal{T}) = AIC(\mathcal{T}) + \alpha(|J_{\tilde{\mathcal{T}}}| - 1)\;, \alpha > 0\;,$$

with α being the cost for each terminal node. The tree $\mathcal{T}$ is better than the root node (in terms of the cost adjusted AIC) as long as $AIC_\alpha(\mathcal{T}) > AIC(\mathcal{T}_0)$, i.e. when $\alpha < (AIC_\alpha(\mathcal{T}) - AIC(\mathcal{T}_0))/(|J_{\tilde{\mathcal{T}}}| - 1) \equiv \alpha_{\mathcal{T}}^{\max}$. Therefore, the best tree $\mathcal{T}^*$ is the one that offers maximum cost-benefit:

$$\mathcal{T}^* : \alpha_{\mathcal{T}^*}^{\max} \geq \alpha_{\mathcal{T}}^{\max}\; \forall\; \mathcal{T}\;.$$

This algorithm can be implemented in *R* using the `SurvCART` function in the `LongCART` package [Kundu, 2021].

The proposals shown above refer to continuous survival times; however, event times are often measured on a discrete time scale. To this extent, algorithms for discrete survival times have been introduced. Among these, the proposal of Schmid et al. [2016], an extension of the recursive partitioning algorithm introduced by Bou-hamad et al. [2009], is shown below. In particular, the authors introduced a survival tree algorithm for discrete failure times directly based on the binary representation of the likelihood function, exploiting the equivalence of the log-likelihood of a discrete survival model and of a regression model for binary outcome data. In details, the likelihood function is:

$$L = \prod_{i=1}^{N} \prod_{t=1}^{\tau_i} h(t|\boldsymbol{X}_i)^{y_{it}} (1 - h(t|\boldsymbol{X}_i)^{1-y_{it}} \ ,$$

where, for the i^{th} subject, τ_i is, as usual, the minimum between the survival time t_i and the censoring time c_i; $\boldsymbol{X}_i$ is the vector of covariates and y_{it} is the binary outcome that is equal to 1 if the subject experienced the event at time t and 0 otherwise. Finally, $h(\cdot)$ is the hazard function.

The proposed algorithm can be summed up as follows:

1. Specification of the data structure. An augmented dataset is considered. In particular, for each observation a data matrix is built as shown below. If a subject experienced the event ($\delta_i = 1$) at time t_i, the matrix is:

$$\begin{pmatrix} 0 & 1 & \boldsymbol{x}_i^{'} \\ 0 & 2 & \boldsymbol{x}_i^{'} \\ 0 & 3 & \boldsymbol{x}_i^{'} \\ \vdots & \vdots & \vdots \\ 1 & t_i & \boldsymbol{x}_i^{'} \end{pmatrix} .$$

In an analogous way, if the subject is censored ($\delta_i = 0$) at time c_i, the augmented data matrix is:

$$\begin{pmatrix} 0 & 1 & \boldsymbol{x}_i^{'} \\ 0 & 2 & \boldsymbol{x}_i^{'} \\ 0 & 3 & \boldsymbol{x}_i^{'} \\ \vdots & \vdots & \vdots \\ 0 & c_i & \boldsymbol{x}_i^{'} \end{pmatrix} .$$

In the two matrices the three columns provide respectively (i) the y_{it} values; (ii) the time variable (ranging from 1 to τ_i); and (iii) $X_i^{'}$, the vector of observed covariates at each timepoint. The final dataset is then composed of the N matrices defined above.

2. Tree construction and estimation of the hazard function. Due to the binomial structure of the likelihood function, recursive partitioning methods for binary data are applied to the augmented dataset. One peculiarity of this algorithm is that the column of times is also used as an ordinal covariate during the tree construction. As a consequence, the terminal nodes of the tree correspond to both values of X and time intervals.

The authors proposed to apply the CART algorithm based on the Gini criterion (G) to the augmented data. In particular, the Gini impurity measure is chosen due to the asymptotic equivalence to the Brier score (a performance measure described in detail in Chapter 4) and the best split for the j^{th} node is the one that minimizes the weighted sum $n_{jl}G_{jl} + n_{jr}G_{jr}$, where n_{jl} and n_{jr} are the sample sizes in the two candidate children nodes.

Finally, for each node an estimate of the hazard function is obtained as the proportion of observed events in that node. This raw probability can also be corrected, for example by using the Laplace correction [Schmid et al., 2016], when nodes contain a small number of observations.
3. Selection of tuning parameters. The tree is then pruned by using the *cardinality pruning*, i.e. using as tuning parameter the minimum number of observations in the terminal nodes. The optimum minimum node size can be determined through information criteria (AIC, BIC) or using the cross-validated predictive log-likelihood values. In particular, results of a simulation study [Schmid et al., 2016] showed that in a low-dimensional scenario with a small sample size the best method of cardinality pruning was the one based on the BIC, while in the other scenarios the cross-validated predictive log-likelihood is preferable. The results obtained using the AIC instead were suboptimal in all the scenarios.
4. Estimation of conditional survival function. An estimate of the survival function is then obtained as:

$$\hat{S}(t|x) = \prod_{s=1}^{t}(1 - \hat{h}(s|x)) \ .$$

All the survival tree algorithms shown up to now refer to the frequentist approach. Notwithstanding, over the years different Bayesian trees have also been proposed [Clarke and West, 2008, Levine et al., 2014]. However, these approaches have not been considered in this work.

A summary of all the proposals presented above is shown in Table 5.

To my knowledge there still does not exist an algorithm to fit survival trees recognized as the best one. Therefore, further simulation studies are needed in literature for understanding which algorithm is the best to use in different contexts.

Table 5: Summary of the proposals on survival trees

Authors	**Splitting Criterion**	**Pruning**	**Other Notes**	***R* package**	**PH** [2]
Gordon and Olshen [1985]	L^2 Wassertein metric (Within-Node measure)	Cost-complexity	- - -	- - -	No
Ciampi et al. [1986]	Log-rank statistic (Between-Node measure)	- - -	Amalgamation algorithm	- - -	No
Ciampi et al. [1987]	Likelihood ratio statistic (Between-Node measure)	Resampling Techniques	Split admissibility	- - -	No
Segal [1988]	Tarone–Ware class of two–sample statistics (Between-Node measure)	Use of the maximum split statistic	- - -	- - -	No
Davis and Anderson [1989]	Exponential log-likelihood loss (Within-Node measure)	Cost-complexity	Constant hazard in each node	- - -	No
LeBlanc and Crowley [1992]	Deviance (Within-Node measure)	Cost-complexity	- - -	`rpart` (`rpart`)	Yes
LeBlanc and Crowley [1993]	Log-rank statistic (Between-Node measure)	Split-complexity	- - -	- - -	No
Molinaro et al. [2004]	Observed loss function	Cost-complexity	Use of the general estimating function methodology through IPCW	- - -	No

(To be continued)

Authors	**Splitting Criterion**	**Pruning**	**Other Notes**	***R* package**	**PH** [2]
Hothorn et al. [2006b]	Use of conditional inference tests and linear statistics	Internal Stopping Criterium Significance Level	Separation of variable and split selection	`partykit` (`ctree`)	No
Zeileis et al. [2008]	Parameter instability tests	Internal Stopping Criterium Significance Level	Estimation of parametric models in each leaf	`partykit` (`mob`)	No
Lostritto et al. [2012]	IPCW loss function	Cost-complexity	"and" and "or" conjunctions of predictors	`partDSA` (`partDSA`)	No
Schmid et al. [2016]	Gini impurity measure	Cardinality pruning	Discrete survival times	- - -	No
Sun et al. [2020a]	Integrated AUC	Concordance-complexity	- - -	`rocTree` (`rocTree`)	No
Kundu and Ghosh [2021]	Use of parametric tests	Cost-adjusted AIC	- Separation of variable and split selection - Censoring Heterogeneity	`LongCART` (`SurvCART`)	No

[2]Model based uniquely on the Proportionality of Hazards (PH) assumption.

3.2 Random Survival Forests

3.2.1 Rationale

Two of the most important disadvantages of trees are that they do not have the same level of predictive accuracy that other regression and classification approaches have and that they have a high variance [James et al., 2013]. To face this drawback random forests (RF), an ensemble method that uses trees as building blocks, were introduced.

In survival analysis different proposals have been presented for growing random survival forests. Before approaching this extension, a description of how classical random forests work is presented in Subsection 3.2.1.1.

3.2.1.1 Random Forests

Random forests have been introduced by Breiman [Breiman, 2001a] on the basis of the work of Amit and Geman [1997]. A RF consists of a set of trees obtained through bagging and random feature selection. More in details, bootstrap is performed drawing B training sets with replacement from the original one (bagging). Then, for each bootstrapped training set a tree is grown to maximum size using random feature selection and without pruning. Random feature selection means that at each node a random sample of m predictors is chosen from the full set of p predictors ($m < p$) as the set of split candidates; thus, the split is allowed to use only one of the m predictors selected randomly. At each split a new sample of m predictors is taken. Finally, the prediction is made averaging all the predictions for regression trees, or considering the majority vote, i.e. the most occurring class among the B predictions, for classification trees [James et al., 2013].
The size of the set of candidate variables is fixed and typically it is chosen to be $m \approx \sqrt{p}$ [James et al., 2013]. When $m = p$ the method coincides with bagging, that therefore can be considered as a particular case of RF. It can be seen that the use of random features selection overcomes the problem of a small reduction in variance over a single tree due to bagging. In fact, if there was one strong predictor, most of bagged trees would use it in the root node; then the bagged trees would look quite similar to each other and predictions would be highly correlated. Therefore, forcing each split to consider only a subset of predictors, the trees will not be highly correlated and the final estimate will be less variable and more reliable [James et al., 2013].
An alternative to random feature selection, involving random linear combinations of a subset of input variables, has been also proposed by Breiman. More in detail, at a given node m variables are randomly selected and combined together with coefficients uniformly distributed on the interval $[-1, 1]$. Then, f linear combinations are generated and among these the best split is chosen. This procedure is called Forest-RC [Breiman, 2001a].

Accuracy of the algorithm

In addition to the fact that bagging seems to improve the algorithm accuracy when also random features selection is used, the choice of using it in the construction of RF is mainly due to the fact that it can provide ongoing out-of-bag estimates of the generalization error of the combined ensemble trees, as well as estimates for strength and correlation [Breiman, 2001a]. The *generalization error* represents the error rate of the classifier. This quantity, together with the estimate of strength (measure of accuracy) of each tree in the forest and the correlation between any two trees in the RF, is helpful to measure the accuracy of this ensemble algorithm. Results obtained by Breiman showed that better (with lower generalization error) RFs have lower correlation between trees and higher strength. The randomness (and consequently the size m of the predictors subset) used in tree construction has the aim to reduce correlation while maintaining a fair level of strength.

All these estimates are done out-of-bag (OOB), i.e. evaluated on the observations that are left out from a bootstrap training set. More in detail, the response variable for the i^{th} observation is predicted using each of the trees in which that observation was not used for growing the tree (around $B/3$ predictions) and then averaging all that predictions or taking a majority vote. Thus, the advantage is that a test set is not needed, because OOB estimates are as accurate as using a test set of the same size as the training set and because, unlike cross-validation, they are unbiased [Breiman, 2001a].

Variable importance

Variable importance is useful for understanding which variables have the most predictive power. It is measured for all the explanatory variables on the basis of noised data: after each tree is grown, the values of the k^{th} variable in the out-of bag sets are randomly permuted and the permuted data is run down the corresponding tree. Then, the obtained prediction for each OOB observation is recorded and compared with the true observed value to give a measure of error. Variable importance can then be evaluated as the percent increase in the error as compared to the OOB error measured when no variables were permuted; higher values indicate more important variables.

Missing data

When growing a RF the use of surrogate variables is not the best approach for dealing with missing data; indeed, finding a surrogate could become computationally time expensive when the number of grown trees is large; moreover, surrogate variables may not be meaningful because at each node a random subset of variables is selected and, as such, variables within a node may be uncorrelated and a surrogate cannot be found [Ishwaran et al., 2008].

Therefore, an alternative method for imputing missing data when growing a RF has been introduced by Breiman [2003] and Liaw et al. [2002] and it is based on a proximity approach. The data are firstly roughly imputed assigning to each missing value the median or the mode (depending if the variable is continuous or categorical respectively) of non-missing values. The imputed data is used to grow a RF. Then, missing data are imputed using the resulting proximity matrix, a $N \times N$ matrix whose (i, j) entry indicates how many times the i^{th} and j^{th} observations occur within the same terminal node. The proximity weighted average of the non-missing data (for continuous variables) or the integer value having the largest average proximity over non-missing data (for categorical variables) are used to impute missing data [Tang and Ishwaran, 2017]. This data is then used as an input in the random forest and the cycle is iterated until a stable solution is reached.

Advantages and disadvantages of random forests
The advantages of RFs are several. First of all, they have a good accuracy and they are relatively robust to outliers and noise. Moreover, through the use of the Strong Law of Large Numbers, Breiman proved that random forests always converge, so that overfitting is not a problem. Furthermore, they are faster than bagging or boosting. In addition, they provide useful internal estimates of error, strength, correlation and variable importance. Finally, they are simple and easily parallelized [Breiman, 2001a]. On the other hand, since they are like black boxes, the main disadvantage is the model interpretability; then, another disadvantage is the amount of storage that can be used if dealing with large datasets.

Over the years several proposals have been presented for growing random survival forests involving: (i) the algorithm to use for building survival trees, (ii) the outcome used for prediction, (iii) the evaluation of the generalization error, (iv) the variable importance method used and (v) the handling of missing data. The following subsection shows some of the main proposals.

3.2.2 Review of the literature on Random Survival Forests

One of the first works about the use of random forests for survival data is that of Ishwaran et al. [2004]. The method proposed by the authors, called relative risk forests, was based on the application of the algorithm for random forests provided by Breiman [2001a] through the use of relative risk trees introduced by LeBlanc and Crowley [1992]. So, the relative risk forests are based on the implicit assumption of proportional hazards.

The general algorithm for obtaining a relative risk forest is the following:

1. A sample of N observations is drawn with replacement from the original one

(bagging);

2. A subset of m covariates is randomly selected from the full set of p explanatory variables;

3. A full-grown relative risk tree is obtained using the bootstrapped sample obtained in the first step and the covariates selected in the second step. A minimum node size is fixed to ensure that nodes are not too small.

4. A measure of risk relative to an automatically selected mean unit in the study is assigned to each terminal node.

The procedure is then repeated independently B times so that B relative risk trees are obtained. Finally, the random forest estimate, the so-called *ensemble relative risk*, related to a set of covariates $\boldsymbol{x}$ is obtained averaging the B corresponding measures of relative risk. This quantity is usually chosen for its simple interpretation.
An alternative measure is the *ensemble log-relative risk* that is preferable as a measurement for prediction purposes.
Finally, a measure of *ensemble survival probability* can be evaluated as:

$$P(T_i \geq t) = \frac{1}{B}\sum_{b=1}^{B} e^{-\theta_{i,b}\hat{H}_{0,b}(t)} \ ,$$

where T_i is the random variable associated with the survival time, $\theta_{i,b}$ is the relative risk for subject i obtained from the b^{th} relative risk tree and $\hat{H}_{0,b}$ is the estimated baseline cumulative hazard from the same tree.

In the paper, the authors showed that, depending on the aim, a certain level of supervision can be used. In particular, they introduced a *weak supervision* on the set of covariates, forcing the algorithm to include in the random subset of covariates a given variable of interest. This supervision is called weak because it does not guarantee that these variable will be used for growing the tree.
An important step is to ensure a sufficient amount of randomization in the model, determining an appropriate number m of randomly selected covariates: too few covariates would lead to trees that are too sparse and too many covariates would lead to highly correlated trees. For assessing the accuracy of the RF Breiman [2001a] used the generalization error, that involves a trade-off between strength of a tree and trees' correlation. However, with censored data this quantity is not easily computable because the outcome, i.e. the relative risk, is not an observed value. Ishwaran et al. [2004] proposed to estimate the generalization error in the following way. They randomly split the dataset into two subsets of equal size and with an even distribution of events and, for both the datasets, trees and RFs were grown separately and independently. Then, two sets of ensemble log-relative risks can be obtained. These values are then plotted against each other and a value of R^2 is obtained by

regressing the vertical value of the ensemble log-relative risk on the horizontal one. The procedure is repeated many times using a different number of covariates and the best value of m is chosen looking at the R^2 and at the obtained graphs (e.g. having a look at the granularity of the plot). An example is provided in Ishwaran et al. [2004]. An alternative method is based on an OOB analysis, evaluating the two sets of log-relative risk out-of-bag. The procedure is then repeated B times and the values are averaged for obtaining ensemble log-relative risks for the two subsets. Finally, even in this case the best value of m is chosen on the basis of the scatterplot of the two sets of estimates.
Finally, the authors also proposed a method for identifying which explanatory variables were more associated to the ensemble log-relative risk. More in detail, the authors suggested to firstly fit a linear regression model with the ensemble log-relative risk as outcome and to perform variable selection through the stochastic variable selection method (SVS). Variables are then ranked and their absolute posterior means computed and translated into a relative importance value ranging from 0 to 100 by dividing by the largest value, i.e. the value associated with the most informative covariate. Finally, Multivariate Adaptive Regression Splines (MARS) analysis is carried out for studying how the log-relative risk varies with respect to the selected covariates. Further methodological details are available in Ishwaran et al. [2004]. Therefore, the authors also showed that relative risk forests can be used to predict relative risk values using the log-relative risk values as the outcome in a nonparametric regression model.

This algorithm can be implemented in *R* using the `rpart` function in the homonymous package to fit relative risk trees.

A second work on random forests for censored data is that of Hothorn et al. [2006a]. To extend ensemble learning techniques to censored data problems, the authors applied the same framework shown in Molinaro et al. [2004], using the *inverse probability of censoring weights* (IPCW) $\omega_i = \frac{\delta_i}{\bar{G}_N(T|X)}$, with $\bar{G}_N(T|X_i)$ being the estimate of the censoring distribution G (e.g. Kaplan-Meier estimate).

In details, the proposed random survival forest algorithm works as follows. For each replication b ($b = 1, \dots, B$) a random vector of case counts $\boldsymbol{v}_b = v_{b1}, \dots, v_{bN}$ is drawn from the multinomial distribution with parameters N and $(\sum_{i=1}^{N} \omega_i)^{-1}\boldsymbol{\omega}$. The generic element v_{bi} of the vector $\boldsymbol{v}_b$ indicates how many times the i^{th} observation occurs on the perturbed learning sample and $\boldsymbol{\omega}$ is the vector of IPCW. Then, B trees are grown using the learning sample with the corresponding case counts $\boldsymbol{v}_b$ and using, at each split, a subset of randomly selected covariates.
It can be noticed that the algorithm coincides with the original one proposed by Breiman [2001a] when all the events have been observed and there are not censored observations.

The predicted status of the outcome, i.e. the log-survival time for an observation with covariates $\boldsymbol{x}$, is computed on the basis of the prediction weights $\alpha_i(\boldsymbol{x})$ that measure the similarity of $\boldsymbol{x}$ to $\boldsymbol{X}_i$ by counting how many times the value $\boldsymbol{x}$ falls into the same node of the i^{th} observation:

$$\alpha_i(\boldsymbol{x}) = \sum_{b=1}^{B} v_{bi} \sum_{j=1}^{|J_{\tilde{\mathcal{T}}}|} I(\boldsymbol{X}_i \in R_{bj} \text{ and } \boldsymbol{x} \in R_{bj}) \ , i = 1, ..., N \ .$$

Then, the prediction can be computed minimizing the loss function weighted by $\alpha_i(\boldsymbol{x})$. For example, using a quadratic loss, the prediction corresponds to the weighted average of the observed log-survival times:
In this context the full data loss function can be evaluated because, by definition, the weights ω_i and consequently the case counts v_{bi} and the prediction weights $\alpha_i(\boldsymbol{x})$ are zero for censored observations.
The main advantage of this algorithm is that it has a very general form, since different loss functions and other base learners can be easily implemented. However, a practical drawback of this algorithm is that OOB error rate estimates cannot be computed when some observations are given a very large weight and are thus appearing in nearly every bootstrap sample [Hothorn et al., 2006a].

This algorithm can be implemented in the `partykiy` package [Hothorn et al., 2015a] of *R* with the `cforest` function.

Finally, another proposal about random forests has been introduced by Ishwaran et al. [2008]. The authors proposed an algorithm satisfying the prescriptions laid out by Breiman [2003] who required that all the aspects of growing random forests have to take into account the outcome, that in the survival context concerns both the survival time and the censoring status.

The proposed algorithm is basically the same as the original one [Breiman, 2001a]. More in details, B bootstrap samples are drawn with replacement from the original one and, as in classical RFs, at each node of the tree a subset of m randomly selected variables is considered. Then, for each of these samples, a survival tree is grown to full size by recursively partitioning the predictor space through a splitting criterion aimed to maximize the survival difference between children nodes and under the constraint that each node has at least d_0 events. Possible splitting rules are:

- a log-rank splitting rule based on the maximization of the log-rank test statistic (as in Segal [1988] and LeBlanc and Crowley [1993]);
- a conservation-of-events splitting rule that splits nodes by finding children nodes closest to the conservation-of-events principle (defined later on);
- a log-rank score rule that splits nodes using a standardized log-rank statistic;

- a random log-rank splitting rule that uses, among the p candidate variables, the one with maximum log-rank statistic evaluated at a random selected split point.

Finally, for each terminal node R_j the cumulative hazard function (CHF) is evaluated through the Nelson-Aalen estimator; below the CHF for the j^{th} node is reported:

$$\hat{H}_j(t) = \sum_{t_{l,h} \leq t} \frac{d_{l,j}}{n_{l,j}} \; j \in J_{\tilde{\mathfrak{T}}} \, ,$$

with $d_{l,j}$ and $n_{l,j}$ being respectively the number of events and of subjects at risk at time $t_{l,j}$ in the node R_j.
Thus, an observation with covariates $\boldsymbol{x}_i$ falling in the terminal node j has a CHF $H(t|\boldsymbol{x}_i) = \hat{H}_j(t)$.

Once the B trees are grown, an ensemble estimate can be obtained averaging all these estimates. In particular, the authors distinguished two ensemble CHF estimates: the bootstrap and the OOB ensemble cumulative hazard functions. The first one is obtained averaging the CHFs $H^*_b(t|\boldsymbol{x}_i)$ obtained from the B survival trees:

$$H^*_e(t|\boldsymbol{x}_i) = \frac{1}{B} \sum_{b=1}^{B} H^*_b(t|\boldsymbol{x}_i) \, .$$

The second one is instead an average over bootstrap samples in which the i^{th} observation is OOB:

$$H^{**}_e(t|\boldsymbol{x}_i) = \frac{\sum_{b=1}^{B} I_{i,b} H^*_b(t|\boldsymbol{x}_i)}{\sum_{b=1}^{B} I_{i,b}} \, ,$$

where $I_{i,b}$ is equal to 1 if the i^{th} observation is out-of-bag for the b^{th} sample and 0 otherwise.
This approach satisfies the *conservation-of-events principle* that, under fairly general conditions, states that the sum of the CHF over the observed time (both censored and not) equals the total number of events. For a terminal node j this means that

$$\sum_{i=1}^{n_j} H_j(T_{i,j}) = \sum_{i=1)}^{n_j} \delta_{i,j} \; \forall \; R_j, \text{ with } j \in J_{\tilde{\mathfrak{T}}} \, .$$

The principle also holds for the entire tree [Ishwaran et al., 2008].
This principle is then used for defining an alternative predicted outcome, the so-called *ensemble mortality*. This estimate is obtained as the expected value of the CHF summed over time T_l (observed times in the non-bootstrapped data), conditioned on a specific set of covariates $\boldsymbol{x}_i$. So, it can be interpreted as the expected total number of events under the null hypothesis of similar survival behavior:

$$M_i = \mathbb{E}_i(\sum_{l=1}^{N} H(T_l|\boldsymbol{x}_i))$$

where $\mathbb{E}_i$ is the expectation under the null hypothesis that all l are similar to i. This null hypothesis is enforced by the structure of survival trees and of random survival forests because individuals in the same terminal node are assumed to have the same estimated hazard function [Ishwaran et al., 2008]. The correspondent bootstrap and OOB ensemble estimates can be obtained.

The prediction error of the random survival forest is estimated using the Harrell's concordance index C (C-index) [Harrell et al., 1982] that provides information about the percentage of right predictions made by the model (more details have been provided in Chapter 4). In particular, Ishwaran et al. [2008] used as predicted outcome (to be compared to the true survival time) the OOB ensemble CHF. The misclassification rate is then evaluated as the complementary to 1 of the C-index, i.e. $PE^* = 1 - C$. It ranges between 0 and 1 and it assumes a value of 0.5 if the prediction is equal to a random guessing.

The authors also proposed a method for evaluating variable importance (VIMP) for each covariate X_k. In details, the OOB observations are dropped down their in-bag survival tree and, whenever a split for X_k is encountered, the observation is assigned randomly to a children node. Then, the CHF estimate from each of these trees is evaluated and averaged. In the end, the variable importance for the k^{th} covariate is obtained evaluating the difference between the prediction errors for the original ensemble and for the new ensemble estimate obtained randomizing X_k assignments. As usual, large importance values denote variables with predictive ability, whereas zero or negative values identify non-predictive variables. Under the C-index VIMP can be interpreted in terms of misclassification: it measures the increase/decrease in misclassification error on the test data if X_k was not available.

Finally, another point discussed within the paper is the handling of missing data. The authors proposed an alternative approach to the proximity one used in RF by Breiman [2003]. Indeed, this approach may have some disadvantages concerning (i) the bias of OOB estimates of the prediction error (and of all the related measures) and (ii) the impossibility of predicting test data with missing values. To this extent Ishwaran et al. [2008] proposed a new algorithm, the *adaptive tree imputation* that uses only in-bag data for imputing missing data, so that, as a consequence, the OOB prediction error is not optimistically biased. Details have been provided below.
For each node j the set $X^*_{k,j}$ of non-missing values for the k^{th} variable is considered together with its empirical distribution function $\mathbb{P}^*_{k,j}$. Before splitting the node, imputation is made for each in-bag observation within the node drawing randomly a value from the corresponding empirical distribution function. The procedure is then repeated for each variable with missing data. Once imputation has been done, the node is split and the imputed data are reset to missing in the corresponding children nodes.

The procedure is then repeated at each split until the full-grown tree is obtained.

The final summary imputed value for a missing observation is based on the in-bag imputed values from the observation's terminal nodes across the forest. In particular, it corresponds to the average of its imputed in-bag values if the variable is continuous or to the most frequently occurring in-bag imputed value if the variable is categorical (in case of ties, a random tie-breaking rule is used).
The advantages of this imputation algorithm are its ability to predict test data with missing values and the fact that it can also be used when outcomes are missing. However, when there is a high percentage of missing data, its accuracy may decrease. To solve this drawback, the authors proposed to improve it iterating the missing data algorithm. More in detail, after the first cycle of growing the forest, imputation of missing data is performed using OOB summary values and this imputed data is used for growing a new forest. For each observation with originally missing values, a random value is drawn from the non-missing in-bag observations within its terminal node; this procedure is repeated for each tree in the new forest. Finally, the observations are imputed using full summary imputation (in-bag and out-of-bag data) and this re-imputed data are used to grow another forest. The procedure is then repeated iteratively.
The authors' findings suggested that the missing data algorithm can be used if low to moderate missing data are observed; while, with increasing percentages of missing data, iterating the algorithm improves the accuracy of imputed values. Finally, the authors also obtained satisfactory results with the proximity imputation approach [Ishwaran et al., 2008].
Concluding, the authors compared the results obtained with their proposed algorithm to those obtained using the RF approach of Hothorn et al. [2006a] and those attained from a Cox regression model using different datasets. It emerged that in almost all the shown examples their algorithm based on the log-rank and log-rank score splitting rules had the lowest prediction error. Moreover, also the other two splitting rules (conservation of events and random log-rank) had a good performance. In particular, they suggested to use the random log-rank splitting rule when computational speed is needed.

Moreover, they showed that the ensemble regression proposed by Hothorn et al. [2006a] depended strongly on the censoring rate: when the amount of censoring was high the performance was poor. Finally, the stability of forests in the presence of noise variables was confirmed, differently from Cox regression, whose performance decreased as the number of noise variables increased.
This random survival forest algorithm can be implemented using the *R* function `rfsrc` in the `randomForestSRC` package [Ishwaran et al., 2022]. Moreover, recently, the *R* package `ranger` has been created for fast implementation of this algorithm [Wright and Ziegler, 2017].
A summary of all the proposals presented above is shown in Table 6.

Table 6: Summary of the proposals on random survival forests

Authors	**Characteristics**	**Ensemble estimate**	***R* package**
Ishwaran et al. [2004]	1.Use of Relative Risk Trees 2. PH assumption	· Relative Risk · Log-Relative Risk · Survival Probability)	`rpart`
Hothorn et al. [2006a]	1. Weighted random forest regression 2. Use of IPCW 3. Use of prediction weights measuring similarity between obs.	Weighted log-survival time	`partykit` (`cforest`)
Ishwaran et al. [2008]	1. Many possible splitting rules: (i)log-rank; (ii) conservation of events; (iii) log-rank score; (iv) random log-rank. 2. Use of C-index for error prediction 3. Adaptive tree imputation algorithm	· Cumuative Hazard Function · Mortality	`randomForestSRC` (`rfsrc`) & `ranger`

4 Evaluation Metrics

In this chapter some measures used to evaluate the goodness of fit of all the models presented in the previous chapters are shown. Indeed, in this context it is not possible to apply the classical measures used for regression as the mean squared error or the R^2, due to censoring [Wang et al., 2019].

Many alternative measures exist for assessing performance of survival methods. The most used [Rahman et al., 2017] are (i) *discrimination measures* and (ii) *overall performance measures*. The first ones only allow to evaluate the ability of the model to distinguish between low and high risk subjects; on the other side the second ones also allow to quantify calibration (agreement between the observed and the predicted outcomes), evaluating the distance between the 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), its integrated version, the Integrated Brier Score (IBS), the Index of Prediction Accuracy (IPA) - a scaled version of BS - and the Mean Absolute Error (MAE).

4.1 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].
The survival times of two subjects can be compared only in the following two cases:

- both the observations are uncensored (Figure 10a);
- the observed event time of the uncensored observation is smaller than the censoring time of the censored one (Figure 10b).

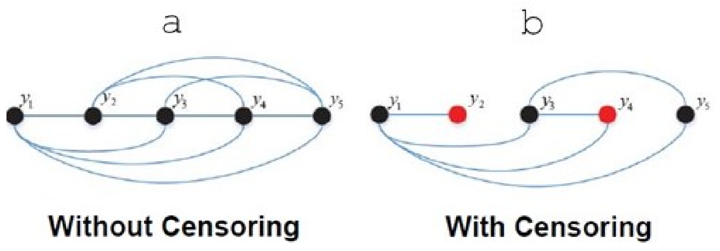


Figure 10: Example of comparable pairs for the evaluation of the C-index. Red and black circles indicate respectively censored and uncensored observations. Source: Wang et al. [2019].

The possible predicted values for each observation are the estimated survival probability $\hat{S}(\cdot)$ or the estimated hazard $\hat{h}(\cdot)$. Considering a comparable pair of subjects (i, j) with actual observed times t_i and t_j, the pair is concordant if $t_i > t_j$ and the i^{th} observation has a better survival prediction than j, i,e. $\hat{S}(t_i) > \hat{S}(t_j)$, or, analogously $\hat{h}(t_i) < \hat{h}(t_j)$. If the opposite occurs, the pair is discordant.

Then, the Harrell's C-index is evaluated as the ratio between the number of concordant pairs and of comparable pairs:

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

or, equivalently:

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

where C is the overall number of comparable pairs, $I[\cdot]$ is the indicator function, δ_i is the binary event indicator, 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 possible drawback of this index it that it does not take into account pairs where the censoring occurs earlier than the event time. To face this issue, a modification to this index has been proposed by Uno et al. [Uno et al., 2011] who used the censoring probability weights; in particular, they used as weight 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:

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

where $\hat{G}(\cdot)$ is the K-M estimator for the censoring distribution.

The above measures are time-independent. A measure of concordance evaluable at each time point has been introduced by Gerds et al. [2013] and corresponds to a time-truncated version of the C-index. It is based on the idea of considering 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 examined 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].

Concluding, the concordance index shows if, and to which extent, the ranking of the occurred events is preserved by the model, and therefore allows to evaluate how many correct predictions have been performed by the model. The index (both Harrell's and Uno's version) ranges between 0 and 1; a value of 0 means perfect disagreement, while a value of 1 indicates perfect concordance. Finally, a value of 0.5 suggests that the model's performance is not better than a random guessing.

4.2 Somer's delta

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

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

This index ranges between -1 and 1 and it is related to the Harrell's C-index as follows:

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

4.3 Time-Dependent ROC Curves and Related Area Under the Curve

Time-dependent Area Under the Curve (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].

For the evaluation of the ROC curve and, consequently, of the AUC at time t, different approaches can be used on the basis of the definition of events (cases) and non-events (controls) [Park et al., 2021]. Then, the ROC curves can be evaluated in a similar way to classical analysis considering the sensitivity and specificity of the model.

1. The *cumulative/dynamic* approach considers as events all the subjects who experienced the event between the interval $[0, t)$ and as controls all those who were event-free at time t.

2. The *incident/dynamic* approach defines cases all the events occurred at time t and controls those who did not experience the event at that timepoint.

3. The *incident/static* approach considers the subjects for which the event occurred at time t as events and those who did not experience it at time t and later until a fixed timepoint as non-events.

So, the AUC indicates the model's performance for discriminating the binary outcome (event/no event) at each 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.

4.4 Brier Score and Integrated Brier Score

The Brier Score is an index firstly introduced by Brier et al. in 1950 and then extended to censored data by Graf et al. [Graf et al., 1999]. It quantifies both discrimination and calibration measuring the distance between observed ($I(T_i > t^*)$) and predicted outcomes ($\hat{S}(t^*|\boldsymbol{X}_i)$) through a quadratic loss function. Since for censored observations the outcome has not been observed, each individual contribution is weighted through the inverse of the respective censoring distribution for compensating loss of information.

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

- Uncensored observations for which the event occurred before t^*, i.e. subjects for which $\delta_i = 1$ and $\tau_i = min(T_i, C_i) = T_i \leq t^*$. For this group the discrepancy between the observed and predicted outcome is $(I(T_i > t^*) - \hat{S}(t^*|\boldsymbol{X}_i))^2 = (0 - \hat{S}(t^*|\boldsymbol{X}_i))^2$.

- Observations for which the event/censoring time occurred after t^*, i.e. subjects for which $\tau_i = min(T_i, C_i) \geq t^* \; \forall \; \delta_i$. In this group $I(T_i > t^*) = 1$ and, therefore, the contribution to the BS is $(1 - \hat{S}(t^*|\boldsymbol{X}_i))^2$.

- 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 this last group the event status at t^* is unknown because censoring occurred earlier than t^* and, consequently, the contribution to BS cannot be evaluated.

Then, the BS at time t^* can be evaluated as follows:

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

Lower values of the index indicate better performance.

If interest is on a given period of time it is possible to integrate the BS over the time interval with respect to a weight function $w(t)$, obtaining the Intgrated Brier Score (IBS):

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

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

4.5 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 KM 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 in terms of predictive performance [Kattan and Gerds, 2018].

4.6 Mean Absolute Error

Finally, in some specific cases, the *Mean Absolute Error* (MAE) can be evaluated. This measure provides an overall assessment of the goodness of the model predictions on the basis of an average of the differences between the predicted times $\hat{t}_i$ and the actual observation times t_i:

$$MAE = \frac{1}{N}\sum_{i)1}^{N}(\delta_i|t_i - \hat{t}_i|) \; .$$

Since it is based on the difference between predicted and observed times it can be evaluated only for survival models which can provide the event time as outcome, as the accelerated failure time models [Wang et al., 2019].

PART II

Novel Contributions

5 Survival Trees with *R*: Computational Issues and Possible Solutions

Survival trees are a useful method for defining homogeneous groups according to their survival probability. In this context, however, there are still some unclear points on how to work with these algorithms 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.

To face these issues two contributions have been presented. The first one mainly aims to disclose the problem and to shed light on the topic [Macis, 2022b] and it is reported in Section 5.1. It has been presented at the SIS2022 - 51st Scientific Meeting of the Italian Statistical Society and it has been published in the conference proceedings. The second one, reported in Section 5.2, provides a practical guide for (i) simulating survival data, (ii) fitting survival trees and (iii) evaluating their performance with the statistical software *R*. In particular, the work was mainly addressed to interested users who want to approach these methods. For each of the three steps the main aspects one should take into account have been shown. In addition, the main available packages have been presented, together with their main characteristics. Then, the focus has been set on the existing drawbacks and on the found solutions. This paper is forthcoming in the *Electronic Journal of Applied Statistical Analysis* (EJASA). Furthermore, a web page (`https://bodai.unibs.it/ml4sd/`) has been created with the aim of providing supplementary materials and updates about the topic. Also, we are still working in the construction of a new *R* package containing the found solutions (some of them already presented in the tutorial paper). Updates will be provided in GitHub and in the above-mentioned web page.

5.1 Recursive Partitioning for Survival Data

Published in the Book of Short Papers of the SIS2022 - 51st Scientific Meeting of the Italian Statistical Society

5.1.1 Introduction

Survival analysis aims to study the occurrence of a specific event (endpoint) during an observed period of time (follow-up) with particular interest on estimation of the survival probability and of the risk of a group of subjects, usually characterized by a set of covariates.

The main 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 time; so, the only known thing about a censored subject is the last time he did not experience the event.

During the years statistical models firstly, and machine learning methods later, have been proposed for analyzing survival data. Among the last ones, survival trees, i.e. tree-based algorithms for censored data, are an interesting tool for defining homogeneous groups according to their survival probability.

Through the years many proposals have been advanced, mainly extending Classification and Regression trees (CART) to survival data (see [Bou-Hamad et al., 2011b]). Most of these works are based on modifications of the splitting criterion and the pruning algorithm using specific tools for right-censored data. Among the most interesting algorithms there are relative risk trees [LeBlanc and Crowley, 1992] and conditional inference trees [Hothorn et al., 2006b, Kundu and Ghosh, 2021]. Other extensions have also been proposed over the years (see Table 7 and [Macis, 2022a]). Despite up to now many proposals have been presented, there are still some unclear points about the best models to use and about how to use these tools practically. A guide for simulating right-censored data, fitting survival trees and evaluating their performance with the statistical software *R* can be found in [Macis, 2022a].

The main aim of this work is to shed light about some of the available packages for fitting and evaluating survival trees for right-censored data, showing the main issues encountered. Particular attention has been given to relative risk trees (RRTs), conditional inference trees based on the *ctree* algorithm (CITs) and conditional inference trees based on the *SurvCART* algorithm (SCTs).

The article is organized as follows. In the next section a brief presentation of the methodological framework is reported. Then, Section 5.1.3 shows the available *R* packages and the related issues. The paper ends with an application on real data.

5.1.2 Survival Trees and Performance Evaluation

Relative Risk Trees. RRTs [LeBlanc and Crowley, 1992] are an extension of CART based on the Cox proportional hazard model. They are equivalent to Poisson trees, due to the connection between the proportional hazards full likelihood and Poisson model likelihood.
In details, the algorithm splits the covariate space into regions that maximize the reduction in one-step deviance realized by the split. Once a large binary tree is grown, it is then pruned through the classical cost-complexity pruning algorithm of CART [LeBlanc and Crowley, 1992]. The node summary is the quantity in (8), that can be interpreted as the ratio between observed and expected number of events in each node under the assumption of no structure in survival times:

$$\theta_k = \frac{\sum_{i \in S_k} \delta_i}{\sum_{i \in S_k} \hat{H}_0^{1}(t_i)} , \tag{8}$$

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]).

Conditional Inference Trees. CITs [Hothorn et al., 2006b] stand on the conditional inference framework. The splitting procedure is divided in variable and split point selection. Firstly, the global null hypothesis of independence between any of the covariates and the outcome variable is tested through a nonparametric permutation test, and the covariate with the strongest association is chosen as splitting variable. Then, the best cut-off is selected through any splitting criterion, e.g. the log-rank test.

Very recently, an alternative way for growing conditional inference survival trees has been proposed [Kundu and Ghosh, 2021]. The *SurvCART* algorithm differs from that of CITs for two main aspects: (i) it incorporates a particular distribution for survival times in the parameter instability tests, and (ii) it allows to also consider censoring heterogeneity. So, SCTs are a powerful tool in presence of *conditionally independent censoring*, i.e. when censoring is independent of the time-to-event distribution only conditional on the set of covariates. To this extent, heterogeneity on both the time-to-event and censoring distributions is tested and the most significant variable is chosen. Then, the best cut-off value is selected, usually through the maximization of log-rank statistic, considering the most heterogeneous distribution. The procedure is then repeated until the null hypothesis of homogeneity cannot be rejected.

For both CITs and SCTs the estimated median survival time and the Kaplan-Meier (K-M) survival curve at each terminal node can be easily obtainable.

Performance Evaluation. The most used measures for evaluating performance of survival methods are the concordance index (C-index), the Area Under the Curve (AUC), the Brier score (BS) and the Integrated BS (IBS). The first two are discrimination measures, while the last two measure at the same time discrimination and calibration.

Both C-index, AUC and BS can be evaluated at a single time point. C-index is a ranking measure based on the comparison of each possible pair of subjects with respect to observed times and predicted survival/hazard and it ranges between 0 and 1. An overall C-index can be obtained through a weighted average; the most used weights are those of Harrell and Uno. AUC also measures the discriminating ability of the model. The main differences with classical ROC analysis are censoring and the fact that the outcome of an observation can change over time.
Finally, BS and IBS quantify through a quadratic loss function the distance between observed and predicted outcome. The lower the value the better the performance. IBS is obtained integrating over the time-period the BS at each time point.
For more details and literature review see [Macis, 2022a].

5.1.3 Growing Survival Trees and Assessing Their Performance in *R*

Many *R* packages allow to fit survival trees. Among these there are `rpart` (with the homonymous function), `partykit` (with the `ctree` function) and `LongCART` (`SurvCART` function) for fitting RRTs, CITs and SCTs respectively. Many other packages exist (see Table 7); however, these are not shown here for the following reasons:

1. `LTRCtrees`, that allows to extend RRTs and CITs to left-truncated and right-censored data, is exactly based on the `rpart` and `ctree` functions. So, in the examined setting (i.e. right-censored data), it provides identical results to the above-mentioned *R* functions;

2. `rocTree` can be used for fitting ROC-guided survival trees. It has been seen that it 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. `partDSA` implements the PDS algorithm; unfortunately there is little documentation for survival trees implementation.

For what concerns RRTs, CITs and SCTs the respective functions are easy to implement. However, some doubts emerge concerning the interpretation of the node outcome for `rpart`. Indeed, 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" means in survival analysis. However, it should represent a measure of risk of the node [LeBlanc and Crowley, 1992]. The other two algorithms, instead, provide very intuitive results. The only drawback for `ctree` is that it is quite complicate to manage the related object. Indeed, there is little documentation about its structure, making it difficult to extrapolate much information about trees structure (useful, for example when performing simulation studies). On the contrary, *SurvCART* is easily manageable and interpretable.

Once a survival tree is grown, its performance can be evaluated. Some of the most used *R* packages are `pec`, `SurvMetrics`, `Hmisc`, `survAUC` and `riskRegression` (see Table 7).

The `pec` package differs from all the others because it requires to transform the fitted tree in a compatible `pec` object through some specific functions; while the others only require a vector of predicted values.

The first issue is that both `pec` and `predict` are not compatible with `SurvCART`. Future research will involve the construction of a new function that allows to evaluate its performance. However, it is still possible to obtain a measure of overall concordance for SCTs extracting, for each observation, the estimated median survival time from the `subj.class` matrix of the `SurvCART` object. Moreover, with `predict` it is not

Table 7: Packages and functions for fitting survival trees for right-censored data (extensions of CART) and evaluating their performance

R package (function)	Algorithm
`rpart` (`rpart`)	Relative risk trees
`partykit` (`ctree`)	Conditional inference trees
`partDSA` (`partDSA`)	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
***R* package (function)**	**Performance Measure**
cindex (pec)	Uno's Concordance index
UnoC (survAUC)	Uno's Concordance index
Cindex (SurvMetrics)	Harrell's Concordance index
rcorr.cens (Hmisc)	Harrell's Concordance index
Score (riskRegression)	Time-dependent AUC[c] with IPCW[d]
AUC.hc (survAUC)	Time-dependent AUC[c]
Brier (SurvMetrics)	Brier Score at a single time-point with Kaplan-Meier IPCW[d]
pec (pec)	Brier Score
IBS (SurvMetrics)	Integrated Brier Score with Kaplan-Meier IPCW[d]
pec (pec)	Integrated Brier Score

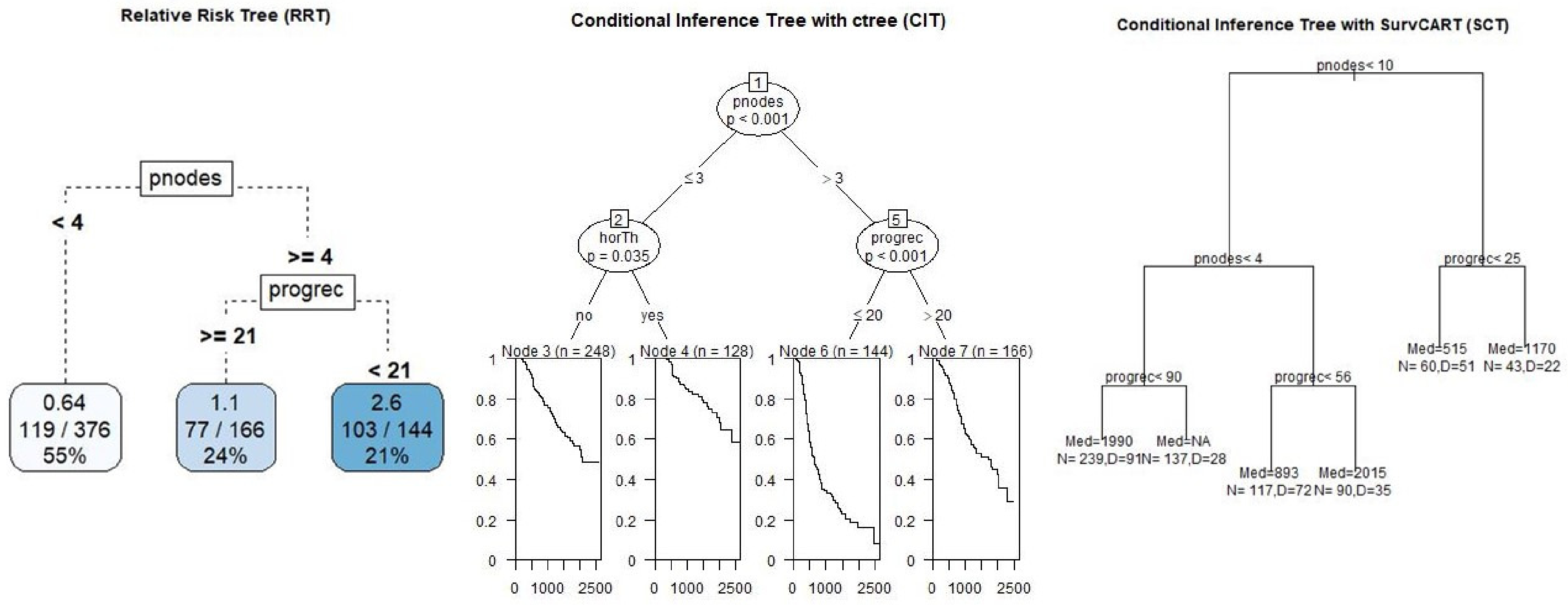
[a] Relative risk trees; [b] Conditional inference trees with *ctree* algorithm; [c] Area Under the Curve; [d] Inverse probability censoring weights

possible to estimate survival probabilities at given time points for all tree algorithms; this issue can be overcome through the use of `pec::predictSurvProb`. However, in some specific cases this function returns NA, making it impossible to evaluate many metrics; future research work will involve this aspect.

Furthermore, also in this step, the main issue concerns little documentation about almost all these functions. Indeed, for many of these, for example, it is not specified if risk or survival predicted values are required. This could imply possible wrong evaluation assessments, since, for example, C-index is based on ranking and survival and risk have an inverse trend. It has been empirically verified, using simulated data, that `Hmisc::rcorr.cens` and `SurvMetrics::Cindex` require survival probabilities, while `survAUC::UnoC` requires a measure of risk. If it is not possible to obtain this measure from trees, a tricky solution is to provide as argument the opposite of survival probabilities, maintaining therefore the right ranking (of course also the vice-versa holds). Moreover, in many cases, it is not declared which weights are used for evaluating C-index. After an in-depth analysis it has been verified that `SurvMetrics::Cindex` and `Hmisc::rcorr.cens` evaluate Harrell's C-index; while `pec::cindex` and `survAUC::UnoC` provide Uno's version.

Time-dependent AUC can be evaluated by using `riskRegression::Score` and `survAUC::AUC.hc`; the two functions usually provide similar results.

Finally, BS can be evaluated with `SurvMetrics::Brier` and `pec::pec`; while IBS can be evaluated by using `SurvMetrics::IBS` and `pec::pec` functions. After the analysis of the `SurvMetrics` functions code (no information was provided in the documentation) it has been verified that it evaluates the index by only using K-M estimator for obtaining the inverse probability censoring weights (IPCWs). Differently, `pec` allows the use of different censoring models.
For both functions an issue related to estimated survival probability was encountered. Indeed, as already hinted, `predictSurvProb` returns NA when only events occurred in a node, implying the impossibility of estimating BS and IBS after that time point. The solution to this problem is deferred to future research work.
A case study. A comparison of the overall performance of RRTs, CITs and SCTs has been carried out on the dataset collected by the German Breast Cancer Study Group [Schumacher et al., 1994]. All the three methods identify *pnodes* as the most important variable, followed by *progrec*. Only CIT also identifies *horTH* as a splitting variable. A higher number of positive nodes, lower levels of the progesterone receptor and not receiving an hormonal therapy lead to a higher risk of recurrence. As already hinted, for SCTs only overall concordance measures could be obtained. In particular, for evaluating C-index for both CIT and SCT, estimated median survival times greater than the length of follow-up were artificially fixed to a maximum value (see [Macis, 2022a]). The three methods lead to similar results both in terms of interpretation and goodness-of-fit (Figure 11).



	RRT	CIT	SCT
Harrel's C	0.686	0.672	0.688
Uno's C	0.614	0.553	0.608
IBS	0.165	0.165	

Figure 11: Performance comparison of RRT, CIT and SCT. Data: German Breast Cancer Study dataset

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

Forthcoming in the Electronic Journal of Applied Statistical Analysis

5.2.1 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 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, 2001b], 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., 2006b, 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, 2017b,a]. 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., 2011b.

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]; therefore, 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., 2006b, 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 5.2.3 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 5.2.4. The paper ends with the final discussion.

5.2.2 Methodological framework

5.2.2.1 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:

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

5.2.2.2 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 (9), 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]:

$$\theta_k = \frac{\sum_{i \in S_k} \delta_i}{\sum_{i \in S_k} \hat{H_0}^1(t_i)} , \tag{9}$$

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).

5.2.2.3 Conditional Inference Trees - `ctree` Algorithm

Conditional inference trees have been firstly proposed by Hothorn et al. in 2006 [Hothorn et al., 2006b] 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., 2006b].

5.2.2.4 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.

5.2.2.5 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:

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

or, equivalently, as:

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

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:

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

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:

$$C_H = \frac{d+1}{2} \ . \tag{10}$$

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].

5.2.3 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.

5.2.3.1 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 8).

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 8: 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 simulating 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 5.2.4).

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 9). Moreover, an example is shown in Figure 12. The two curves represent the survival function for two groups: the dashed line represents the survival probability of a sample with survival time $T \sim Exp(0.4)$

Table 9: 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%

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 13. 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, 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.

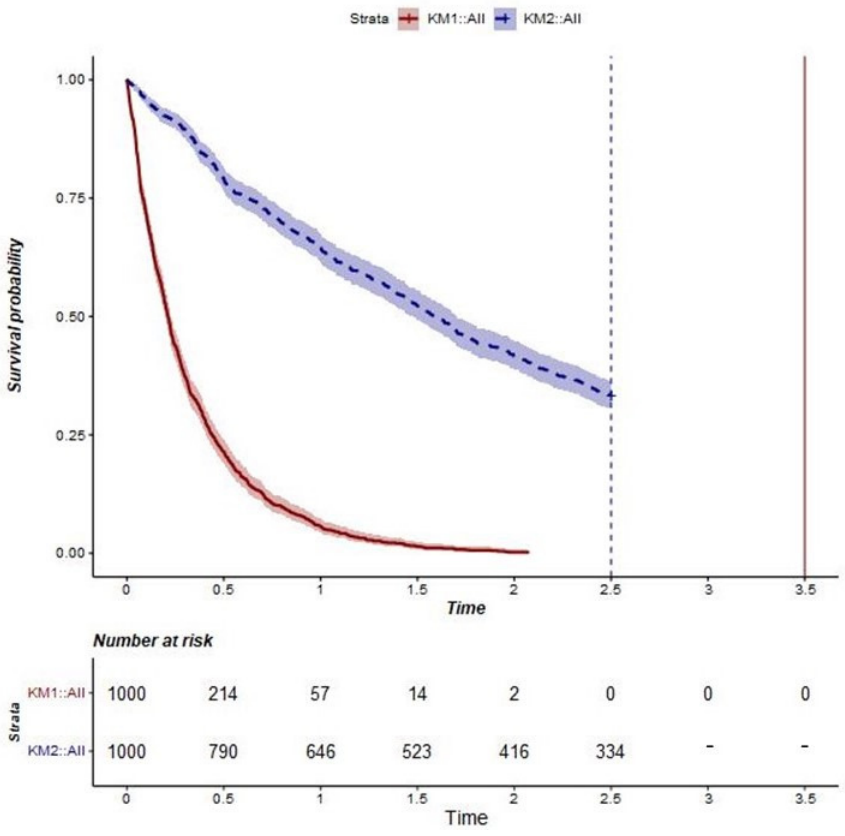


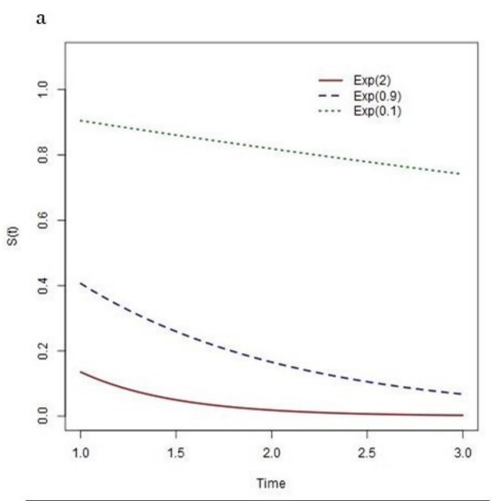
Figure 12: Example of K-M survival curves for two different settings. The dashed vertical line indicates the fixed duration of follow-up (*maxtime*) for the dashed survival curve ($T \sim Exp(\lambda = 0.4), maxtime = 2.5$). Similarly the solid vertical line indicates the fixed *maxtime* for the solid survival curve ($T \sim Exp(\lambda = 3)$, *maxtime*=3.5)

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

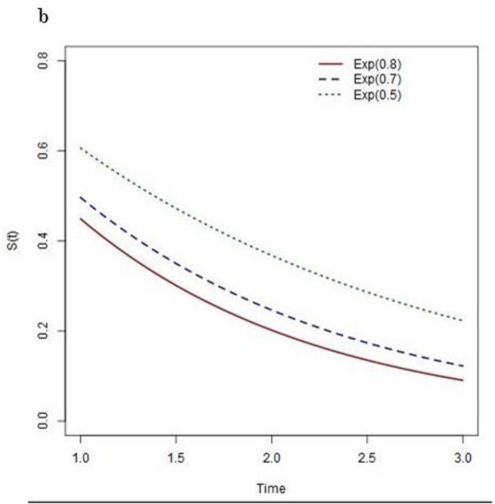
5.2.3.2 Model fitting

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 10). 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., 2006b]. Finally, another noteworthy function is `SurvCART` in the `LongCART` package [Kundu, 2021], which has been introduced very



	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%



	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 13: 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)

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 10: 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 14a). 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 14b). 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 14c. 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.

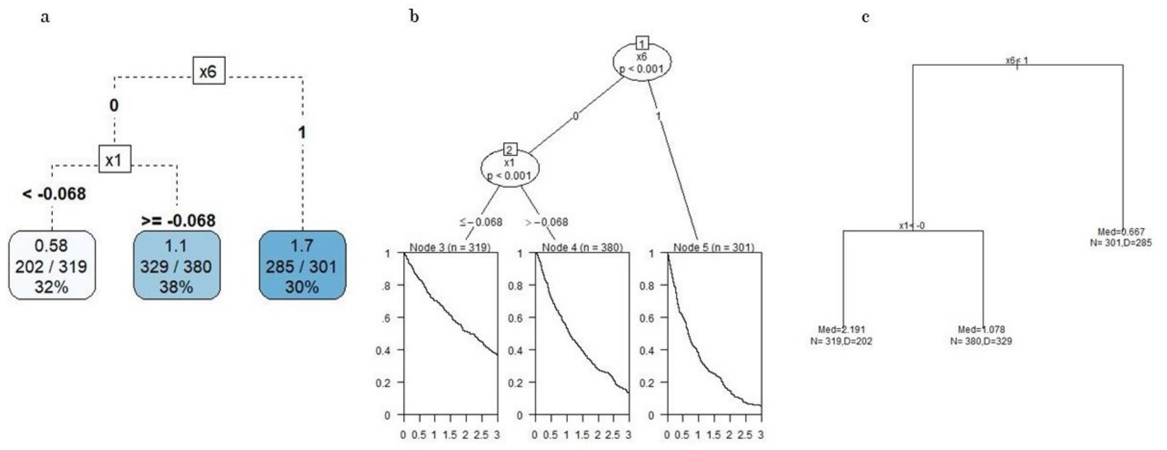


Figure 14: Example of plots of survival trees. a) Plot of a Relative Risk Tree, obtained through the `rpart` function. b) Plot of a Conditional Inference Tree obtained with the `ctree` algorithm. c) Plot of a Conditional Inference Tree obtained with the `SurvCART` algorithm

5.2.3.3 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 11). 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 (10); 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.

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 15. 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).

5.2.4 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/` -

Table 11: 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

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

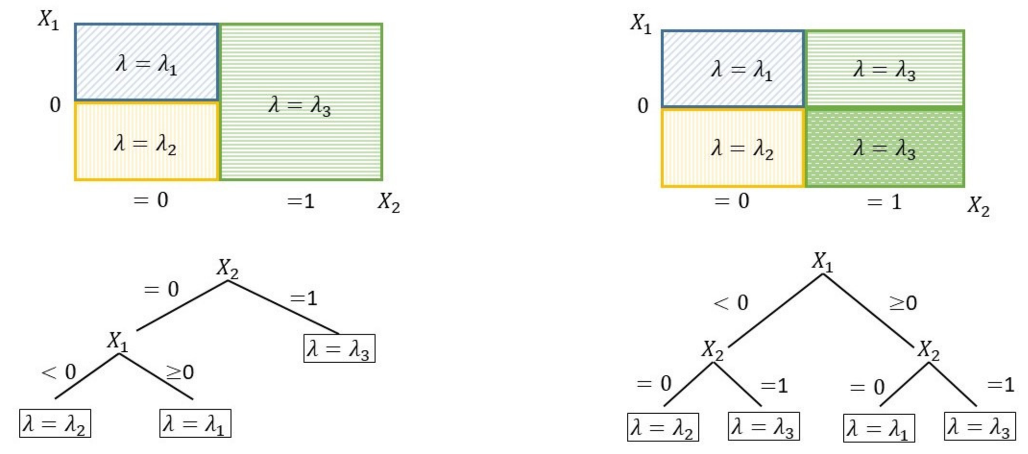


Figure 15: Example of two equivalent trees induced by the same theoretical partition. X_1 is a continuous variable split into two partitions by the cut-off 0; X_2 is a dichotomous variable. Each partition is characterized by a given parameter λ. The two resulting trees have a different structure (different root node, different order of splitting variables, different number of terminal leaves) but they are equivalent

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 5.2.3.3, with these functions when NA occurred in the estimated survival probabilities (see the example in the Supplementary File 1).

5.2.5 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.

5.2.6 Supplementary Materials

Supplementary Materials are available online at `https://bodai.unibs.it/ml4sd/` (reported in the Appendix).

6 Statistical Models of Survival Analysis: Applications to Basketball Analytics

This chapter presents an application of statistical models, in particular Cox Regression and LASSO Cox regression, to basketball analytics. This work was born with the idea of applying survival analysis methods in a new context, different from the classical ones. In particular, the novelty of this work is the application of survival analysis for evaluating players' performance. Indeed, in this field, survival analysis is usually used for studying return to sport or retirement of athletes after injuries [Sochacki et al., 2019, Jack et al., 2019, Mai et al., 2017], or athletes and coaches drop-out [Pion et al., 2015, Moulds et al., 2020, Wangrow et al., 2018]; but, until now it has never been used, to the best of our knowledge, for studying performances. This contribution deals with the analysis of the offensive performance of NBA players in terms of the amount of minutes taken to exceed a given point threshold during the post All-Star season segment, with the main aim of performing variable selection, for identifying which are the players' achievements that significantly impact the outcome.

Section 6.1 reports a short paper presented at the Joint Congress 13th World Congress of Performance Analysis of Sport and the 13th International Symposium on Computer Science in Sport (WCPAS2022 & IACSS2022) that was held in Wien on September 2022. It has been submitted for the Conference Proceedings (Springer) [Macis et al., 2022b].

Then, Section 6.2, a paper that has been submitted for the special issue "Statistics for Performance and Match Analysis in Sports" in Statistica Applicata - Italian Journal of Applied Statistics [Macis et al., 2022a] is reported. It shows two different study settings with two distinguished amounts of censoring.

These works have also been presented at the AUEB Sports Analytics Workshop 2022 in Athens.

Finally, I have been invited for presenting these contributions (together with the one reported in Chapter 7) during the cycle of webinars held by the S-training group (Sports – Training and Research in DAta ScIence Methods for ANalytics and INjury Prevention Group - `https://s-training.eu/`). S-training, managed by Prof. Cristophe Ley (University of Luxembourg), is an inter-disciplinary group of experts in sports science, sports medicine and data science that also collaborates with BDsports (`https://bodai.unibs.it/bdsports/`). It usually organizes monthly webinars dealing with applications in sport analytics.

6.1 Which Achievements Are Associated With a Better Offensive Performance in NBA? A Survival Analysis Study

Submitted in the Proceedings of the WCPAS2022 & IACSS2022 Joint Congress

6.1.1 Introduction

Data analysis for sports has developed consistently over the last years. Several studies aimed to investigate the players' performance or to predict the outcome of a match or of a tournament. Moreover, many studies dealt with the analysis of dropout rate in many sports or the time to return to sport after an injury. In these last studies, survival analysis has been used to analyze not only the factors associated with the event of interest (e.g. drop-out or return to sport), but also the time needed to experience that event. So, survival models can be very interesting when dealing with longitudinal data. To the best of our knowledge, up to now, no studies used this kind of models for analyzing the performance of athletes. This study aims to analyze the offensive performance of NBA players during the 2020-2021 regular season. The offensive performance was measured in terms of overall points gained during the post All-Stars (AS) season segment, and the interest was focused on studying, for each player, the time taken to exceed a given points threshold P during that period, based on a set of baseline covariates, collected just before the AS game. The final aim was to identify which are the main players' achievements that have a significant impact on the outcome.

6.1.2 Methods

Due to the longitudinal nature of data, survival analysis methods have been used; in particular, Cox regression [Cox, 1972] and Lasso Cox regression models [Tibshirani, 1997] were used for identifying the main features associated to the *hazard function*, i.e. the probability of experiencing the event of interest at a given timepoint, given that the subject did not experience it until that timepoint.

The covariates denoting the total number of achievements of each player have been normalized dividing by his minutes played, in order to make them comparable among players. Finally, all the explanatory variables have been standardized (mean 0 and standard deviation 1).

Follow-up (play-by-play) data have been kindly made available by BigDataBall (`www.bigdataball.com`), a reliable source of validated and verified data for NBA, while the covariates have been extracted from the NBA website. Further details on the methods and the study design are provided in Macis et al. [2022a].

6.1.3 Results

The overall sample included 359 players, after the exclusion of those who changed team during the season and those who played less than 48 minutes during the follow-up. The threshold P was fixed equal to 255 points (observed median), ensuring a 50% amount of censoring. The first step of variable selection mainly involved the use of technical knowledge about basketball, excluding variables that are redundant or that are function of other variables, e.g., the number of attempted Field Goals, that is the sum of attempted two- and three-point shots (2PA and 3PA respectively). Thus, only the variables of interest that were suspected to have an impact on the probability of exceeding P were considered. Then, the analysis proceeded with the use of (i) Cox regression through a stepwise selection and (ii) LASSO Cox. The first obtained results are shown in the 2^{nd} and 3^{rd} columns of Table 12. The estimated hazard ratios (HR) suggested that the moves related to defense and game construction (defensive rebounds - DREB, assists - AST, steals – STL, and blocks - BLK) are negatively related to the outcome. However, looking at the results of univariate models it emerged that some of these relationships (with the exception of STL) had been inverted (results not shown here). After having estimated the variance inflation factors (VIF), it was clear that Fantasy Points (FP) induced multicollinearity in the model ($VIF = 9.63$); therefore, this variable was excluded, improving significantly the model (in terms of multicollinearity). The final model and the related results are shown in the 4^{th} and 5^{th} columns of Table 12. It can be observed that DREB, AST and BLK were no more selected by the two models. Results show that the two models identify almost the same variables, with some exceptions (amount of minutes played (MIN), number of free throws attempted (FTA) and percentage of realized two-point shots (2P%)). It emerged that 2PA, 3PA, double doubles (DD2) and the selection for the AS Game have the highest impact on the probability of exceeding P. Figure 16 shows an example: the players who have been selected for playing the AS game (solid blue line) have a higher probability of exceeding P earlier than those who have not been selected (dashed red line).

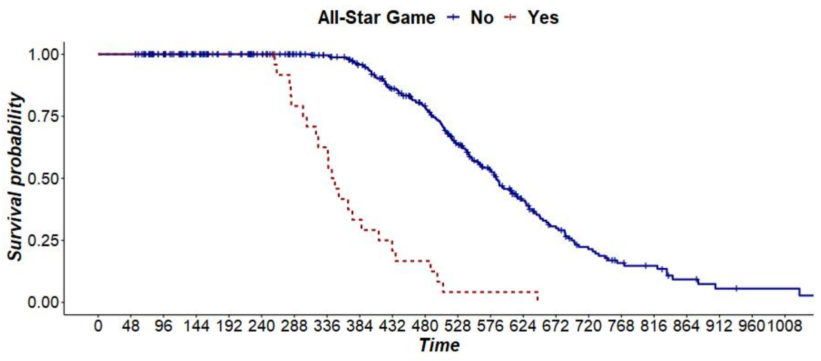


Figure 16: Survival curves of the sample stratified for selection or not for the All-Stars Game (No: solid blue line – Yes: dashed red line)

Finally, models' performance has been evaluated through time-dependent AUC. Both models have a good performance ($AUC(t) > 0.85$) and LASSO slightly overperforms Cox regression (results not shown here).

6.1.4 Conclusions

In this work we have shown a model aimed to understand which are the main variables that affect the probability of exceeding P in a shorter time. It emerged that a higher number of 2PA and 3PA, having been selected for the AS game, having a higher 2P% and gaining more achievements (DD2) increase the probability of exceeding P. Also, MIN and FTA were identified by LASSO, even if with an estimated HR almost equal to 1, suggesting a low impact on the outcome. Moreover, the only variable related to defense and game construction selected by the two models is STL, suggesting that players more involved in defense are penalized in terms of gained points. However, in the end, variables like AST, DREB and BLK have not been selected and, from this point of view, defense seems not to be influential on the scored points [Macis et al., 2022a].

The study has some limitations associated to possible issues related to the assumption of random censoring that may be not respected; however, the assumption of independent censoring can be retained valid [Kleinbaum et al., 2012, Macis et al., 2022a]. Moreover, models' performance may have been overestimated because it has been evaluated in-sample due to the relatively low sample size.

Some improvements will also consider the possible presence of interactions among covariates and nonlinear effects through the use of machine learning methods.

Table 12: Results of the Stepwise and LASSO Cox Regression Models with and without the inclusion of Fantasy Points. The hazard ratios (HR) and the p-values (for Step Cox) are reported.

	With FP		Without FP	
Covariates	Step Cox HR (p)	LASSO HR	Step Cox HR (p)	LASSO HR
Age				
% Won Matches (Player)				
% Won Matches (by the team)				
PTS - Points (Team)				
MIN - Minutes played		1.08		1.08
FTA - Free Throws attempted		1.06		1.07
FT% - % Free Throws made				
2PA - 2 Points Shots attempted	3.51 (< 0.001)	3.59	5.72 (< 0.001)	3.63
2P% - % 2 Points Shots made			1.54 (0.003)	
3PA - 3 Points Shots attempted	2.33 (< 0.001)	2.34	3.53 (< 0.001)	2.35
3P% - % 3 Points Shots made				
OREB - Offensive Rebounds				
DREB - Defensive Rebounds	0.61 (0.007)			
AST - Assists	0.66 (0.003)			
TOV - Turnovers				
STL - Steals	0.64 (0.001)	0.94	0.77 (0.018)	0.95
BLK - Blocks	0.71 (0.039)			
PF - Personal Fouls				
FP - Fantasy Points	2.71 (< 0.001)	1.04	$---$	$---$
DD2 - Double Doubles	1.36 (0.002)	1.20	1.27 (0.003)	1.21
TD3 - Triple Doubles				
$+/-$ - Plus/Minus				
All-Star Game (ref: No)	2.26 (0.006)	1.85	2.27 (0.004)	1.86
NBA G-League (ref: No)				

6.2 A Survival Analysis Study to Discover Which Skills Determine a Higher Scoring in Basketball

Submitted to Statistica Applicata - Italian Journal of Applied Statistics for the special issue "Statistics for Performance and Match Analysis in Sports"

6.2.1 Introduction

Sport analytics has developed consistently over the years. For what concerns basketball, Data Science has been widely used to answer different questions and several studies have been carried out with a wide variety of aims. Among these there are the analysis of players' performance and of the impact of high pressure game situations, the prediction of the outcomes of a game or a tournament, the identification of factors that distinguish successful and unsuccessful teams and the monitoring of playing patterns with reference to roles. These are only some examples; many other studies with different aims have also been carried out [Zuccolotto and Manisera, 2020].

Statistics has also been used for studying the relationship between specific features and dropout of young athletes in many sports as gymnastics [Pion et al., 2015] and swimming [Moulds et al., 2020], or for evaluating the career length of professional basketball players [Fynn and Sonnenschein, 2012]. Other studies, instead, analyzed the criterion determining the decision of a football coach of doing the first substitution during a match [Del Corral et al., 2008]. Furthermore, some studies used statistics for studying the effect of team performance in the dismissal of NBA coaches [Wangrow et al., 2018]. Moreover, some authors investigated the duration of Olympic success and the factors determining national-medal winning at the Olympic Games [Csurilla and Fertő, 2022]. Concluding, other works studied return to sport or retirement of athletes after injuries [Sochacki et al., 2019, Jack et al., 2019, Mai et al., 2017]. The aims of all these studies were investigated using survival analysis, a class of statistical methods devoted to the study of the occurrence of an event during a given observation time.

Up to now, to the best of our knowledge, this kind of analysis has not been used for studies about players' performance. This work aims to study the offensive performance of NBA players, measured in terms of exceeding of a given amount of points during a season segment, and to determine the main achievements related to the occurrence of this event.

The article is organized as follows. The following section reports the methodological framework. Then, Sections 6.2.3 and 6.2.4 show respectively the data used for carrying out the study and the obtained results. The paper ends with the final discussion.

6.2.2 Methods

Survival analysis aims to study the occurrence of a particular event during an observed period of time. The main feature of this kind of data is censoring. A subject is censored when for him/her the event of interest has not been observed during the observation time, so that the only known thing is the last time he/she did not experience the event [Collett, 2015]. In this context a subject is denoted by three elements: (i) a time point τ that can be the observed time t or the censoring time c; (ii) an event indicator δ that equals 1 if the subject experienced the event and 0 if he/she is censored; and (iii) a vector of observed covariates $\boldsymbol{x}$. More in detail:

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

Key elements in survival analysis are the survival and hazard functions. The first one represents the probability that an individual survives (does not experience the event) beyond a given time t and can be defined as

$$S(t) = P(T \geq t) = \int_t^{\infty} f(u)du \; .$$

The hazard function, instead, measures the probability that an individual has the event of interest at time t conditional that the event has not occurred until that time. Formally, it is defined as

$$h(t) = \lim_{\Delta t \to 0} \frac{P(t \leq T < t + \Delta t | T \geq t)}{\Delta t} \; .$$

6.2.2.1 Cox regression model

The Cox proportional hazards (PH) regression model [Cox, 1972] is one of the classical methods most used for analyzing survival data. It allows to estimate the hazard of a subject depending on a set of covariates. The main assumption of the model is the proportionality of hazards, implying that the hazards of two groups of subjects are proportional, so that their ratio is constant over time:

$$\Psi = \frac{h_1(t)}{h_2(t)} \qquad \forall t,$$

where Ψ is a constant called *hazard ratio* (HR) or *relative hazard.*
The Cox PH model can then be expressed as

$$h_i(t) = h_0(t)\Psi = h_0(t)e^{\beta_1 X_{1i} + \beta_2 X_{2i} + \ldots + \beta_p X_{Ki}} = h_0(t)e^{\sum_{k=1}^{K} \beta_k x_{ki}} \; ,$$

where $h_i(t)$ represents the hazard function for the i^{th} subject; $h_0(t)$ is the baseline hazard, that is the risk for a subject whose values of all the independent variables are

equal to zero; x_{ki} is the observed value of the k^{th} covariate for the i^{th} subject and β_k is the related coefficient.
Each coefficient represents the estimated change in the logarithm of the hazard ratio in correspondence of a change of the corresponding covariate. Usually their exponential is considered, measuring the hazard ratio. A value of e^{β_k} greater (lower) than 1 indicates that as the value of the continuous variable X_k is increased by one unit the hazard is increased (decreased) of e^{β_k}, or in an analogous way, if X_k is categorical, that a subject in group k has a higher (lower) hazard (equal to e^{β_k}) relative to a subject in the reference group.

The Cox PH model is called semiparametric because it is based on a nonparametric component (the baseline hazard - no distributional assumption is made for survival times) and on a parametric term (the coefficients vector).

6.2.2.2 Regularized Cox regression model

In high-dimensional data contexts, usually, the interest is on variable selection in order to identify, among the many available variables, the most important ones. Thus, variable selection helps determining all the (informative) variables that are strictly related to the outcome, removing uninformative variables that decrease the precision and increase the complexity of the model. So, variable selection provides a balance between parsimony and goodness of fit of the model. To this extent, regularized models are a good choice because they can allow to obtain a sparse model with many estimated coefficients equal to zero. In particular, they are based on the minimization of a loss function under a constraint that penalizes the flexibility of the model. Also for Cox PH regression model different regularizations have been proposed (e.g. LASSO, Ridge, Elastic Net). Among these the LASSO (Least Absolute Shrinkage and Selection Operation), firstly proposed by Tibshirani in 1997 [Tibshirani, 1997], is one of the most used if the aim is variable selection. This because LASSO is based on the use of a ℓ_1-norm penalty, that allows to obtain a well-defined solution with few nonzero coefficients β_k [Simon et al., 2011]. Therefore, LASSO is advantageous in terms of interpretation of the model and computational convenience [Hastie et al., 2015]. Moreover, it is an interesting and useful method because it simultaneously performs feature selection among all the covariates and estimates the regression coefficients. The only requirement is to have standardized variables.

Regularized parameters can be obtained minimizing the negative log partial likelihood $l(\beta)$ under the constraint that the sum of the absolute values of the parameters is bounded by a constant (LASSO penalty):

$$\hat{\beta} = argmin_{\beta} - l(\beta) \, \text{subject to} \sum_{k=1}^{K} |\beta_k| \leq s \text{ ,with } s > 0.$$

The regularization parameter s is a non-negative tuning parameter that controls the impact of the penalty. The larger the value the lower the amount of shrinkage [Ekman, 2017].

K-fold cross-validation is used for identifying the best parameter s; the optimal regularization parameter is the one that minimizes the cross-validation error. In survival analysis one of the most used performance measures is the Harrell Concordance index [Harrell et al., 1982]. As a higher C-index value means a better performance, the cross-validation error is measured as $1 - C$, that is the complementary to one of the C-index [Tay et al., 2022].

The LASSO technique for variable selection in the Cox model is a worthy competitor to stepwise selection [Tibshirani, 1997], a variable selection procedure usually performed on the basis of the Akaike Information Criterion (AIC). From the simulation studies performed by Tibshirani in 1997 [Tibshirani, 1997] it emerged that LASSO is less variable than stepwise Cox, still yielding interpretable models.

Analyses have been performed by using *R* [R Core Team, 2021]. In particular, `survival` and `glmnet` packages have been used for estimating the Cox and LASSO Cox regression models. Finally, `riskRegression` has been used for evaluating time-dependent AUCs.

6.2.3 Data and study design

NBA data were analyzed. In detail, we considered the 2020-2021 regular season and divided it in two segments, the pre- and the post All-Star game.
A sample of n players has been considered. For each player a set of baseline covariates $\boldsymbol{X} = (X_1, X_2, ..., X_K)$ corresponding to the main achievements gained in the pre All-Star game has been observed. Let's denote with $\boldsymbol{x}_i = (x_{1i}, x_{2i}, ..., x_{Ki})$ the vector of observed baseline covariates for the i^{th} player.
Play-by-play data of the post All-Star game season segment have been analyzed for extracting the needed information relative to the outcome variable. In details, for each player the minutes played until different time points (time referred to the second season segment - $t_1, ..., t_J$) and the corresponding points gained were collected. Let's denote with M_j and P_j respectively the variables relative to the minutes played until time t_j ($j = 1, 2, ..., J$) and the corresponding points gained. For each player we recorded the amount of minutes m_{ij} played at time t_j, and the points p_{ij} gained after having played m_{ij} minutes (see Table 13). So, the time variable was treated as player-time: m_{ij} increases when the player is in the court and remains constant when he is not playing.

Then, we fixed a given threshold P of scored points and we defined the event as the exceeding of that threshold. Censoring occurred when the player did not exceed the fixed amount of points at the end of the post All-Star game regular season segment.

Let's set $j_i^*(P) = \text{argmin}_{j=1,...,J} p_{ij} > P$; the time at which the player exceeds the threshold is therefore $t_{j_i^*(P)}$. Thus, the outcome of the study is composed of (i) the time-to-event $\tau_i = \min\{m_{ij_i^*(P)}, m_{iJ}\}$, where $m_{ij_i^*(P)}$ is the amount of minutes played by the player before reaching the threshold and m_{ij} corresponds to the amount of minutes played at the end of the season segment, and of (ii) the event $\delta_i = I[p_{ij} > P]$, defined as:

$$\delta_i = \begin{cases} 1 & \text{if } j_i^*(P) \in \{1, ..., J\} \text{ and } \tau_i = m_{ij_i^*(P)} \\ 0 & \text{if } j_i^*(P) \notin \{1, ..., J\} \text{ and } \tau_i = m_{iJ} \end{cases}$$

Table 13: Example of collected data. Each row refers to a player, and each column to a time point. In each cell the overall amount of minutes played and points gained until that given time point are recorded

t_1	t_2	$\cdots$	t_J
(m_{11}, p_{11})	(m_{12}, p_{12})	$\cdots$	(m_{1J}, p_{1J})
$\vdots$	$\vdots$	$\ddots$	$\vdots$
(m_{n1}, p_{n1})	(m_{n2}, p_{n2})	$\cdots$	(m_{nJ}, p_{nJ})

The 2020-2021 regular season was analyzed. Play-by-play data have been kindly made available by BigDataBall (`www.bigdataball.com`), a reliable source of validated and verified data for NBA, MLB, NFL and WNBA. Baseline covariates were collected from the NBA website. Moreover, besides players' achievements, also some statistics of the relative team were recorded. Finally, two factor variables were created: All-Star game (if the player was selected for playing at this competition or not) and G-League (if the player also played in the young championship). All the variables included in this study have been chosen to characterize the performance abilities of each player.

6.2.4 Results

The overall sample included 359 players, after having excluded those who changed team during the season and those who played less than 48 minutes in all the post All-Star game season segment. Two distinct cases have been analyzed depending on the percentage of censoring, 20% and 50% respectively, to examine if the covariates have a different impact on the outcome. In the first setting the points' threshold was fixed to 99 points; in the second one, instead, P was fixed equal to 255 points.

The game variables denoting the total number of achievements of each player in the analyzed period have been normalized dividing by his minutes played (to have comparable results). Then, all the covariates have been standardized (to have mean 0 and standard deviation 1).

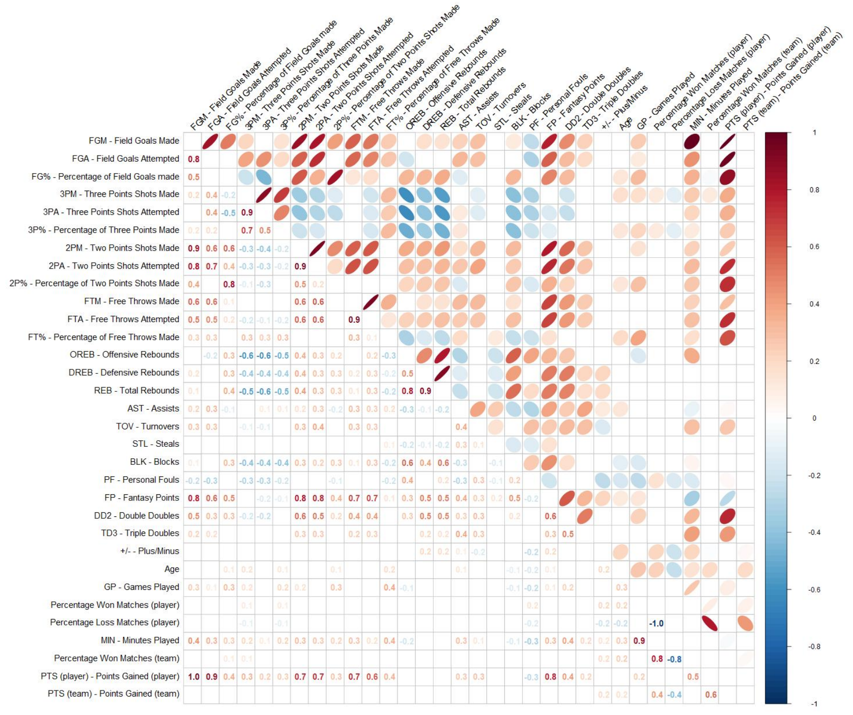
The first step of variable selection was carried out on the basis of prior knowledge and through the examination of correlations analysis. Indeed, considering all the NBA stats would imply a high risk of multicollinearity, due to the presence of highly correlated variables (see Figure 17a). Therefore, we excluded some redundant variables (Figure 17b). So, after this step, the set of baseline covariates passed from 34 to 23. All these variables, as already hinted, refer to the first season segment, i.e. the part of the regular season before the All-Star game. Among the excluded variables there is also the amount of fantasy points, due to the high multicollinearity found in another study [Macis et al., 2022b].

The following step of variable selection involved the use of Cox regression model and its regularized version through LASSO for selecting the most important variables.

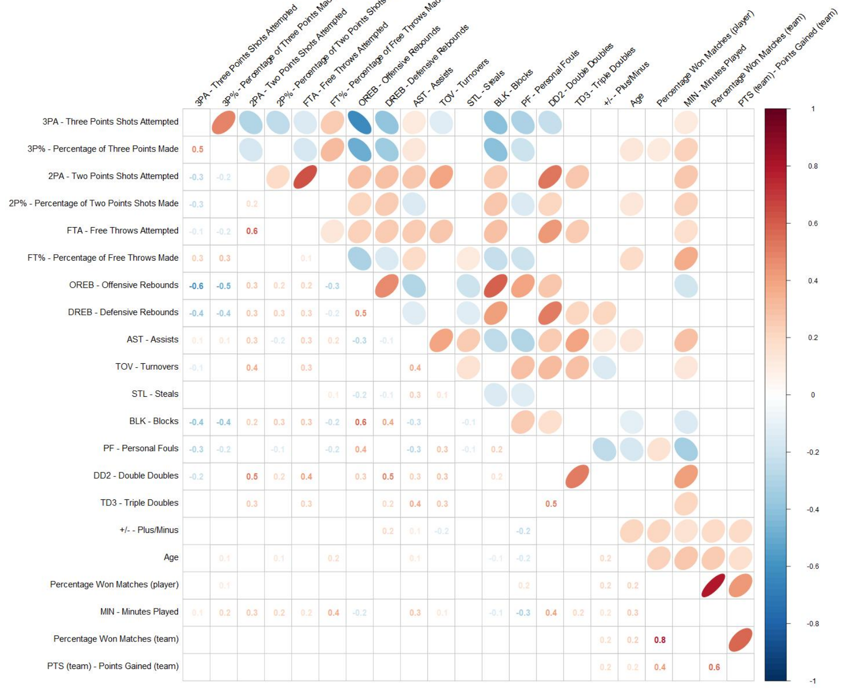
6.2.5 Performance of NBA players

Using as a threshold 99 points, 287 players exceeded the given amount of points (corresponding to the 80% of the sample). Table 14 shows the results obtained fitting the two models. It can be seen that Cox regression and LASSO identified almost the same variables (with the exception of the All-Star game variable). More in detail, both the two models selected the amount of minutes played in the pre All-Star season segment, the number of attempted shots (free throws -FTA-, two -2PA- and three -3PA- pointers), the percentage of two points shots made (2P%) and the number of gained double doubles (DD2). All these variables resulted positively associated with the outcome: an increase in the number of these achievements is associated to a higher probability of exceeding the threshold. Moreover, Cox regression and LASSO Cox also identified the number of steals (STL), even if the estimated hazard ratio was not statistically significant in Cox regression ($p = 0.144$) and that in LASSO it seems that the estimated impact on the outcome was low (HR approximately equal to 1.00). Finally, LASSO also identified the All-Star game variable, even if with an estimated HR approximately equal to 1.

Increasing the threshold to 255 points, only one half (179 players) of the sample exceeded the points' cut-off. In this setting almost all the variables identified in the previous one (with a 20% of censoring) were selected by the two models, even with some differences (Table 14). More in details, the amount of minutes played in the previous season segment and the number of FTA were only identified by LASSO with an estimated HR near to 1.00, suggesting a lower impact on the outcome. The 2P% instead, was only identified by Cox regression. Moreover, increasing the threshold of points, the number of STL was found negatively associated with the outcome. The estimated coefficients resulted lower than 1 (equal to 0.77 and 0.95 in Stepwise Cox and Lasso respectively), indicating that this move is negatively associated with the outcome: a unit increase of this achievement leads to a lower probability of exceeding the threshold. Finally, with a higher threshold the All-Star game resulted to be a



(a) Entire Set of Covariates



(b) Definitive Set of Covariates

Figure 17: a) Correlation plot of all the covariates. b) Correlation plot after variable selection. The game variables denoting the total number of achievements of each player have been normalized dividing by his minutes played; all the covariates have been standardized. Ellipses towards left (right) indicate negative (positive) correlations

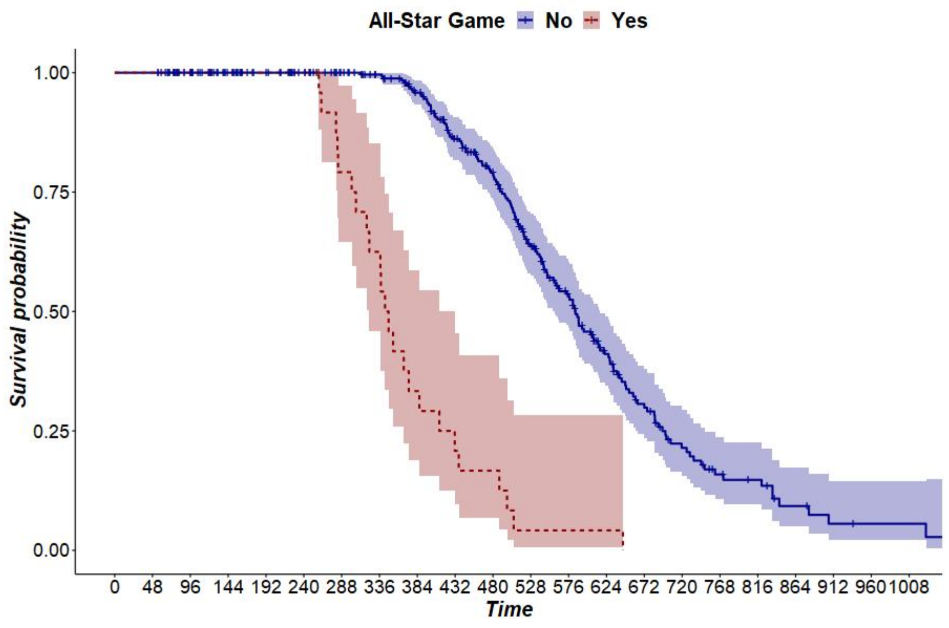
relevant feature identified by both the two models: having been selected for playing at the All-Star game doubles the probability of gaining more than 255 points.

Interestingly, most of the variables identified in this setting have a higher effect (higher HR) than in the setting with a lower threshold.

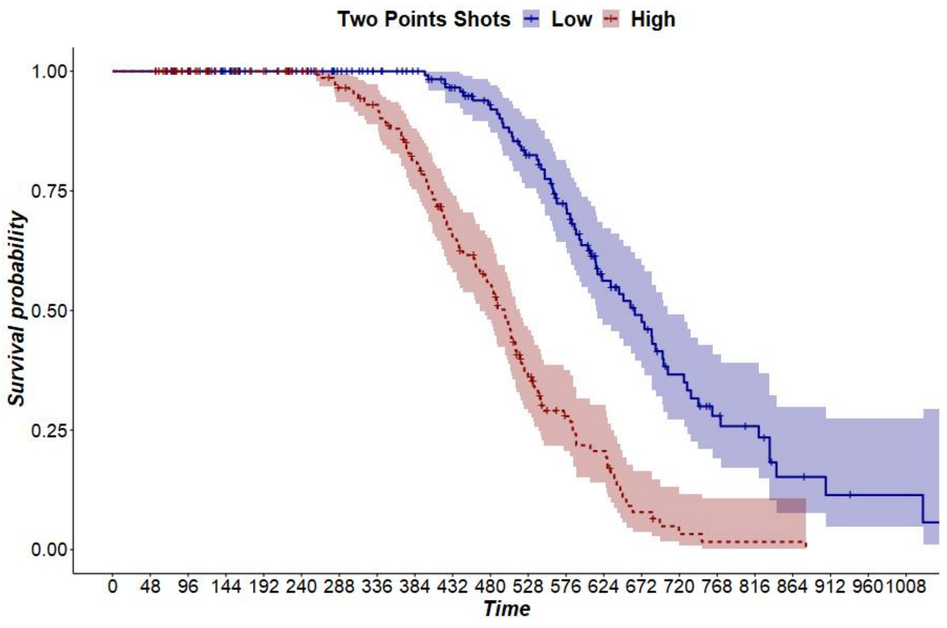
Table 14: Results of the variables selection procedure in both the two settings (20% and 50% of censoring)

	20% censoring		50% censoring	
Variables Included in the Model	Stepwise HR (p)	LASSO HR	Stepwise HR (p)	LASSO HR
3PA - 3 Points Shots attempted	2.80 (< 0.001)	2.16	3.53 (< 0.001)	2.35
3P% - % 3 Points Shots made				
2PA - 2 Points Shots attempted	3.22 (< 0.001)	2.66	5.72 (< 0.001)	3.63
2P% - % 2 Points Shots made	1.32 (0.004)	1.06	1.54 (0.003)	
FTA - Free Throws attempted	1.30 (0.002)	1.24		1.07
FT% - % Free Throws made				
OREB - Offensive Rebounds				
DREB - Defensive Rebounds				
AST - Assists				
TOV - Turnovers				
STL - Steals	0.90 (0.144)	1.00	0.77 (0.018)	0.95
BLK - Blocks				
PF - Personal Fouls				
DD2 - Double Doubles	1.27 (0.001)	1.21	1.27 (0.003)	1.21
TD3 - Triple Doubles				
$+/-$ - Plus/Minus				
Age				
% Won Matches (by the player)				
MIN - Minutes played	1.24 (0.004)	1.20		1.08
% Won Matches (by the team)				
Points Gained (by the team)				
All-Star Game (Yes/No)		1.05	2.27 (0.004)	1.86
NBA G-League (Yes/No)				

Figure 18 shows two examples of the estimated survival curves of the sample stratifying for 2 of the more important variables for the setting with a 50% of censoring. The number of 2PA (normalized and standardized) has been dichotomized in two categories with respect to the median. From Figure 18a it can be seen that players selected for the All-Star game reach the fixed amount of points very earlier than those who have not been selected for the match. Similarly, Figure 18b shows that players who attempt a higher number of two-points shots have a higher probability of gaining the given threshold earlier than those who attempt a lower number of shots.



(a) All-Star game



(b) Two Points Shots

Figure 18: Survival Curves of the sample stratified for: a) All-Star game and b) Number of attempted two-points shots. The 2PA variable (normalized and standardized) has been dichotomized with respect to the corresponding median. The dashed line refers to the selection of the All-Star game and to the high category of 2PA, the solid line refers to having not been selected for the All-Star game and to the low category of 2PA

6.2.6 Evaluation of models' performance

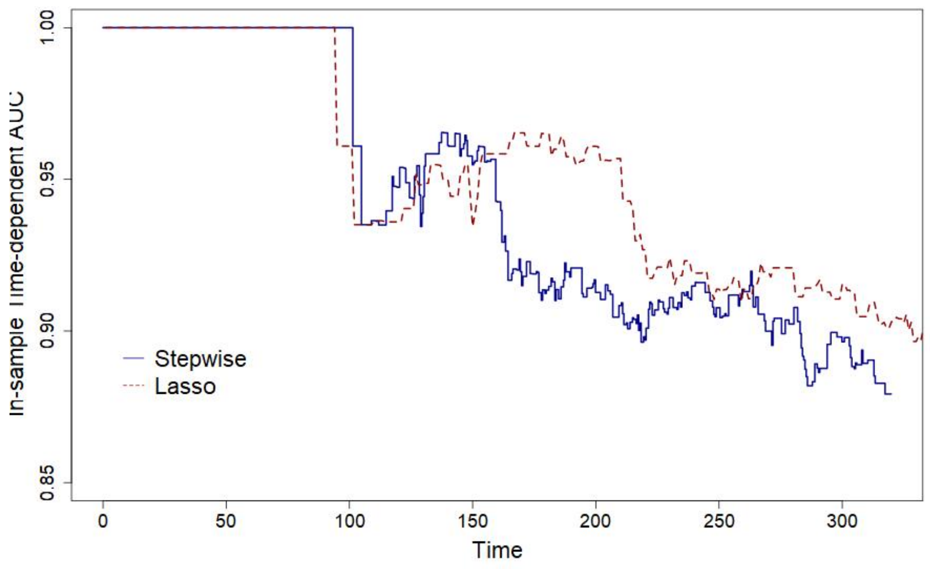
For evaluating the performance of LASSO Cox model and for checking the satisfaction of the assumption of proportionality of hazards, a Cox model with the variables selected by LASSO was fitted.

The two models satisfied the assumption of proportionality of hazards (null hypothesis) in both settings, as measured by the statistical test based on Shoenfield residuals ($p = 0.072$ and 0.077 respectively for Stepwise and LASSO Cox models when the percentage of censoring was equal to 20% and $p = 0.093$ and 0.077 when the percentage of censoring was equal to 50%).

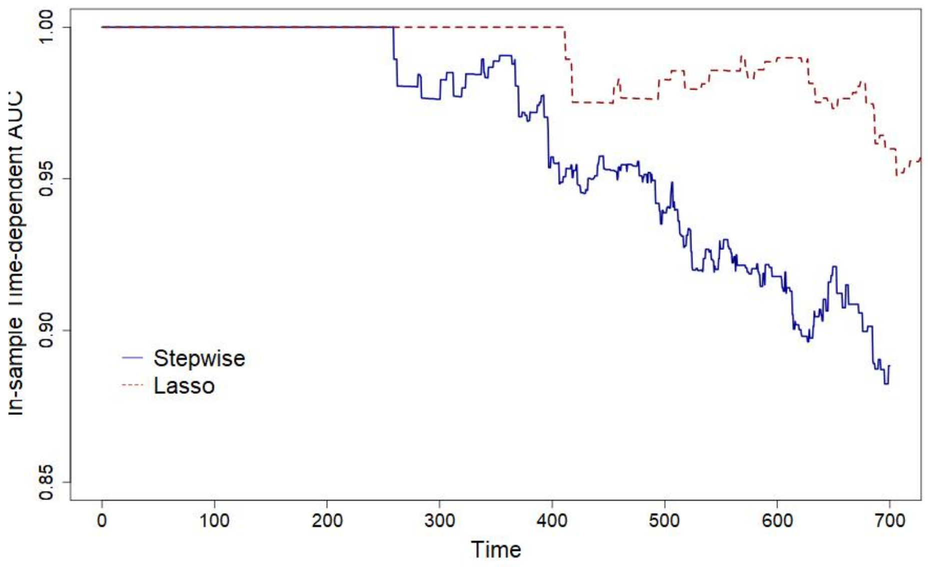
Then, model's performance has been evaluated through the evaluation of time-dependent AUCs. The performance assessment has been made in-sample, so the performance could have been overestimated. It can be seen that in both the two settings the performance of the two models is good (greater than 0.85). Moreover, it can be observed that LASSO seems to slightly overperform Stepwise Cox in both the two settings (Figure 19). Finally, the Harrell concordance index has been evaluated through 10-fold cross-validation for both the two models. When the percentage of censoring was equal to 20% the concordance was equal to 0.82 for both the two models, while for 50% the C-index was equal to 0.88 and 0.86 respectively for Stepwise and Lasso Cox. The results suggest a high concordance between the observed and predicted outcome and, consequently, a good performance of both models in the two settings.

6.2.7 Discussion

Survival analysis has been already used for sport analytics for many aims; however, up to now, to the best of our knowledge, it has never been used for evaluating players' performance. This study shows the use of a classical method of survival analysis, as Cox regression, and of a more recent extension, regularized Cox regression through LASSO, for identifying the achievements that are highly associated to the offensive performance of NBA players measured in terms of exceeding of a given threshold of points. Two settings were analyzed, one with a lower threshold, equal to 99 points, and one with a higher one (255 points) for examining whether there is a different impact of the considered variables on the outcome. Summarizing, in the two settings almost all the same variables were selected by both the two models. In particular, from both Cox regression and LASSO, it emerged that (as expected) players attempting more shots, free throws, 2 and 3 points shots have a higher probability of exceeding the fixed amount of points. In particular, it emerged that 2PA and 3PA are the most relevant ones and that their impact increases when the threshold is higher. Moreover, both the two models suggest that gaining more achievements (as measured by the number of DD2) is associated to an increase of the probability of success. In addition, the number of STL was identified as negatively associated to the outcome. Thus, it



(a) 20% censoring



(b) 50% censoring

Figure 19: Time-dependent AUC for Stepwise Cox and LASSO in the two settings. a) 20% of censoring. b) 50% of censoring

seems that this variable decreases the probability of reaching the two thresholds (HR < 1). Its importance is relevant when the threshold is high, while it is not statistically significant ($p > 0.05$) when the fixed threshold of points is low. This is an attractive result because it is the only variable related to defense included in the models. This may suggest that who is more involved in defense is then penalized in terms of points gained. On the other hand, variables like rebounds and blocks have not been selected by the models and, from this point of view, defense seems to be not influential on the scored points. These remarks about the role of defense should be deepened with some research specifically devoted to answer this question.

Finally, an interesting result emerged from the comparison of the results of the two settings; indeed, All-Star game becomes an important factor when considering a higher threshold. Therefore, when the fixed amount of points is higher, being a very good player (and so having been selected for playing at the All-Star game) almost doubles the probability of reaching that threshold (HR=2.3 and 1.9 for Stepwise Cox and LASSO respectively).

The study has some limitations associated to possible issues related to the assumption of *random censoring*. Random censoring occurs when the subjects who are censored at time t are representative of all the study subjects who remained at risk at time t with respect to their survival experience [Kleinbaum et al., 2012]. This hypothesis may be not respected due to the fact that it is likely that censored players (i.e. players who didn't exceed the threshold) have not the same abilities of players who manage to exceed that amount of points. On the other side, the assumption of *independent censoring* can be retained valid. Indeed, it is reasonable to assume that within any subgroup of interest, the subjects who are censored at time t are representative of all the subjects in that subgroup who remained at risk at time t with respect to their survival experience. So, random censoring could be assumed conditional on each level of covariates [Kleinbaum et al., 2012]. For this reason, once having taken into account the abilities of each player, as measured through the covariates introduced in the model, the probability of being censored can be considered independent of the probability that the event of interest occurs. Future research will deepen this issue.

Moreover, due to the relatively low number of subjects, all the sample has been used for fitting the model and performance has been evaluated in-sample. Future work will consider the use of data relative to 2021-2022 season as test set.

Finally, some improvements include considering the possible presence of interactions among covariates, nonlinear effects of the predictors and threshold effects. Nonparametric and machine learning methods will be used to examine this point.

7 Machine Learning Algorithms of Survival Analysis: Applications to Basketball

This chapter shows the application of survival trees and random survival forests to basketball data. The paper below is a working paper that will be submitted shortly. It will also be presented at the CMStatistics 2022 in London in the invited organized session *Sport analytics*.

7.1 Survival analysis in basketball: an analysis of the NBA players' offensive performance

Working Paper to be submitted (draft)

7.1.1 Introduction

Sport analytics has become widespread in recent years for answering different questions concerning several fields. Among these, there is of course the performance analysis of players/teams and the prediction of tournament/matches outcomes. Among the used methods in this field there is also survival analysis, which was used for evaluating the occurrence of injuries, return to sport after a given injury [Sochacki et al., 2019, Jack et al., 2019, Mai et al., 2017] or for evaluating the drop-out of athletes and coaches [Pion et al., 2015, Moulds et al., 2020, Fynn and Sonnenschein, 2012, Wangrow et al., 2018].

The aim of our work is to investigate which are the main achievements (e.g. attempted shots, double doubles and rebounds) associated to the probability of exceeding a given amount of points during the post All-Stars season segment. Until now, to the best of our knowledge, no studies have used this kind of analysis for studying athletes' performance. The only exception regards a previous study [Macis et al., 2022b,a] in which we used Cox regression model and LASSO for answering the above question. It emerged that having attempted more shots (mainly two- and three- points shots) in the previous season segment, having been selected for the All-Stars game, and having gained more achievements (higher number of double doubles) increase the probability of exceeding the threshold of 255 points. Moreover, it resulted that steals were negatively associated with the survival outcome, suggesting that who is more involved in defense gains less points. However, the other defense variables were not selected by the models. Therefore, the role of defense should be further investigated [Macis et al., 2022a].

The problem under investigation, analyzed in the same framework of the previous papers, was the presence of nonlinear relationships and interaction effects between the main NBA achievements and the probability of exceeding the cut-off of 255 points in a shorter time. This analysis was performed using two machine learning methods, survival trees and random survival forests. This choice has been made because these are nonparametric methods, that allow high flexibility taking into account nonlinear effects and possible interactions between the predictors without specifying them beforehand.

The reminder is organized as follows. The following section describes methodological details of the used algorithms; then, Section 7.1.3 presents the data used and the study design. Then, the results are shown. The paper ends with the final discussion.

7.1.2 Methods

Survival analysis aims to study the occurrence of an event of interest during a given period of time (follow-up). It is characterized by the presence of *censoring*. Censoring occurs when the event has not been observed for a subject during the observation time, so that the only known thing is the last time he/she did not experience the event [Collett, 2015]. In this context, the i^{th} player is defined by three elements:

- the observed time τ_i, that can be the time t_i at which the event occurred or the censoring time c_i;
- the event indicator δ_i that assumes value 1 if the subject experienced the event and 0 otherwise;
- the vector of observed covariates $\boldsymbol{x}_i$.

Machine learning methods for survival analysis have been used. Methodological details about survival trees and random survival forests are reported in Subsections 7.1.2.1 and 7.1.2.2 respectively.

7.1.2.1 Survival Trees

Survival trees are an extension of decision trees to censored data with the aim of identifying, by a recursive partitioning of the covariate space, subgroups of subjects homogeneous in terms of survival probability. Several proposals have been advanced over the years, but still there is not an algorithm recognized as the best [Macis, 2022a]. In this work we fitted a Relative Risk Tree (RRT) [LeBlanc and Crowley, 1992], a Conditional Inference Tree (CIT) [Hothorn et al., 2006b] and a Model-Based recursive partitioning algorithm (MOB) [Zeileis et al., 2008].

The first algorithm, RRT, exploits the connection between the likelihoods of the proportional hazards and Poisson models. To this extent RRTs are equivalent to Poisson trees. The final output is a tree with a relative risk estimated at each terminal node.

CITs are instead based on a different probabilistic framework, known as conditional inference; they are based on nonparametric tests that study the association between the outcome and the variables. This algorithm is characterized by the separation of variable and split-point selection, ensuring unbiasedness and reducing the risk of overfitting [Hothorn et al., 2006b].

Finally, MOBs are based on the hypothesis that it is unreasonable assuming that a single global model fits all observations well. To this extent, MOBs fit a segmented parametric model by computing a tree in which every leaf is associated with a fitted model.

One of the main advantage of these algorithms is their flexibility and the interpretability of the obtained results.

7.1.2.2 Random Survival Forests

Random survival forests (RSF) are an extension of random forests [Breiman, 2001a] to censored data. One of the most used algorithms is the one proposed by Ishwaran et al. [2008]. This method follows the prescriptions laid out by Breiman [2003] and it basically works as classical random forests. It starts with B bootstrapped samples from the original one; then, at each node a random subset of covariates is considered for splitting and B full-grown trees are obtained. In particular, trees can be built using different splitting criteria (log-rank, conservation of events, standardized log-rank and random log-rank splitting rules). Finally, two ensemble estimates, the bootstrap and the out-of-bag estimates, are evaluated averaging all the obtained estimates considering respectively the entire set of trees or only the trees in which the corresponding observation was not used for growing them. As outcome, the cumulative hazard function (CHF) is usually used, together with the so-called "*mortality*", a measure that can be interpreted in terms of the expected total number of events [Ishwaran et al., 2008]. One of the main advantages of using forests, instead of trees, is that they are more stable with respect to noise data. Moreover, they allow to analyze both linear and nonlinear relationships and to detect interactions between covariates without the need of specifying them beforehand, as needed in a Cox or in a Cox penalized model.

7.1.3 Data and Study Design

The 2020-2021 NBA regular season was examined. In details, the season was split in two segments with respect to the All-Stars (AS) game (pre- and post- AS). The first part of the season was used for extracting the baseline covariates just before the AS game. The second part was instead used as follow-up period. Baseline covariates were obtained from the NBA website; moreover, information about the selection for the AS game was included. Play-by-play data, kindly made available by BigDataBall (`www.bigdataball.com`, a reliable source of validated and verified data for NBA) have been used for extracting the survival outcome. In order to make the number of achievements comparable among players, all the variables indicating the total number of achievements of each player have been normalized dividing them by his minutes played. Finally, all the explanatory variables have been standardized (mean 0 and standard deviation 1).

In details, for each player M_j, the minutes played and the corresponding points P_j gained until different time points t_j $(j = 1, 2, \ldots, J)$ were recorded. The time variable was treated as player-time: it increases when the player is in the court and remains constant when he is not playing. Then, a given threshold P of scored points was fixed and the event was defined as the exceeding of that threshold. Let's set $j_i^*(P) = \text{argmin}_{j=1,\ldots,J} p_{ij} > P$; the time at which the player exceeds the threshold is therefore $t_{j_i^*(P)}$, with p_{ij} being the amount of points gained by the i^{th} player until time t_j. Thus, the outcome of the study is composed of (i) the time-to-event $\tau_i = \min\{m_{ij_i^*(P)}, m_{ij}\}$, where $m_{ij_i^*(P)}$ is the amount of minutes played by the player before reaching the threshold and m_{ij} corresponds to the amount of minutes played at the end of the season segment, and of (ii) the event indicator $\delta_i = I[p_{ij} > P]$.

7.1.4 Results

The overall sample consisted of 359 NBA players, after having excluded those who played less than 48 minutes during the post AS game season segment and those who changed team during the season. The threshold P was set equal to 255, ensuring a 50% of censoring. Figure 20 shows the resulting pruned RRT. Each terminal node reports the following information: (i) the estimated measure of relative risk; (ii) the ratio between the number of events and observations; (iii) the percentage of subjects in the node. It can be seen that the first splitting variable is the number of two-points shots attempted (2PA), followed by the number of three-points shots attempted (3PA) and the percentage of realized three-points shots (3P%). The other selected variables were the number of free throws attempted (FTA) and the percentage of realized free throws (FT%) and the selection for the AS game. It resulted that players who attempted more two-points shots and had a higher 3P% in the first part of the season and, additionally, have been selected for playing at the AS game have the highest probability of exceeding the fixed amount of points during the follow-up (as indicated by the measure of relative risk of the right terminal node, equal to 8.1). Moreover, as the number of attempted and realized shots decrease, the probability of gaining more than 255 points decreases substantially.

Similarly, Figure 21 shows the final output obtained through the *ctree* algorithm. Each terminal node shows the corresponding survival curve measuring the probability of exceeding the threshold later than the examined timepoint. The obtained tree has a different structure of that of the RRT; however it can be seen that some of the identified variables are the same (2PA, 3PA and 3P%). In particular, it can be seen that even in this case the first split is due to the number of 2PA. Then, differently, the free throws do not appear in the model. Finally, the amount of minutes played (MIN) and double doubles (DD2) emerged as variables significantly related to the outcome.

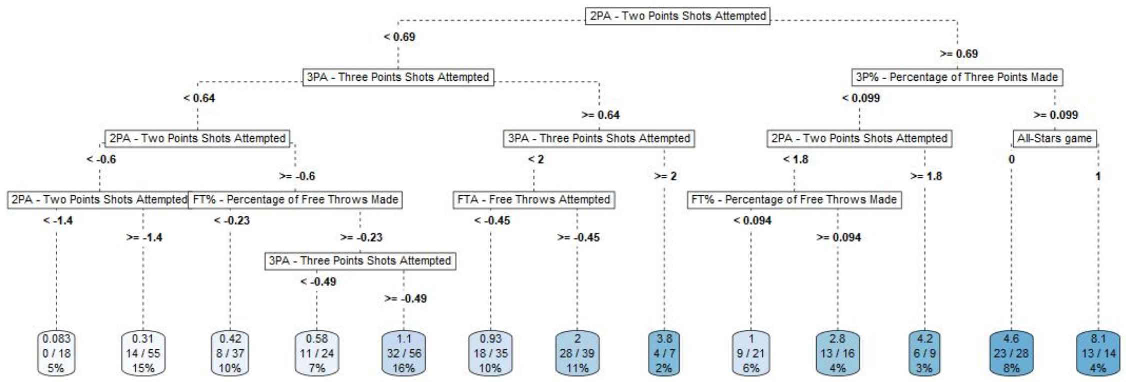

2PA - Two Points Shots Attempted
< 0.69
>= 0.69
3PA - Three Points Shots Attempted
3P% - Percentage of Three Points Made
< 0.64
>= 0.64
< 0.099
>= 0.099
2PA - Two Points Shots Attempted
3PA - Three Points Shots Attempted
2PA - Two Points Shots Attempted
All-Stars game
< -0.6
>= -0.6
< 2
>= 2
< 1.8
>= 1.8
0
1
2PA - Two Points Shots Attempted
FT% - Percentage of Free Throws Made
FTA - Free Throws Attempted
FT% - Percentage of Free Throws Made
< -1.4
>= -1.4
< -0.23
>= -0.23
< -0.45
>= -0.45
< 0.094
>= 0.094
3PA - Three Points Shots Attempted
< -0.49
>= -0.49
0.083 0 / 18 5%
0.31 14 / 55 15%
0.42 8 / 37 10%
0.58 11 / 24 7%
1.1 32 / 56 16%
0.93 18 / 35 10%
2 28 / 39 11%
3.8 4 / 7 2%
1 9 / 21 6%
2.8 13 / 16 4%
4.2 6 / 9 3%
4.6 23 / 28 8%
8.1 13 / 14 4%

Figure 20: Estimated Relative Risk Tree. Each leaf shows (1) the corresponding measure of relative risk, (2) the ratio between the number of events and observations in the node and (3) the percentage of subjects in the node

The players selected for the AS game who have attempted more two-points shots and have played more minutes in the first part of the season have a survival probability that decreases quickly (as shown by the Kaplan-Meier survival curve plotted in the corresponding leaf), and therefore have a higher probability of experiencing the event of interest.

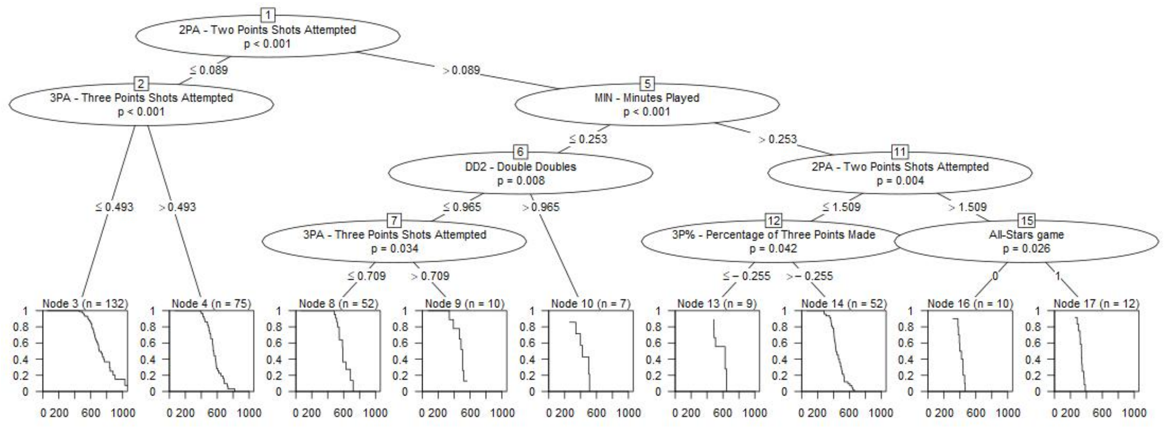
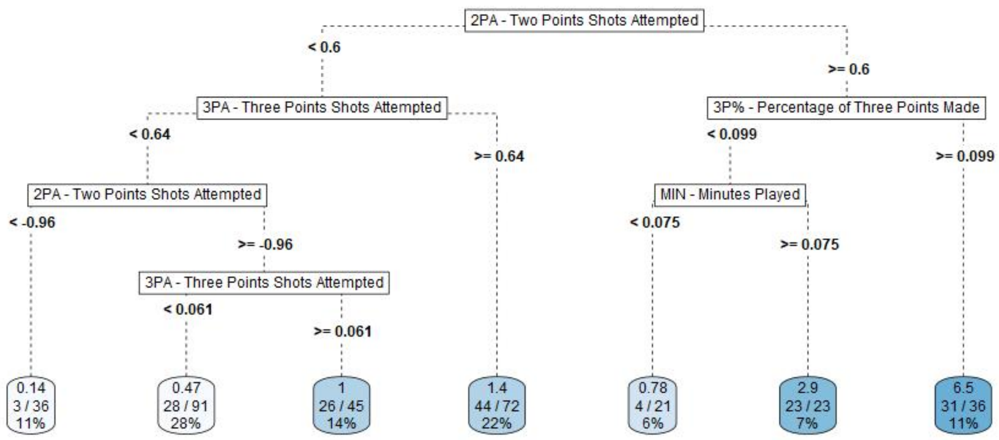


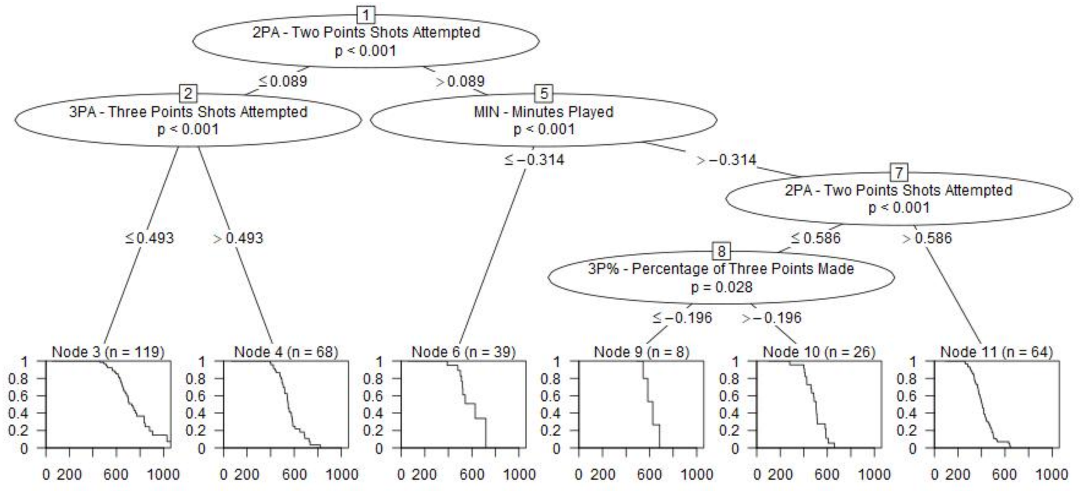
Figure 21: Estimated Conditional Inference Tree. Each leaf represents the survival probability of the corresponding node, measuring the probability of not exceeding the threshold

However, it is known that the structure of trees is not stable with respect to noise. This can be seen in Figure 22 that shows the RRT and CIT obtained on the basis of a slightly perturbed dataset (using only 90% of the entire sample). The resulting trees have a different structure and some variables that had been previously selected, as the All-Stars game, disappeared.

Therefore, following, random survival forests were used, due to the instability of survival trees. In details, the RSF was grown with 1000 trees and randomly selecting 8 variables at each split. One of the main interesting features of random forests is the variable importance that measures the impact of each predictor on the outcome, taking



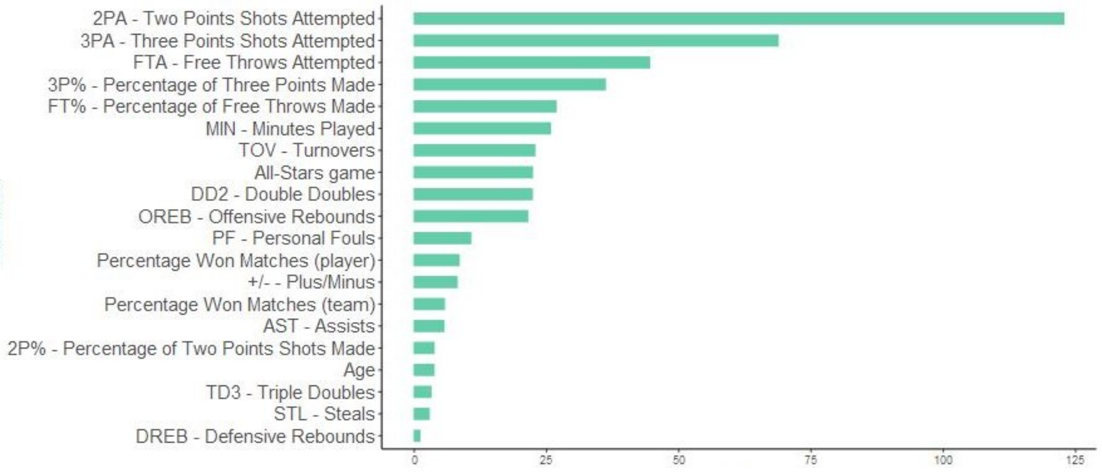
(a) Relative Risk Tree fitted on the perturbed dataset



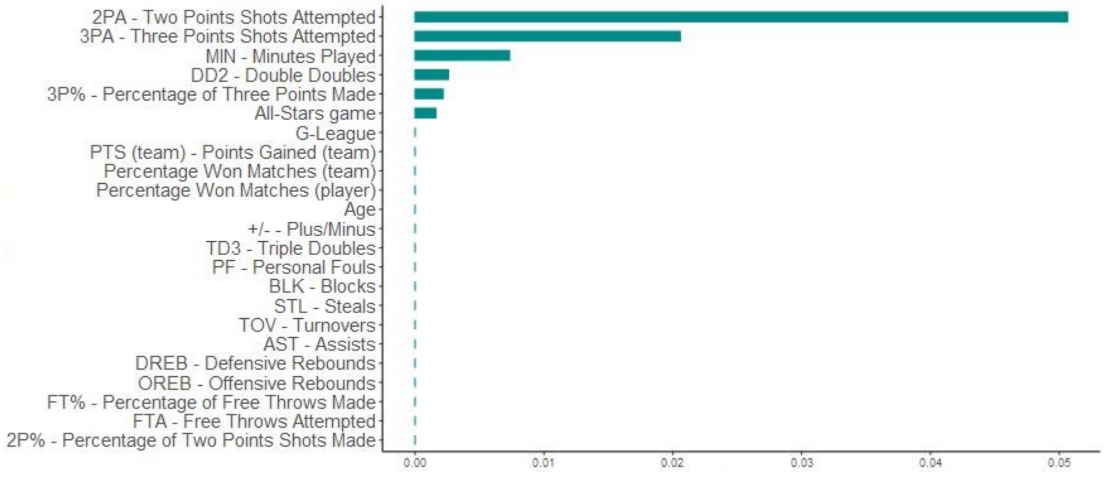
(b) Conditional Inference Tree fitted on the perturbed dataset

Figure 22: Survival Trees fitted on a perturbed dataset in which only 90% of the overall sample was included as training observations

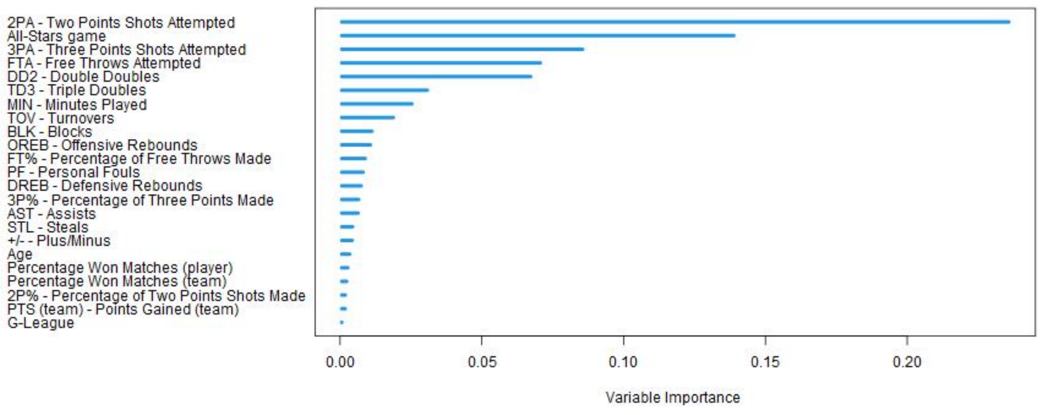
into account the existing interactions and nonlinearities. The values obtained are reported in Figure 23c. Interestingly, it can be seen that the most important variables are the number of attempted two-points shots and the selection for the All-Stars game, followed by the number of 3PA, DD2 and FTA. These results are confirmed only in part by those obtained from the RRT and the CIT grown in the entire sample (Figure 23a and 23b respectively). Indeed, it can be seen, for example, that All-Stars has a lower importance in both the two cases. Moreover, while the RSF suggests that the percentage of realized shots is not so important, the two survival trees give them higher importance. It is important to notice that CIT evaluates the variable importance only for the covariates included in the final model.



(a) Relative Risk Tree



(b) Conditional Inference Tree



(c) Random Survival Forest

Figure 23: Variable Importance obtained from the three machine learning algorithms

Finally, we decided to plot the surrogate tree of the forest by using the obtained OOB predictions, i.e. the so-called OOB "*mortality*", that measures the expected number of events under a null hypothesis of similar survival behaviour [Ishwaran et al., 2008]. Specifically, the extracted predictions were used as outcome variable of a regression tree. The final resulting tree (see Figure 24) does not need pruning (it is already the best pruned tree). It can be observed that the only variables appearing in the tree are the number of attempted shots (in order of importance 2PA, 3PA and FTA) and the selection for the All-Stars game. Each terminal node shows a mean estimate of the "*mortality*" in that node: it follows, as already seen, that players who have attempted many two-points shots and have been selected for the All-Stars game have the highest expected cumulative risk of exceeding the threshold. On the contrary, players who have attempted the lowest number of two- and three- points shots have the lowest expected number of events.

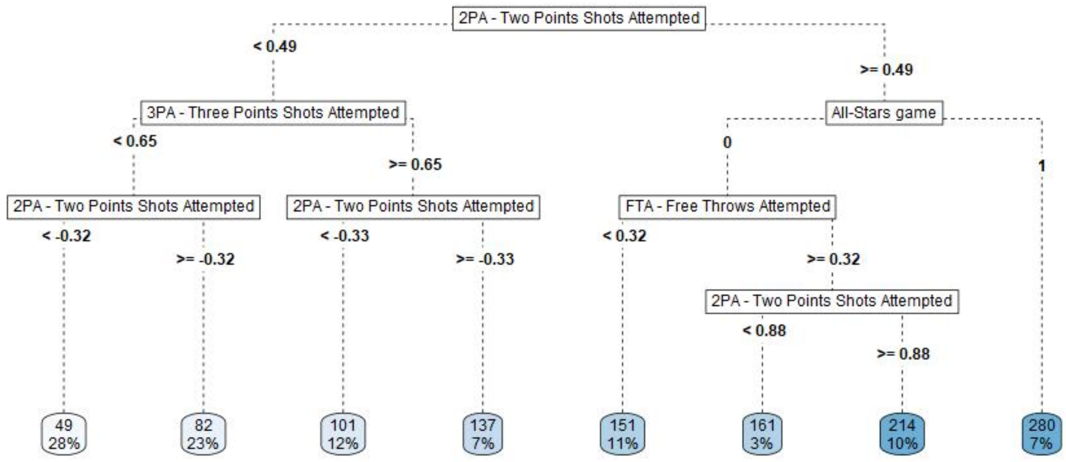

2PA - Two Points Shots Attempted
< 0.49
>= 0.49
3PA - Three Points Shots Attempted
All-Stars game
< 0.65
>= 0.65
0
1
2PA - Two Points Shots Attempted
2PA - Two Points Shots Attempted
FTA - Free Throws Attempted
< -0.32
>= -0.32
< -0.33
>= -0.33
< 0.32
>= 0.32
2PA - Two Points Shots Attempted
< 0.88
>= 0.88
49 28%
82 23%
101 12%
137 7%
151 11%
161 3%
214 10%
280 7%

Figure 24: Surrogate Survival Tree

An alternative approach based on the MOB algorithm has also been used to verify whether different models could be identified in different subgroups of the sample. The MOB was fitted using a Cox model in which the variables included in the surrogate tree were used as regressors and all the covariates were used as possible partitioning variables. The resulting output is shown in Figure 25. In the terminal nodes the β coefficients of the different regressors have been reported. It can be seen that the only splitting variable is the number of DD2 which, beyond 2PA, All-Stars game, 3PA and FTA is the other achievement identified by the RSF with a high variable importance (see Figure 23c). In the subgroups identified in the model the regressors have a different effect on the survival outcome. In particular, it can be seen that the All-Stars game and the number of 3PA have a higher effect in the subgroup of players performing a lower number of double doubles. Therefore, if players who have scored a lower number of DD2 have been selected for the AS game or have attempted many three-point shots in the first part of the season, their probability of scoring more than 255 points in a shorter time is higher. On the contrary, the number of FTA has a higher effect on the survival outcome in the group with a higher number of DD2. Finally, the number of

2PA has the same effect in the two groups.

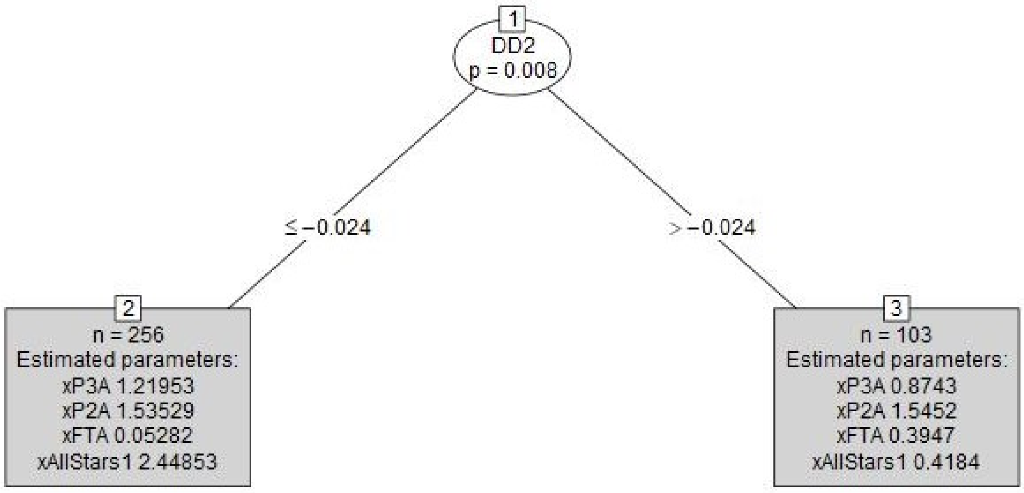


Figure 25: Model-based recursive partitioning algorithm. Each leaf reports the estimated β coefficients obtained fitting a Cox regression model

Furthermore, to examine the effect of double doubles on the probability of scoring more than 255 points, the Kaplan-Meier curves in the two subgroups identified by the MOB have been estimated and are shown in Figure 26. It can be seen that those players who have gained a higher number of DD2 in the pre All-Stars game season segment have a higher probability of exceeding the threshold earlier than the others.

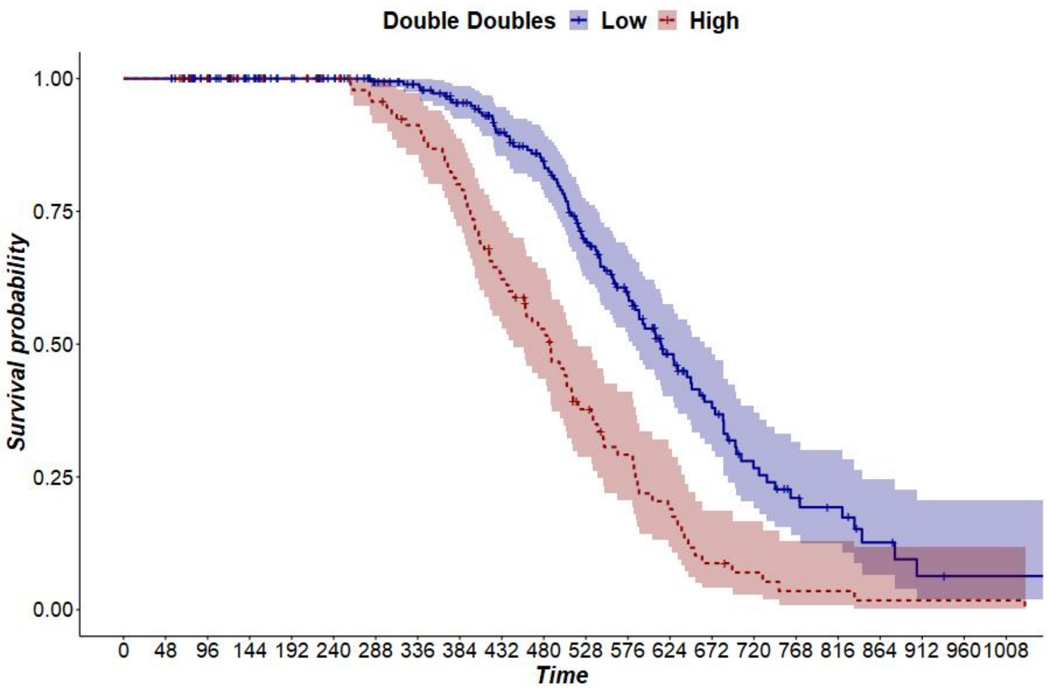


Figure 26: Estimated Kaplan-Meier survival curves stratified for double doubles. The two groups have been identified by the Model-based recursive partitioning algorithm

7.1.5 Performance Evaluation

As it can be seen in Figure 27 the out-of-bag error rate of the random survival forest (measured as the complementary to one of the Harrell's Concordance index [Harrell et al., 1982]) becomes stable after around 300 trees.

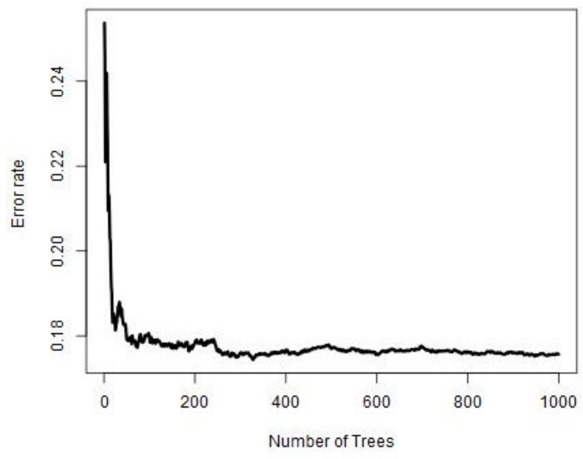


Figure 27: Error rate of the random survival forest for number of trees

Then, some performance measures have been evaluated for each of the three machine learning methods shown here (RRT, CIT, RSF and MOB) and have been compared with those of Stepwise Cox and LASSO Cox models presented in [Macis et al., 2022a] (Section 6.2). These measures have been evaluated in-sample and so they represent more precisely a measure of fitting of the different algorithms. Figure 28 represents the estimated time-dependent AUCs. For the performance assessment of MOB the weighted mean of the two time-dependent AUCs estimated for the two Cox models fitted in the identified subgroups has been evaluated, using as weights the corresponding sample sizes. The results show that the random survival forest is the algorithm with the best fitting. Then, RRT, MOB, Cox and LASSO Cox seem to have a comparable (high) goodness of fit; finally, the algorithm with worse fitting seems to be the CIT. The similar performance of RRT, MOB, Stepwise Cox and LASSO Cox may be also due to the same assumption on which the models are based, i.e. the proportionality of hazards.

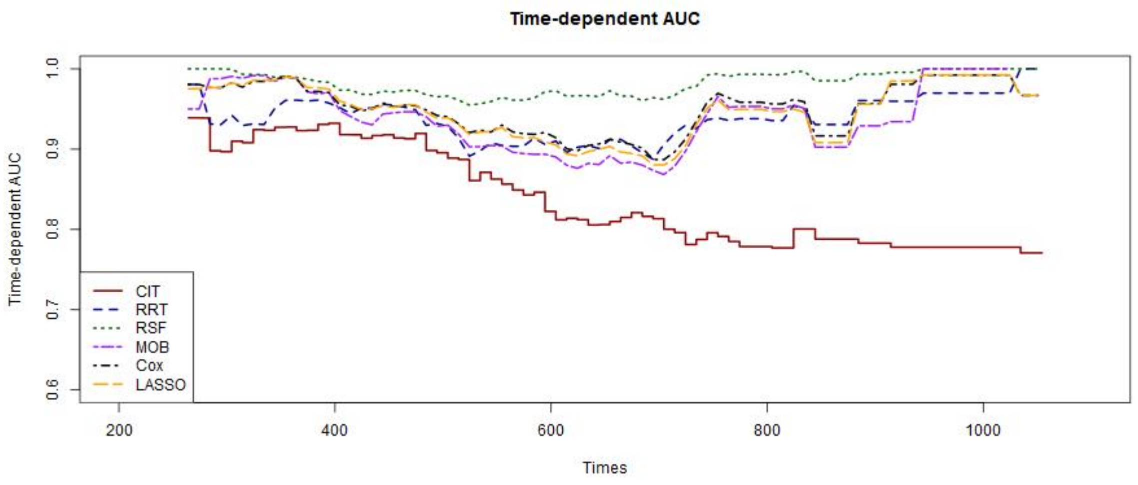


Figure 28: In-sample Time-dependent AUCs evaluated for each fitted model: (i)CIT: Conditional Inference Tree; (ii) RRT: Relative Risk Tree; (iii) RSF: Random Survival Forest; (iv) MOB: Model-Based Recursive Partitioning; (v) Cox: Stepwise Cox; (vi) LASSO: LASSO Cox

These results are also confirmed - in outline - by the overall error rate (in-sample) measured as the complementary to one of the Harrell Concordance index. Some noteworthy differences can be observed among the adopted techniques (see Table 15).

Table 15: In-sample error rates for each fitted model

Model	**Error Rate**
Relative Risk Tree (RRT)	0.145
Conditional Inference Tree (CIT)	0.180
Model-based recursive partitioning algorithm (MOB)	0.188
Random Survival Forest (RSF)	0.114
Cox regression	0.130
LASSO Cox	0.133

7.1.6 Discussion

Up to now, to the best of our knowledge, survival analysis has not been widely used for assessing players' performance. This work aims to investigate the achievements more associated with the probability of exceeding a fixed amount of points P (in this example $P = 255$). Another study already tried to examine this problem using statistical models, in particular Cox regression and LASSO Cox [Macis et al., 2022a,b]. This study showed that the most important variables were the number of two- and three-points shots, the selection for the All-Stars game and to some extent also the number of free-throws, the number of double doubles and the percentage of 2-points shots. All these achievements increased the probability of exceeding the fixed threshold in a shorter time. Moreover, it seemed that also steals played a role in the exceeding of the threshold: a higher number of steals was associated with a lower probability of exceeding P. However, in Macis et al. [2022a] interactions and possible nonlinear effects were not considered.

This study, therefore, aimed to take into account these aspects through the use of survival trees and random survival forests. The obtained results confirmed both the instability of trees and the power of random survival forests in overcoming this issue. Moreover, it emerged that the RSF had the better performance, in comparison with all the other competitors, i.e. RRT, CIT, MOB, stepwise Cox and LASSO Cox. However, it is important to keep in mind that performance has been evaluated in-sample for all the models (for having comparable results with the statistical models reported in Macis et al. [2022a] and in Section 6.2); thus, the performance may have been overestimated and these results should be more appropriately interpreted as a measure of goodness of fit.

Interestingly the obtained results are coherent with the previous work with the only exception concerning the role of steals that does not seem to have a high importance (see Figure 23c). In details, the most important variables, as also shown by the fitted surrogate tree, are the number of attempted shots (2PA, 3PA and FTA) and the selection for the All-Stars game. On the contrary, the percentage of realized shots (in the first part of the season) does not seem to have a so high impact on the outcome, as suggested by the obtained variable importance values (Figure 23c). Moreover, also the number of double doubles has a significant effect on the probability of exceeding the fixed threshold, as suggested by the MOB. In addition, results showed that this achievement allows to identify two subgroups of players in which the most important variables (2PA, AS, 3PA and FTA) have a different effect on the outcome. Interestingly, even if players have scored a lower number of double doubles, if they have been selected for the AS game or if they have scored a higher number of 3PA, their probability of scoring more than 255 points earlier increases significantly.

Concluding, the study has some limitations associated to possible issues related to the assumption of random censoring [Kleinbaum et al., 2012] that may be not respected [Macis et al., 2022a]: it is likely that very good players, who play many minutes, have a higher probability than less good players of exceeding the fixed amount of points; however, the assumption of independent censoring (i.e. random censoring conditional on the set of covariates) can be retained valid (Kleinbaum et al. [2012], Macis et al. [2022a]). Future research will try to definitely address this problem by using transformation models proposed by Hothorn and Zeileis [Hothorn and Zeileis, 2021].

Conclusions

The focus of this thesis is survival analysis, a collection of methods used in longitudinal studies for investigating the occurrence of a given event of interest during a follow-up period. Although at the beginning only statistical models had been introduced, over the years also machine learning methods have been proposed. Among these, the thesis dealt with survival trees and random survival forests.

The first step of the research involved literature review in order to build the necessary background. During this study, it resulted clear that there are several proposals and no harmonization among them. Moreover, together with the possible theoretical issue of not knowing which are the main advantages of each algorithm and thus which is the best one in the different situations, practical issues emerged concerning the application of these methods with *R*. Several drawbacks emerged, mainly concerning the performance assessment: many functions presented little documentation making it difficult the approach to these methods, and this lack of clarity has been also confirmed by the absence of tutorials (both in literature and on the web) about their use.

For this reason, one of the main aims of this work was to make order among all the proposed algorithms and *R* functions and to provide a solution of many of the faced computational issues. Therefore, I wrote two contributions on the topic. The first one [Macis, 2022b] is a conference paper that sheds light on the lack of harmonization of the existing methods, presenting the main faced problems; the second one [Macis, 2022a], instead, aims to provide a guide on how (i) simulating censored data; (ii) fitting survival trees with different algorithms and (iii) evaluating trees performance through many indexes. In this paper, the existing issues have been shown together with the found solutions. Therefore, this work, making order among the many existing proposals and the many *R* functions and packages, enables interested users to approach these methods in an easier way.

In this field future research will also involve the creation of a new *R* package (on working) that will include functions that could fix the existing issues. We are writing the new functions; updates will be provided in GitHub and in the web page `https://bodai.unibs.it/ml4sd/`.

A second contribution aimed, instead, to apply survival analysis in a non-usual context. Indeed, even if survival methods are usually used in sport analytics for examining injuries and drop-out rates, they have not been already used, to the best of my knowledge, for analyzing players' performance. More in details, we used both statistical models and machine learning algorithms for survival analysis for analyzing the offensive performance of NBA players, measured in terms of scored points. The final purpose was to identify which were the main achievements associated to the probability of exceeding a given threshold during the post All-Stars game of 2020-2021 season. This work has some limitations related to the assumption of *random censoring* [Kleinbaum et al., 2012], which requires independence between survival and censoring time. However, *independent censoring*, that is random censoring conditional on the set of covariates, can be retained valid [Macis et al., 2022a].

To this extent future research will address this problem in several ways, including the use of transformation models proposed by Hothorn and Zeileis [Hothorn et al., 2018].

Concluding, this thesis aimed to lay the ground for the development of a unified and harmonized framework able to make order among the existing fragmented approaches and without all the computational issues of the existing *R* packages. Moreover, this work showed that the applicability of these methods can be extended to several fields.

The findings of this work highlighted that in survival analysis there are still many unexplored directions that can be considered and investigated and that novel methodological proposals could be provided.

APPENDIX

A.1 Tutorial on How Simulating Right-Censored Data, Fitting Survival Trees, Evaluating Survival Trees Performance with R

In this document an example for simulating right-censored data, growing survival trees and evaluating their performance is reported, together with the related results.

1. Random generation of the set of covariates

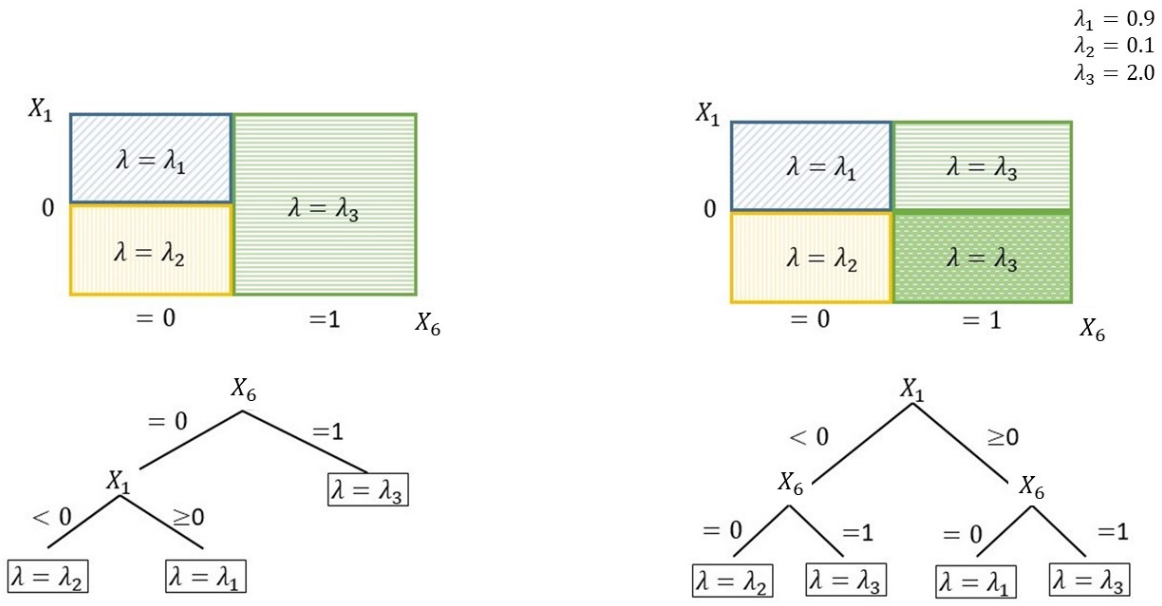
The code below randomly generates 10 covariates: the first half followed a Standard Normal distribution (X_1, X_2, X_3, X_4 and X_5) and the second half (X_6, X_7, X_8, X_9 and X_{10}) a Bernoulli distribution with different probability parameters p ranging respectively from 0.3 to 0.7 with steps of 0.1. For example, in a medical context the covariates could be clinical and socio-demographic information about the patient.

```r
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)
```

2. Definition of the theoretical recursive partitioning model

In this example a Data Generating process with a tree structure was used. For building a recursive partitioning model, different distributions for survival times in the terminal nodes must be assumed (one for each leave of the tree). The tree was characterized by three terminal nodes with the following partitions and distribution of the survival times:

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



So, I hypothesized that the first node had the highest hazard (equal to $\lambda = 2$), the second one had an intermediate hazard, while the third one had the lowest risk of experiencing the event. In a medical context the survival outcome could be the relapse or the death of the patient. However, survival analysis can be carried out also in other settings; examples are the work setting (e.g. dismissal, failure of a firm) or the sport context (e.g. to evaluate the minutes needed to reach a given threshold of points).

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

Finally, we fixed the length of the follow-up equal to 3 years:

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

During a simulation study many datasets of different sample sizes should be examined. Here, for simplicity, only one dataset has been generated.

3. Survival data simulation and definition of the training and test sets

I used the *simsurv* package to simulate survival data. 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 the follow-up; and (v) the **seed** to use in order to get reproducible results.

```r
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, maxt=maxtime,
                              x=as.data.frame(1:sum(node1)), seed=7052022)[,2:3]
# Node2
```

```
simdata[node2,1:2] <- simsurv(dist="exponential", lambda=0.9, maxt=maxtime,
                              x=as.data.frame(1:sum(node2)), seed=7052022)[,2:3]
# Node3
simdata[node3,1:2] <- simsurv(dist="exponential", lambda=0.1, maxt=maxtime,
                              x=as.data.frame(1:sum(node3)), seed=7052022)[,2:3]

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

```
##   eventtime status
## 1 0.2534565      1
## 2 3.0000000      0
## 3 0.1797119      1
## 4 3.0000000      0
## 5 0.5632367      1
## 6 0.3993597      1
```

It can be seen that *simsurv* provides as outcome a dataset with three columns (*id*, *eventtime* and *status*). The *eventtime* column represents the time-to-event/censoring time, while the *status* column indicates the occurrence ($status = 1$) or not ($status = 0$) of the event.

```
str(data)
```

```
## 'data.frame':    1000 obs. of  12 variables:
##  $ eventtime: num  0.253 3 0.18 3 0.563 ...
##  $ status   : int  1 0 1 0 1 1 0 0 1 1 ...
##  $ X1       : num  -0.9326 -0.1384 0.0249 -0.2564 0.1795 ...
##  $ X2       : num  -1.322 -0.859 0.519 -0.833 0.719 ...
##  $ X3       : num  -2.44733 -1.37352 -0.00419 2.04342 -0.83023 ...
##  $ X4       : num  -0.71 1.142 0.369 0.633 -1.432 ...
##  $ X5       : num  -0.773 -0.328 0.214 -0.902 -0.242 ...
##  $ X6       : Factor w/ 2 levels "0","1": 2 1 2 1 1 1 1 1 2 1 ...
##  $ X7       : Factor w/ 2 levels "0","1": 2 2 1 2 1 1 1 2 1 1 ...
##  $ X8       : Factor w/ 2 levels "0","1": 2 2 2 2 1 1 2 1 1 1 ...
##  $ X9       : Factor w/ 2 levels "0","1": 1 2 2 1 2 1 2 1 2 2 ...
##  $ X10      : Factor w/ 2 levels "0","1": 2 2 2 1 2 1 2 1 2 1 ...
```

Once data has been simulated the training and test set can be randomly generated, taking a similar percentage of events in the two sets.

```
set.seed(70522)
library(caret)
trainset <- createDataPartition(y=data$status, p=0.5, list=F)
data_train <- data[trainset,]
data_test  <- data[-trainset,]
dim(data_train)
```

```
## [1] 500  12
```

```
dim(data_test)
```

```
## [1] 500  12
```

```
table(data_train$status)
```

```
##
##   0   1
## 143 357
```

```
table(data_test$status)
```

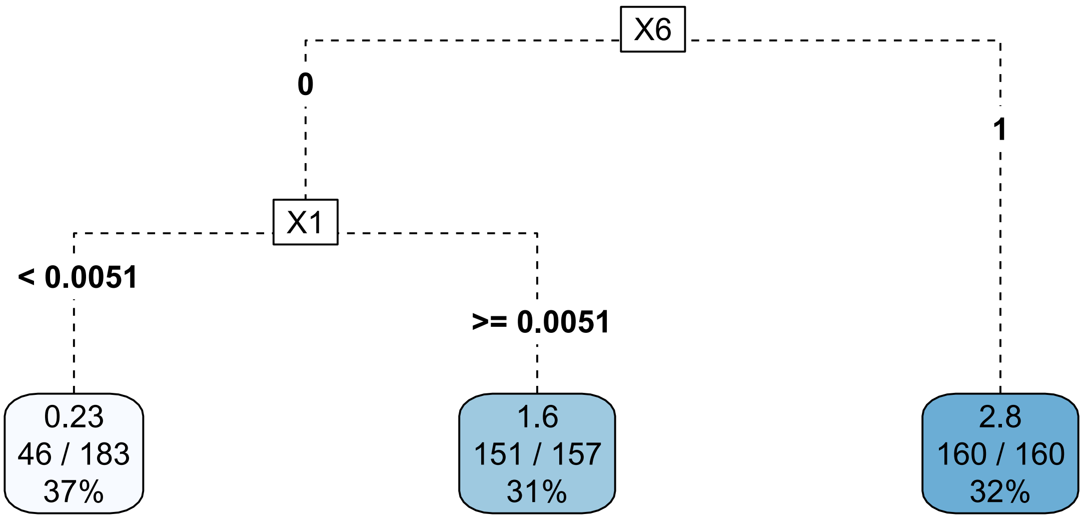
```
##
##   0   1
## 140 360
```

4. Fitting the model on the training set

Relative Risk Trees

The *rpart* function uses Poisson trees for building a Relative Risk Tree (RRT). The required arguments are: (i) a survival formula, i.e. a formula with a *Surv* object (in the *survival* package) 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)
```



It can be seen that after pruning, the resulting tree is a tree with three leaves corresponding with the right tree theoretical partitions. In each leave the following information can be retrieved:

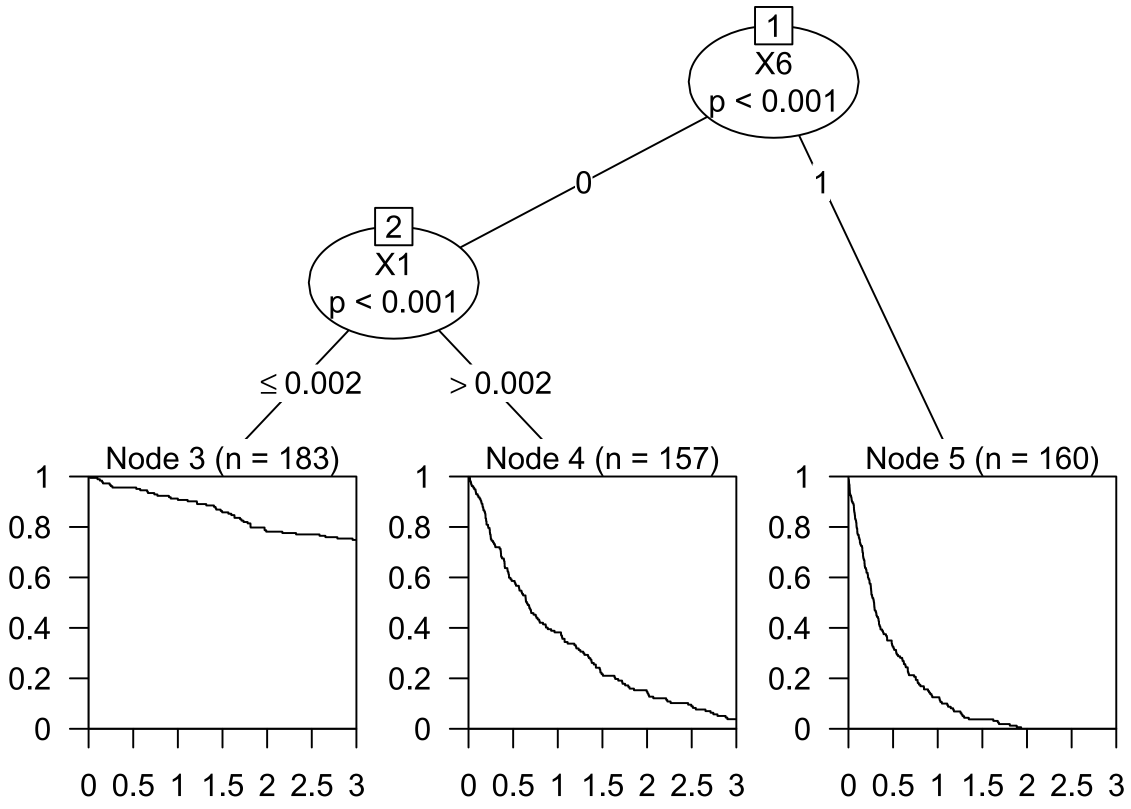
- the relative hazard (e.g. for the left terminal node equal to 0.23). i.e. the hazard of the node relative to that in the root node;
- the ratio between events and the number of observations included in that node (e.g. 46 and 183 respectively in the left terminal node);
- the percentage of observations included in that node (e.g. the left node contains the 37 of the entire training sample).

In particular, it can be seen that the estimated relative hazards in each node are coherent with the theoretical ones. So, for example, the observations which have a value of $X6$ equal to 0 and a value of $X1 < 0.0051$ have the lower hazard with respect to the overall sample.

Conditional Inference Trees with *ctree*

The *ctree* function in *partykit* allows to grow a Conditional Inference Tree through the *ctree* algorithm (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 splitpoint 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.

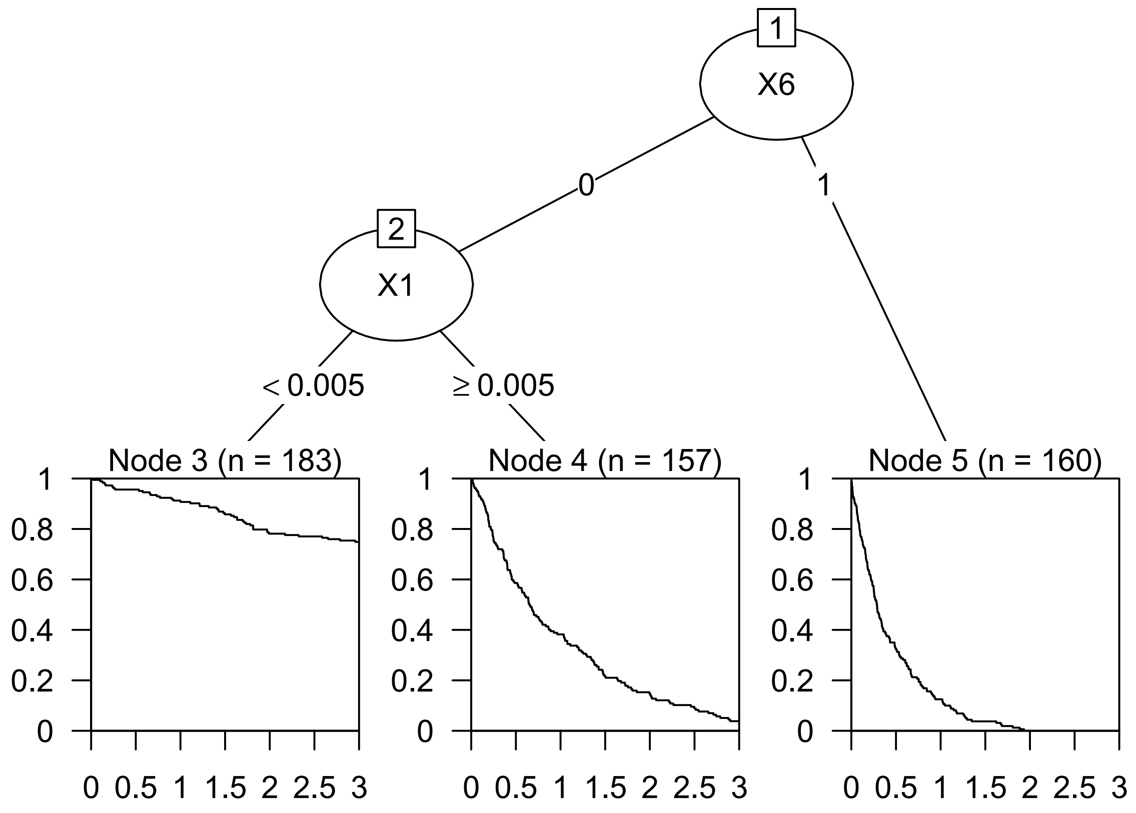
```
library(partykit)
CIT <- ctree(formula=Surv(eventtime, status)~., data=data_train,
                control=ctree_control(minsplit=20))
# Plotting CIT
plot(CIT)
```



The plot of a CIT provides for each terminal node the number of subjects included in the model and the estimated Kaplan-Meier survival curve. It can be seen that even in this case the model identifies the right partitions. In particular, the observations in Node 3 have the lower hazard, followed by the observations in Node 4 who have an intermediate level of risk and the observations in Node 5 who have the highest risk of experiencing the event (and consequently the lowest survival probability).

The same graphical representation of CITs can be obtained for RRTs coercing the *rpart* object into a *party* one:

```
library(partykit)
plot(as.party(pruned_RRT))
```



Conditional Inference Trees with *SurvCART*

SCTs can be grown using the *SurvCART* function in *LongCART* package. It requires that the 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**); (iii) the name of the status variable (**censorvar**); (iv) a vector including the names of the partitioning variables to use (**gvars**) and (v) a vector indicating the type of 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 (vi) 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**).

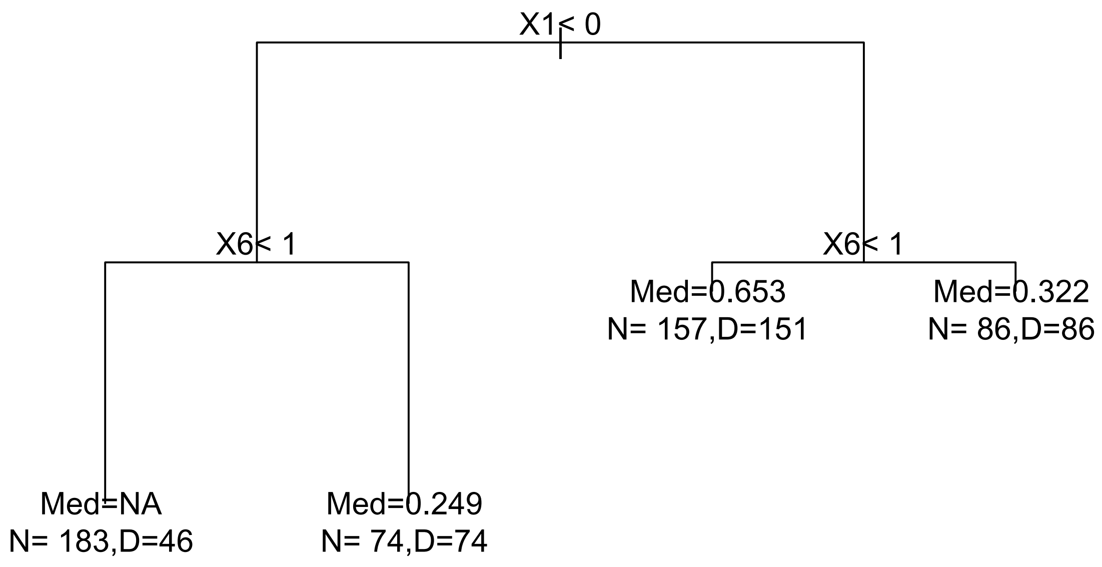
```
# Data pre-processing
data_train_SCT <- data_train
id <- 1:nrow(data_train_SCT)
data_train_SCT <- cbind(data_train_SCT,id)
data_train_SCT[,paste0("X",6:10)] <-
                        lapply(data_train_SCT[,paste0("X",6:10)], as.numeric)
data_train_SCT[, paste0("X",6:10)] <- data_train_SCT[, paste0("X",6:10)]-1

# Model Fitting
```

```
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)
```

```
##   ID   n   D median.T median.C loglik    AIC var index p (Instability) improve
## 1  1 500 357    0.928        3 -592.9 1187.8  X1     0           0.000   117.9
## 2  2 257 120       NA        3 -291.8  585.7  X6     1           0.000   129.1
## 3  4 183  46       NA        3 -153.0  308.1 X10    NA           0.068      NA
## 4  5  74  74    0.249       NA   -9.7   21.5  X2    NA           0.073      NA
## 5  3 243 237    0.488        3 -183.2  368.4  X6     1           0.000    15.3
## 6  6 157 151    0.653        3 -149.8  301.6  X1    NA           0.930      NA
## 7  7  86  86    0.322       NA  -18.1   38.3  X9    NA           0.837      NA
##   Terminal
## 1    FALSE
## 2    FALSE
## 3     TRUE
## 4     TRUE
## 5    FALSE
## 6     TRUE
## 7     TRUE
```

A first output of the tree is the *Treeout* matrix that provides summary information of tree fitting for each node (terminal and non-terminal ones). It contains the identity number of each node (*ID*), the number of observations in each node (*n*), the number of events observed in the node (*D*), the median survival and the median censoring time at each node (*median.T*, *median.C*). It also includes the log-likelihood at each node (*loglik*), the AIC at the node (*AIC*), the splitting variable (*var*), the cut-off value of each splitting variable (*index*), the p-vaue for the parameter instability test (*p(Instability)*), the improvement in deviance given by the split (*improve*) and an indicator variable of terminal node (*Terminal*).

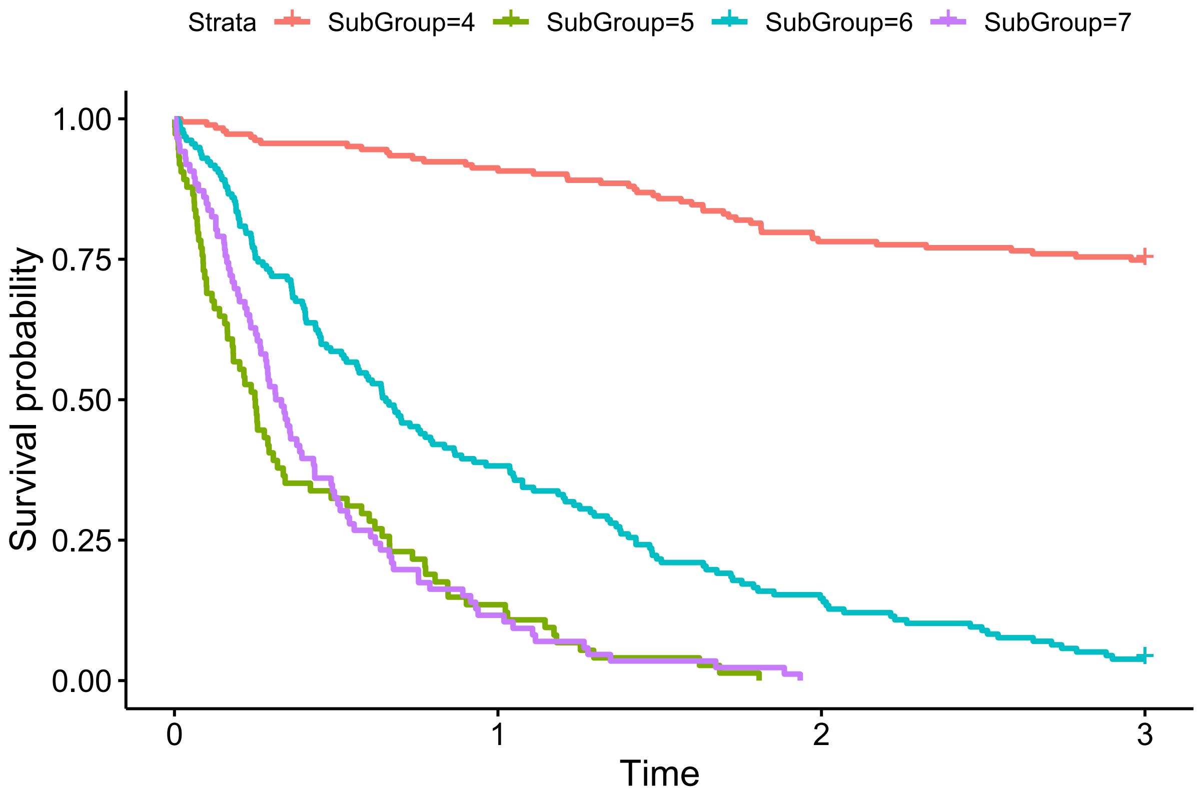
```
# Plotting SCT
par(xpd=TRUE)
plot(SCT, compress=TRUE)
text(SCT, use.n=TRUE)
```



In this case the plot represents a structure that is equal to the equivalent theoretical structure (see the figure of the theoretical structure above). In particular left splits represent the condition written in the node (e.g. the left leave represents observations with a value of $X1 < 0$ and a value of $X6 < 1$, corresponding to $X6 = 0$). For each node the corresponding number of observations and events, together with the estimated median survival time, is shown. It can be seen that for the left node a median survival time equal to NA is estimated. This because the median survival time in that node is greater than the follow-up time (see Subgroup 4 in the figure below). Finally, it is intesting to see that the estimated median survival times for the nodes corresponding to $X1 < 0$ and $X6 = 1$ and $X1 \geq 0$ and $X6 = 1$ are very similar (as expected in the equivalent theoretical structure).

Moreover, the Kaplan-Meier survival curves can be plotted for each terminal node as follows. Each subgroup represents a terminal node; in detail, each subgroup is named as the ID of the corresponding node (ID reported in the *Treeout* matrix).

```
KMPlot.SurvCART(x=SCT, type=1)
```



5. Performance evaluation in the test set

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* (see Supplementary Materials 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* functions for SCTs).

Harrel's C-index

Harrel's C-index can be evaluated through *rcorr.cens* and *Cindex* functions in *Hmisc* and *SurvMetrics* packages respectively.

The first function requires as first object a vector of survival predictions (one for each observation) and a *Surv* object. So, as a first thing the survival predictions must be obtained. If it is not possible to retrieve them (but only risk estimates can be obtained) a possible solution is to replace risk predictions with a decreasing function of these values. This is due to the fact that concordance index 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 one can use the *predict* function. It returns for each observation the corresponding node outcome, i.e. a measure of risk.

```
pred_risk_RRT <- predict(object=pruned_RRT, newdata=data_test)
head(pred_risk_RRT)
```

```
##         2         3         4         8         9        10
## 0.2260154 2.8183587 0.2260154 0.2260154 2.8183587 1.6012796
```

The concordance index can then be evaluated as shown below:

```
# 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)
```

```
##       C Index            Dxy           S.D.              n        missing
##   7.622413e-01   5.244827e-01   2.174412e-02   5.000000e+02   0.000000e+00
##     uncensored Relevant Pairs     Concordant      Uncertain
##   3.600000e+02   2.300400e+05   1.753460e+05   1.946000e+04
```

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. It can be seen that RRT has a good performance, with a C-index equal to 0.76.

For an overall C-index for CITs it is possible to estimate the median survival time for each observation in the test set and to provide it to the function. If *Inf* values are returned, a possible solution is to artificially replace them with a value greater than all the other observed median survival times (of course this trick can be used only for evaluating ranking measures).

```
pred_surv_CIT <- predict(object=CIT, newdata=data_test, type="response")
head(pred_surv_CIT)
```

```
##         2         3         4         8         9        10
##       Inf 0.2880490       Inf       Inf 0.2880490 0.6534732
```

```
# 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)
```

```
##       C Index            Dxy           S.D.              n        missing
##   7.622413e-01   5.244827e-01   2.174412e-02   5.000000e+02   0.000000e+00
##     uncensored Relevant Pairs     Concordant      Uncertain
##   3.600000e+02   2.300400e+05   1.753460e+05   1.946000e+04
```

The fitted CIT performs as well as the RRT (this is an expected result since the two algorithms return the same tree).

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 index
## 0.7622413
```

```
C_ind_CIT <- Cindex(object=ySurv_test, predicted=pred_surv_CIT)
```

```
##       C Index            Dxy           S.D.              n        missing
##   7.622413e-01   5.244827e-01   2.174412e-02   5.000000e+02   0.000000e+00
##     uncensored Relevant Pairs     Concordant      Uncertain
##   3.600000e+02   2.300400e+05   1.753460e+05   1.946000e+04
```

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 is compatible only with a *pec* object; so, the fitted RRT and CIT must be transformed through the *pecRpart* and *pecCtree* functions. Please note that *pecCtree* uses the old *party* package and not *partykit*.

```
library(pec)
RRT_pec <- pecRpart(formula=Surv(eventtime, status)~., data=data_train,
                    cp=bestcp)

CIT_pec <- pecCtree(formula=Surv(eventtime, status)~., data=data_train)
```

The *predictSurvProb* function returns a matrix with a number of rows equal to the number of subjects in the test set and a number of columns equal to the length of *times* vector (*tvec*):

```
pred_RRT_pec <- predictSurvProb(object=RRT_pec, newdata=data_test,
                                times=tvec)
colnames(pred_RRT_pec) <- paste0("time=", tvec)
head(pred_RRT_pec[,c(1,6,11,16,21)])
```

```
##         time=1  time=1.5    time=2   time=2.5     time=3
## [1,] 0.9125683 0.8579235 0.7814208 0.77049180 0.74863388
## [2,] 0.1250000 0.0375000        NA         NA         NA
## [3,] 0.9125683 0.8579235 0.7814208 0.77049180 0.74863388
## [4,] 0.9125683 0.8579235 0.7814208 0.77049180 0.74863388
## [5,] 0.1250000 0.0375000        NA         NA         NA
## [6,] 0.3821656 0.2165605 0.1464968 0.08917197 0.03821656
```

```
pred_CIT_pec <- predictSurvProb(object=CIT_pec, newdata=data_test,
                                times=tvec)
colnames(pred_CIT_pec) <- paste0("time=", tvec)
head(pred_CIT_pec[,c(1,6,11,16,21)])
```

```
##         time=1  time=1.5    time=2   time=2.5     time=3
## [1,] 0.9125683 0.8579235 0.7814208 0.77049180 0.74863388
## [2,] 0.1250000 0.0375000 0.0062500 0.00625000 0.00625000
## [3,] 0.9125683 0.8579235 0.7814208 0.77049180 0.74863388
## [4,] 0.9125683 0.8579235 0.7814208 0.77049180 0.74863388
## [5,] 0.1250000 0.0375000 0.0062500 0.00625000 0.00625000
## [6,] 0.3821656 0.2165605 0.1464968 0.08917197 0.03821656
```

We can see that for RRT there are some NA values. This will imply problems with the evaluation of the performance indices. Going into more details we can see that, for example, the second observation belongs to the node with relative risk estimate equal to 2.8, thus to the right node (as can be seen in the graphical representation of the RRT). It can be noticed that, despite all observations experienced the event, their survival is estimated equal to NA starting from time=2.

```
pred_RRT_node<-predict(object=pruned_RRT, newdata=data_test[2,])
```

```
##        3
## 2.818359
```

This is an error because when the event occurs for all subjects in a group their survival is equal to 0 after the last observed timepoint. Indeed, if we estimate the survival probability in that node in the training set, the resulting curve is the following:

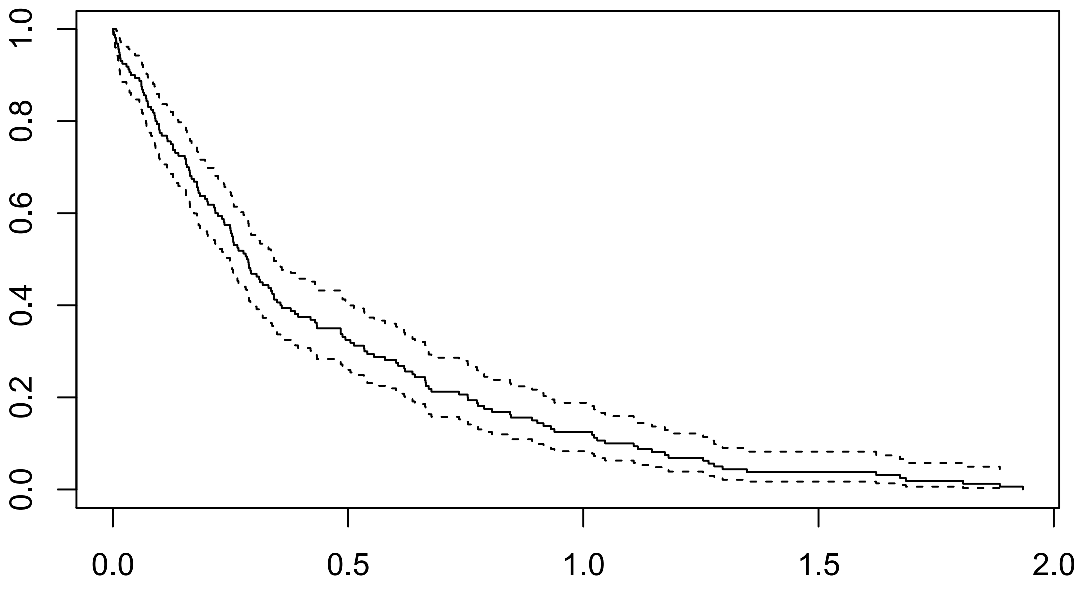
```r
# Risk predictions obtained with RRT in the training set
pred_risk_RRT_train <- predict(object=pruned_RRT, newdata=data_train)
# Subset of observations in the node with highest risk (node in which all
# subjects experience the event)
sub_node <- data_train[which(pred_risk_RRT_train==pred_RRT_node),]

surv_node <- survfit(formula=Surv(eventtime, status)~1, data=sub_node)
surv_node_summary <- cbind(surv_node$time, surv_node$surv)
colnames(surv_node_summary)<- c("Time", "Surv. Prob.")
head(surv_node_summary)
```

```
##             Time Surv. Prob.
## [1,] 0.001000609     0.99375
## [2,] 0.002167734     0.98750
## [3,] 0.004988887     0.98125
## [4,] 0.006337641     0.97500
## [5,] 0.007549165     0.96875
## [6,] 0.010812549     0.96250
```

```r
tail(surv_node_summary)
```

```
##            Time Surv. Prob.
## [155,] 1.622786     0.03125
## [156,] 1.673325     0.02500
## [157,] 1.685413     0.01875
## [158,] 1.807326     0.01250
## [159,] 1.885141     0.00625
## [160,] 1.934352     0.00000
```

```r
plot(surv_node)
```



```
max(sub_node$eventtime)
```

```
## [1] 1.934352
```

It can be seen that after the last observed timepoint, i.e. 1.934351784, the survival probability is equal to 0. So, a possible solution to solve the issue of NA values is to replace them with some consistent values. In this case, for example, after time 1.9 the survival probability can be set equal to 0.

```
pred_surv_RRT_new <- pred_RRT_pec
pred_surv_RRT_new[which(is.na(pred_surv_RRT_new))]<-0
head(pred_surv_RRT_new[,c(1,6,11,16,21)])
```

```
##         time=1  time=1.5    time=2   time=2.5     time=3
## [1,] 0.9125683 0.8579235 0.7814208 0.77049180 0.74863388
## [2,] 0.1250000 0.0375000 0.0000000 0.00000000 0.00000000
## [3,] 0.9125683 0.8579235 0.7814208 0.77049180 0.74863388
## [4,] 0.9125683 0.8579235 0.7814208 0.77049180 0.74863388
## [5,] 0.1250000 0.0375000 0.0000000 0.00000000 0.00000000
## [6,] 0.3821656 0.2165605 0.1464968 0.08917197 0.03821656
```

Once the survival probabilities have been estimated, a single column of the matrix (survival probabilities at a specific timepoint) has to be provided to the function (as **x** argument for *rcorr.cens* and *predicted* in *Cindex*). 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:

```
# Truncated Concordance Index for RRT
C_ind_RRT_vec <- C_ind_RRT_vec1 <- rep(NA, length(tvec))
names(C_ind_RRT_vec) <- names(C_ind_RRT_vec1) <- 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)
   }
```

```
##         1       1.1       1.2       1.3       1.4       1.5       1.6       1.7
## 0.7588490 0.7615150 0.7640076 0.7632779 0.7636416 0.7649862 0.7652261 0.7643473
##       1.8       1.9         2       2.1       2.2       2.3       2.4       2.5
## 0.7638402 0.7633837 0.7473524 0.7475462 0.7448584 0.7423143 0.7436200 0.7433438
##       2.6       2.7       2.8       2.9         3
## 0.7427685 0.7446914 0.7439127 0.7443319 0.7449650
```

```
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_vec1[i] <- Cindex(object=ySurv, predicted=pred_RRT_pec[,i])
   }
```

```
## Error in Cindex(object = ySurv, predicted = pred_RRT_pec[, i]): The input vector cannot have NA
```

```
##         1       1.1       1.2       1.3       1.4       1.5       1.6       1.7
## 0.7588490 0.7615150 0.7640076 0.7632779 0.7636416 0.7649862 0.7652261 0.7643473
##       1.8       1.9         2       2.1       2.2       2.3       2.4       2.5
## 0.7638402 0.7633837        NA        NA        NA        NA        NA        NA
##       2.6       2.7       2.8       2.9         3
##        NA        NA        NA        NA        NA
```

It can be seen that, starting from time 2 the truncated C-index cannot be evaluated with the *Cindex* function in the *SurvMetrics* package, due to the presence of NA in the matrix of survival predictions.

```
# Truncated Concordance Index for CIT
C_ind_CIT_vec <- C_ind_CIT_vec1 <- rep(NA, length(tvec))
names(C_ind_CIT_vec) <- names(C_ind_CIT_vec1) <- 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_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])
  }
```

```
C_ind_CIT_vec
```

```
##         1       1.1       1.2       1.3       1.4       1.5       1.6       1.7
## 0.7588490 0.7615150 0.7640076 0.7632779 0.7636416 0.7649862 0.7652261 0.7643473
##       1.8       1.9         2       2.1       2.2       2.3       2.4       2.5
## 0.7638402 0.7633837 0.7643987 0.7643804 0.7630886 0.7616710 0.7621788 0.7619918
##       2.6       2.7       2.8       2.9         3
## 0.7615452 0.7623159 0.7619367 0.7619821 0.7622413
```

```
C_ind_CIT_vec1
```

```
##         1       1.1       1.2       1.3       1.4       1.5       1.6       1.7
## 0.7588490 0.7615150 0.7640076 0.7632779 0.7636416 0.7649862 0.7652261 0.7643473
##       1.8       1.9         2       2.1       2.2       2.3       2.4       2.5
## 0.7638402 0.7633837 0.7643987 0.7643804 0.7630886 0.7616710 0.7621788 0.7619918
##       2.6       2.7       2.8       2.9         3
## 0.7615452 0.7623159 0.7619367 0.7619821 0.7622413
```

Uno's C-index

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)
```

```
## [1] 0.61984
```

```
UnoC_CIT <- UnoC(Surv.rsp=ySurv_train, Surv.rsp.new=ySurv_test,
                 lpnew= -pred_surv_CIT)
```

```
## [1] 0.61984
```

As expected and as already seen, the two fitted trees (RRT and CIT) have the same performance. The obtained values indicate a moderate performance (lower than that evaluated with Harrel's C-index).

Time-truncated C-index

Finally, an alternative function for evaluating the concordance index is *cindex* in the *pec* package, that evaluates the time-truncated C-index. 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 Inverse Probability Censoring Weights (IPCWs) are available; among these there are the Cox regression PH model (*"cox"*) and the KM estimator (*"marginal"*), that is the 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)
```

```
##
## The c-index for right censored event times
##
## Prediction models:
##
```

```
## RRT CIT
## RRT CIT
##
## Right-censored response of a survival model
##
## No.Observations: 500
##
## Pattern:
##                 Freq
##  event           360
##  right.censored 140
##
## Censoring model for IPCW: marginal model (Kaplan-Meier for censoring distribution)
##
## No data splitting: either apparent or independent test sample performance
##
## Estimated C-index in %
##
## $AppCindex
##     time=1 time=1.1 time=1.2 time=1.3 time=1.4 time=1.5 time=1.6 time=1.7
## RRT   75.9     76.2     76.4     76.3     76.4     76.5     76.5     76.4
## CIT   75.9     76.2     76.4     76.3     76.4     76.5     76.5     76.4
##     time=1.8 time=1.9 time=2 time=2.1 time=2.2 time=2.3 time=2.4 time=2.5
## RRT     76.4     76.3   31.8     32.0     32.2     32.6     32.8     32.9
## CIT     76.4     76.3   76.4     76.4     76.3     76.2     76.2     76.2
##     time=2.6 time=2.7 time=2.8 time=2.9 time=3
## RRT     33.2     33.4     33.5     33.7   33.8
## CIT     76.2     76.2     76.2     76.2   76.2
```

We can see that this function allows to evaluate simultaneously the index at different timepoints. Due to the presence of NA in RRT predictions (as seen above) the evaluation of the index for RRT starting from time 2 is biased.

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 internal function *predictRisk* when evaluating a *ctree* object. The first reason is that *Score* requires a *party* object and not a *partykit* one:

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

```
## Error: Cannot find function (S3-method) called predictRisk.constpartyCannot find
## function (S3-method) called predictRisk.party
```

```
# Fit CIT with the old package party
CIT_party <- party::ctree(Surv(eventtime, status)~., data=data_train)
```

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

The second reason is related to the measure that the function extracts from the tree.

```
## Error: Column 'risk' is type 'list' which is not supported for ordering currently.
```

Indeed, the internal function *riskRegression:::predictRisk.BinaryTree* does not extract the right quantity. Let's see the structure of the function:

```
riskRegression:::predictRisk.BinaryTree
```

```
## function (object, newdata, ...)
## {
##     requireNamespace("party")
##     treeresponse <- party::treeresponse
##     sapply(treeresponse(object, newdata = newdata), function(x) x[1])
## }
## <bytecode: 0x000000002ee66ff0>
## <environment: namespace:riskRegression>
```

It uses the *treeresponse* function. Let's try to use it:

```
predrisk_CIT <- sapply(party:::treeresponse(CIT_party, newdata=data_test),function(x) x[1])
predrisk_CIT[,1:3]
```

```
##           [,1]       [,2]        [,3]
## n         183        160         183
## time      Numeric,47 Numeric,160 Numeric,47
## n.risk    Numeric,47 Numeric,160 Numeric,47
## n.event   Numeric,47 Numeric,160 Numeric,47
## n.censor  Numeric,47 Numeric,160 Numeric,47
## surv      Numeric,47 Numeric,160 Numeric,47
## std.err   Numeric,47 Numeric,160 Numeric,47
## cumhaz    Numeric,47 Numeric,160 Numeric,47
## std.chaz  Numeric,47 Numeric,160 Numeric,47
## type      "right"    "right"     "right"
## logse     TRUE       TRUE        TRUE
## conf.int  0.95       0.95        0.95
## conf.type "log"      "log"       "log"
## lower     Numeric,47 Numeric,160 Numeric,47
## upper     Numeric,47 Numeric,160 Numeric,47
## call      Expression Expression  Expression
```

It can be seen that instead of extracting a measure of risk, *treeresponse* extracts for each observation a list of elements including the timepoints, the number of subjects, the number of events and of censored observations and the estimated survival probability.

The issue can be solved modifying this function, extracting a measure of risk from the tree (e.g. KM hazard estimate):

```
predictRisk.BinaryTree <- function (object, newdata, ...)
{
  requireNamespace("party")
  treeresponse <- party::treeresponse
  sapply(treeresponse(object, newdata = newdata),
         function(x) sum(x$n.event)/x$n)
}
assignInNamespace("predictRisk.BinaryTree", predictRisk.BinaryTree,
                  pos="package:riskRegression")

predrisk_CIT <- riskRegression:::predictRisk.BinaryTree(CIT_party, data_test)
head(predrisk_CIT)
```

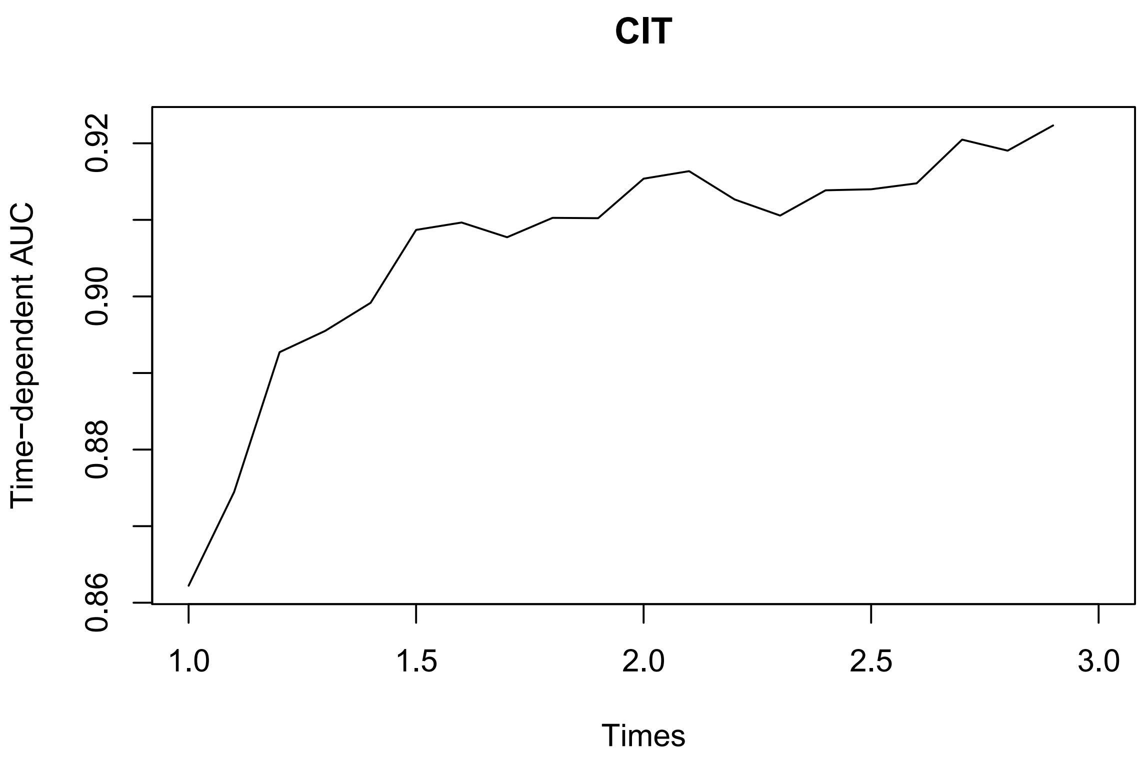
```
## [1] 0.2513661 1.0000000 0.2513661 0.2513661 1.0000000 0.9617834
```

Now, the function seems to work correctly:

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

```
##
## Metric AUC:
##
## Results by model:
##
##         model times  AUC lower upper
##  1: CIT_party     1 86.2  83.2  89.3
##  2: CIT_party     1 87.4  84.5  90.4
##  3: CIT_party     1 89.3  86.6  91.9
##  4: CIT_party     1 89.5  87.0  92.1
##  5: CIT_party     1 89.9  87.4  92.5
##  6: CIT_party     2 90.9  88.5  93.2
##  7: CIT_party     2 91.0  88.6  93.3
##  8: CIT_party     2 90.8  88.4  93.2
##  9: CIT_party     2 91.0  88.7  93.4
## 10: CIT_party     2 91.0  88.7  93.4
## 11: CIT_party     2 91.5  89.2  93.8
## 12: CIT_party     2 91.6  89.3  93.9
## 13: CIT_party     2 91.3  88.9  93.6
## 14: CIT_party     2 91.1  88.7  93.4
## 15: CIT_party     2 91.4  89.0  93.7
## 16: CIT_party     2 91.4  89.0  93.7
## 17: CIT_party     3 91.5  89.1  93.8
## 18: CIT_party     3 92.0  89.8  94.3
## 19: CIT_party     3 91.9  89.6  94.2
## 20: CIT_party     3 92.2  90.0  94.4
## 21: CIT_party     3  NaN   NaN   NaN
##         model times  AUC lower upper

##
## NOTE: Values are multiplied by 100 and given in %.

## NOTE: The higher AUC the better.
```

```
plot(x=tvec, y=AUC_Score_CIT$AUC$score$AUC, type="l", xlab="Times",
     ylab="Time-dependent AUC", main="CIT")
```



Evaluating time-dependent AUCs for an *rpart* object with *Score* is simpler. An example is provided below:

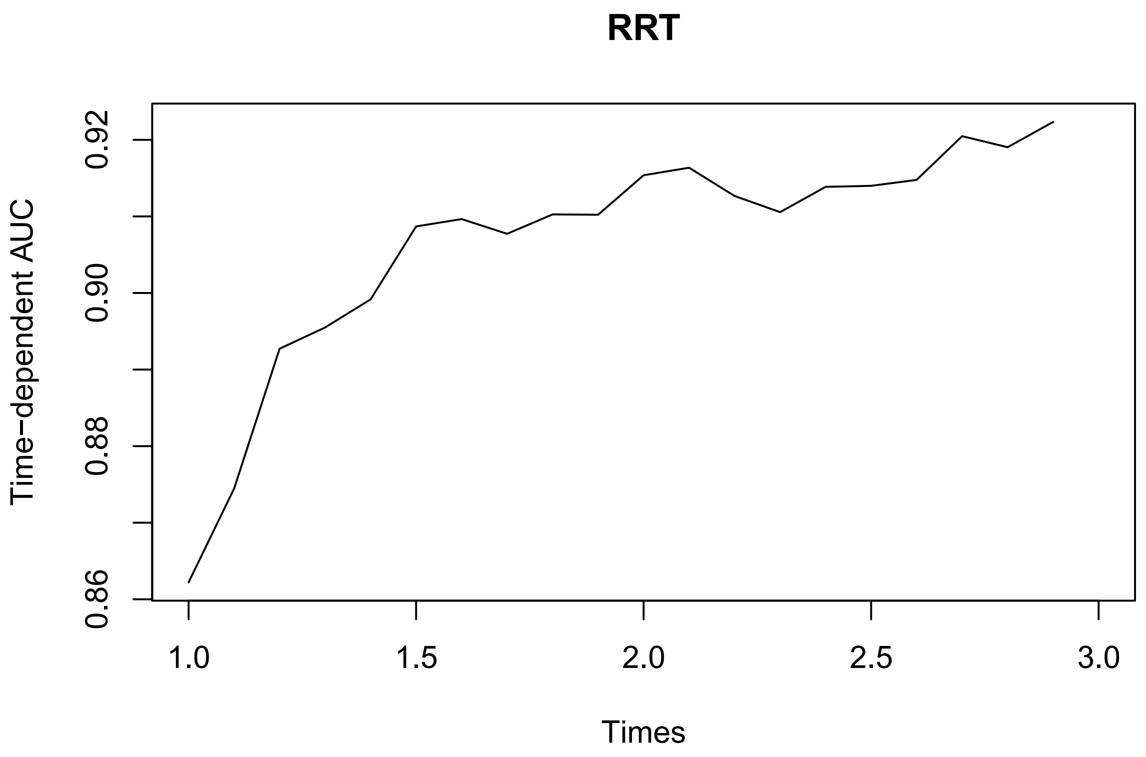
```
AUC_Score_RRT <- Score(object=list("RRT"=pruned_RRT), metrics="AUC",
                       formula=Surv(eventtime, status)~X1+X2+X3+X4+X5+X6+X7+X8+
                       X9+X10, data=data_test, cens.model="km",
                       times=tvec)
```

```
##
## Metric AUC:
##
## Results by model:
##
##     model times  AUC lower upper
##  1:   RRT     1 86.2  83.2  89.3
##  2:   RRT     1 87.4  84.5  90.4
##  3:   RRT     1 89.3  86.6  91.9
##  4:   RRT     1 89.5  87.0  92.1
##  5:   RRT     1 89.9  87.4  92.5
##  6:   RRT     2 90.9  88.5  93.2
##  7:   RRT     2 91.0  88.6  93.3
##  8:   RRT     2 90.8  88.4  93.2
##  9:   RRT     2 91.0  88.7  93.4
## 10:   RRT     2 91.0  88.7  93.4
## 11:   RRT     2 91.5  89.2  93.8
## 12:   RRT     2 91.6  89.3  93.9
## 13:   RRT     2 91.3  88.9  93.6
## 14:   RRT     2 91.1  88.7  93.4
## 15:    RRT     2 91.4  89.0  93.7
## 16:    RRT     2 91.4  89.0  93.7
## 17:    RRT     3 91.5  89.1  93.8
## 18:    RRT     3 92.0  89.8  94.3
## 19:    RRT     3 91.9  89.6  94.2
## 20:    RRT     3 92.2  90.0  94.4
## 21:    RRT     3  NaN   NaN   NaN
##      model times  AUC lower upper
```

```
##
## NOTE: Values are multiplied by 100 and given in %.
```

```
## NOTE: The higher AUC the better.
```

```
plot(x=tvec, y=AUC_Score_RRT$AUC$score$AUC, type="l", xlab="Times",
     ylab="Time-dependent AUC", main="RRT")
```



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
##  [1] 0.7715580 0.7883319 0.8098143 0.8136056 0.8202333 0.8387887 0.8435947
##  [8] 0.8499367 0.8528467 0.8587003 0.8715396 0.8771286 0.8634870 0.8439041
## [15] 0.8465800 0.8466676 0.8379205 0.8436683 0.8509202 0.8448936 0.0000000
##
## $times
##  [1] 1.0 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2.0 2.1 2.2 2.3 2.4 2.5 2.6 2.7 2.8
## [20] 2.9 3.0
##
## $iauc
## [1] 0.7868997
##
## attr(,"class")
## [1] "survAUC"
```

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

```
## $auc
##  [1] 0.7715580 0.7883319 0.8098143 0.8136056 0.8202333 0.8387887 0.8435947
##  [8] 0.8499367 0.8528467 0.8587003 0.8715396 0.8771286 0.8634870 0.8439041
## [15] 0.8465800 0.8466676 0.8379205 0.8436683 0.8509202 0.8448936 0.0000000
##
## $times
##  [1] 1.0 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2.0 2.1 2.2 2.3 2.4 2.5 2.6 2.7 2.8
## [20] 2.9 3.0
##
## $iauc
## [1] 0.7868997
##
## attr(,"class")
## [1] "survAUC"
```

Brier Score and Integrated Brier Score

Two of the available packages for evaluating Brier Score (BS) and Integrated Brier Score (IBS) are *SurvMetrics* and *pec*. The two functions in *SurvMetrics* are *Brier* and *IBS*. They both 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. It uses KM IPCWs. An example is provided below.

```
library(SurvMetrics)
# Evaluation of BS at time 1 for RRT
brier_RRT_1 <- Brier(object=ySurv_test, pre_sp=pred_RRT_pec[,1], t_star=1)
```

```
## Brier Score
##   0.1347755
```

```
# Evaluation of BS at time 2 for RRT
brier_RRT_2 <- Brier(object=ySurv_test, pre_sp=pred_RRT_pec[,11], t_star=2)
```

```
## Error: The input probability vector cannot have NA
```

```
# Evaluation of BS at time 1 for CIT
brier_CIT_1 <- Brier(object=ySurv_test, pre_sp=pred_CIT_pec[,1], t_star=1)
```

```
## Brier Score
##   0.1347755
```

```
# Evaluation of BS at time 2 for CIT
brier_CIT_2 <- Brier(object=ySurv_test, pre_sp=pred_CIT_pec[,11], t_star=2)
```

```
## Brier Score
##   0.0921548
```

Even in this case the two models have an equal performance at time 1 (remember that lower values of BS indicate better performance). It can be seen that, as hinted, from time 2 the BS cannot be evaluated for the RRT using the output of *predictSurvProb*, due to the presence of NA. The problem can be solved using the matrix in which we have replaed NAs with zeros. Using these estimates the function provide the same results of CIT, suggesting that the solution makes sense.

```
brier_RRT_2 <- Brier(object=ySurv_test, pre_sp=pred_surv_RRT_new[,11], t_star=2)
```

```
## Brier Score
##  0.09216824
```

Integrated Brier Score can then be evaluated as follows for RRT and CIT. However, the *IBS* function is not compatible with NA values, so it is not evaluable for RRT using the matrix obtaiend with *predictSurvProb*. The issue can be solved using the matrix with the zeros instead of NAs.

```
ibs_RRT <- IBS(object=ySurv_test, sp_matrix=pred_RRT_pec, IBSrange=tvec)
```

```
## Error: The input probability matrix cannot have NA
```

```
ibs_RRT <- IBS(object=ySurv_test, sp_matrix=pred_surv_RRT_new, IBSrange=tvec)
```

```
##        IBS
## 0.09799359
```

```
ibs_CIT <- IBS(object=ySurv_test, sp_matrix=pred_CIT_pec, IBSrange=tvec)
```

```
##        IBS
## 0.09798687
```

Both the two trees have a satisfactory performance.

The two indices can also be evaluated through the *pec* function in the homonymous package. This function allows to specify a different censoring model than KM (*"marginal"*). BS can be evaluated as follows:

```
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")
```

```
##
## Prediction error curves
##
## Prediction models:
##
## Reference       RRT       CIT
## Reference       RRT       CIT
##
## Right-censored response of a survival model
##
## No.Observations: 500
##
## Pattern:
##                 Freq
##  event          360
##  right.censored 140
##
## IPCW: marginal model
##
## No data splitting: either apparent or independent test sample performance
##   time n.risk Reference   RRT   CIT
## 1    1    241      0.25 0.135 0.135
```

Finally, the IBS can be evaluated as below:

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

```
##
## Prediction error curves
##
## Prediction models:
##
## Reference       RRT
## Reference       RRT
##
## Right-censored response of a survival model
##
## No.Observations: 500
##
## Pattern:
##                 Freq
##  event          360
##  right.censored 140
##
## IPCW: marginal model
##
## No data splitting: either apparent or independent test sample performance
##
## Cumulative prediction error, aka Integrated Brier score  (IBS)
##  aka Cumulative rank probability score
##
## Range of integration: 0 and time=1.9 :
##
##
## Integrated Brier score (crps):
##
##           IBS[0;time=1.9)
## Reference           0.216
## RRT                 0.121
```

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

```
##
## Prediction error curves
##
```

```
## Prediction models:
##
## Reference       CIT
## Reference       CIT
##
## Right-censored response of a survival model
##
## No.Observations: 500
##
## Pattern:
##                 Freq
##  event          360
##  right.censored 140
##
## IPCW: marginal model
##
## No data splitting: either apparent or independent test sample performance
##
## Cumulative prediction error, aka Integrated Brier score  (IBS)
##  aka Cumulative rank probability score
##
## Range of integration: 0 and time=3 :
##
##
## Integrated Brier score (crps):
##
##           IBS[0;time=3)
## Reference         0.216
## CIT               0.109
```

Also for this package the problem of NA values occurs. It can be noticed, indeed, that for RRT the IBS is evaluated until time=1.9 (just right before the first timepoint with NA values). However, for *pec* it is not easily possible to solve the issue because the function requires the tree object and not the predicted survival probabilities.

A2. Structure of the trees objects in R

In this document, an example of how extracting information from the three examined tree algorithms is provided.

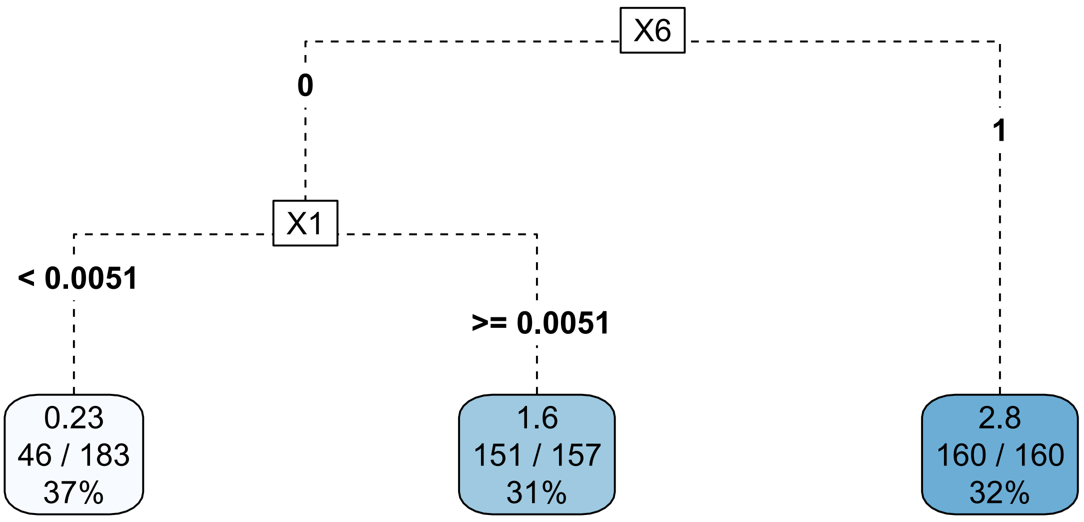
Data Simulation

```
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

### Generate Standard Normal independent r.v.
library(mvtnorm)
set.seed(70522)
Norm_RV <- rmvnorm(n=N, mean=rep(0,n_rv/2), sigma=diag(n_rv/2))
### Generate Bernoulli independent r.v.
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)

### Definition of the partitions
node1 <- cov[,"X6"]==1
node2 <- cov[,"X6"]==0 & cov[,"X1"]>=0
node3 <- cov[,"X6"]==0 & cov[,"X1"]<0

maxtime <- 3                     #Length of the follow-up

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, maxt=maxtime,
                              x=as.data.frame(1:sum(node1)), seed=7052022)[,2:3]
# Node2
simdata[node2,1:2] <- simsurv(dist="exponential", lambda=0.9, maxt=maxtime,
                              x=as.data.frame(1:sum(node2)), seed=7052022)[,2:3]
# Node3
simdata[node3,1:2] <- simsurv(dist="exponential", lambda=0.1, maxt=maxtime,
                              x=as.data.frame(1:sum(node3)), seed=7052022)[,2:3]

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

### Training and testing sets
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,]
```

Relative Risk Trees

```
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)
# Plot RRT
library(rpart.plot)
rpart.plot(x= pruned_RRT, type= 5, tweak=1.0, gap=0.02, branch.lty=2)
```



The main object representing the structure of the Relative Risk Tree (RRT) is the *frame* data frame. Each row represents one node. The columns include:

- *"var"*, a factor with the name of the splitting variable (or if the node is terminal it returns "<leaf>");
- *"n"*, the number of observations in the node;

- *"yval"*, the outcome at the node.

Further information are also available in the function's help (see *?rpart.object*).

```
pruned_RRT$frame
```

```
##      var   n  wt      dev      yval  complexity ncompete nsurrogate     yval2.1
## 1     X6 500 500 780.1231 1.0000000 0.218058734        4          5   1.0000000
## 2     X1 340 340 463.9415 0.6559016 0.218058734        4          5   0.6559016
## 4 <leaf> 183 183 179.6887 0.2260154 0.009346277        0          0   0.2260154
## 5 <leaf> 157 157 112.3782 1.6012796 0.004791221        0          0   1.6012796
## 3 <leaf> 160 160 147.8310 2.8183587 0.008424651        0          0   2.8183587
##       yval2.2
## 1 357.0000000
## 2 197.0000000
## 4  46.0000000
## 5 151.0000000
## 3 160.0000000
```

Starting from this dataframe, the size of the tree (number of overall nodes, terminal nodes and splits) can be obtained as follows:

```
nodes_RRT <- nrow(pruned_RRT$frame)                   # Number of overall nodes
```

```
## [1] 5
```

```
term_nodes_RRT <- sum(pruned_RRT$frame$var=="<leaf>") # Number of terminal nodes
```

```
## [1] 3
```

```
nsplit_RRT <- nodes_RRT - term_nodes_RRT              # Number of splits
```

```
## [1] 2
```

As can be seen also by the above figure, RRT has 5 nodes, of which 3 are terminal nodes, determined by 2 splits.

The splitting variables can be determined through the *frame* matrix, excluding the rows of terminal nodes.

```
split_vars_RRT <- pruned_RRT$frame[-which(pruned_RRT$frame[,1]=="<leaf>"),"var"]
```

```
## [1] "X6" "X1"
```

The splitting variables are $X6$ and $X1$. The cut-off values used for the splits are recorded in the *splits* matrix of the *rpart* object. The row label is the name of the split variable. Columns include:

- *"count"*, the number of observations sent left or right by the split;
- *"ncat"*, the number of categories/levels for the variable (+/-1 for a continuous variable; the sign determines whether the subset $x < cutpoint$ or $x > cutpoint$ is sent to the left);
- *"improve"*, the improvement in deviance given by the split;
- *"index"*, the numeric split point if the splitting variable is continuous. If the splitting variable is a factor it reports the row number of the *csplit* matrix.
- *"adj"*, the adjusted concordance for surrogate splits.

```
pruned_RRT$splits
```

```
##    count ncat     improve       index         adj
## X6   500    2 168.3721078 1.000000000 0.000000000
## X1   500   -1 143.4749318 0.002519567 0.000000000
## X2    500    1   7.5533962  1.585330148 0.000000000
## X5    500   -1   6.3595358  1.989192954 0.000000000
## X4    500    1   5.2467345  1.774409986 0.000000000
## X3      0    1   0.6920000 -2.088180969 0.037500000
## X1      0   -1   0.6880000  2.322958899 0.025000000
## X2      0    1   0.6860000 -2.647884604 0.018750000
## X5      0   -1   0.6840000  2.563222676 0.012500000
## X4      0    1   0.6820000 -2.250619612 0.006250000
## X1    340   -1 171.8893227  0.005131272 0.000000000
## X5    340   -1   5.9499559  1.767275761 0.000000000
## X2    340    1   4.8396352  1.577770805 0.000000000
## X4    340   -1   4.0196610 -1.708539523 0.000000000
## X3    340    1   3.8843340  0.660761765 0.000000000
## X5      0    1   0.5647059 -0.480078346 0.057324841
## X3      0    1   0.5588235 -0.992950301 0.044585987
## X10     0    2   0.5500000  2.000000000 0.025477707
## X2      0    1   0.5441176 -1.837232024 0.012738854
## X4      0    1   0.5411765 -0.955194476 0.006369427
```

For categorical variables the *csplit* matrix, together with the *splits* matrix, has to be examined. It includes one row for each categorical splitting variable, while the number of columns corresponds to the largest number of levels in the factors. The row to examine is given by the *index* column of the *splits* matrix. The columns record 1 if that level of the factor goes to the left, 3 if it goes to the right and 2 if that level is not present at that node of the tree.

```
pruned_RRT$csplit
```

```
##      [,1] [,2]
## [1,]    1    3
## [2,]    3    1
```

For example, for $X6$ the *index* value in the *splits* matrix is equal to 1, so the first row of *csplit* must be considered. The first level of the factor (0) goes to the left, while the second level (1) goes to the right.

To extract the splitpoint (the row of *csplit*) for the continuous (categorical) splitting variable the following procedure can be used. The elements to take into account are the number of observations in the node and the splitting variable used. If the same splitting variable is repeated more than once in the matrix the first row is considered (the one with the highest improvement).

```
#### Cut-off values
splits_RRT <- pruned_RRT$splits
frame_RRT <- pruned_RRT$frame
cutoff_RRT <- rep(NA,length(split_vars_RRT))
ncat_RRT <- rep(NA,length(split_vars_RRT))

for(j in 1:length(split_vars_RRT))
  {
    temp <- which(splits_RRT[,1]==frame_RRT[j,"n"] &
                rownames(splits_RRT)==split_vars_RRT[j])
    #If the same variable is repeated more times the first row is considered
    cutoff_RRT[j] <- splits_RRT[temp[1],"index"]
    ncat_RRT[j] <- splits_RRT[temp[1],"ncat"]
  }

split_vars_RRT
```

```
## [1] "X6" "X1"
```

```
cutoff_RRT
```

```
## [1] 1.000000000 0.005131272
```

```
pruned_RRT$csplit[cutoff_RRT[1],]
```

```
## [1] 1 3
```

```
ncat_RRT
```

```
## [1]  2 -1
```

The first splitting variable is $X6$ that is a factor, and we already know that the cut-off is 0/1. Looking at the first row of *csplit* we can see that the observations with $X6 = 0$ go to the left node. The cut-off for the continuous outcome $X1$ is 0.0051. Since the corresponding *ncat* is equal to -1 the observations with values of $X1 < 0.0051$ go to the left.

To extract the outcomes of the nodes, i.e. the relative risk at each node, the following R code can be used:

```
outcome_nodes_RRT <- frame_RRT[which(frame_RRT[,1]=="<leaf>"),"yval"]
```

```
## [1] 0.2260154 1.6012796 2.8183587
```

The outcomes are in order from the left leaf to the right one. They are respectively equal to 0.2260154, 1.6012796 and 2.8183587.

Finally, the paths of the tree can be identified as follows:

```
paths_RRT <- rpart.plot:::rpart.rules(pruned_RRT,roundint=FALSE)
```

```
##  Surv
##  0.23 when X6 is 0 & X1 <  0.0051
##  1.60 when X6 is 0 & X1 >= 0.0051
##  2.82 when X6 is 1
```

The resulting output is a dataframe. To extract the relative information (e.g. to understand which node is split) the following procedure can be used:

```
node_split <- matrix(NA, nrow=2, ncol=term_nodes_RRT)
for(i in 1:term_nodes_RRT)
  {
    if(paths_RRT[i,6]=="&")
      {
        node_split[1,i] <-  paths_RRT[i,c(3)]
        node_split[2,i] <- if(paths_RRT[i,6]=="&") paths_RRT[i,c(5)]
      }
  }
```

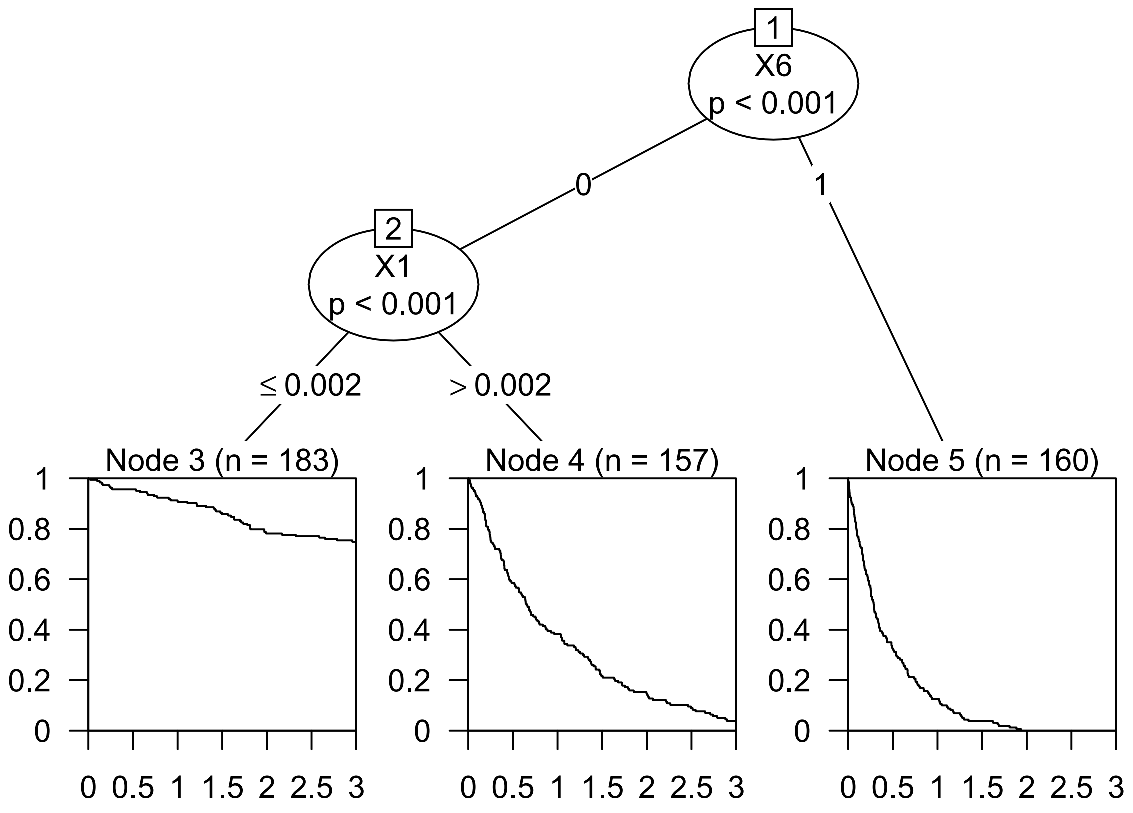
```
##      [,1] [,2] [,3]
## [1,] "X6" "X6" NA
## [2,] "0"  "0"  NA
```

The node that is split is therefore the one corresponding to $X6 = 0$.

Conditional Inference Trees with the *ctree* algorithm

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

```r
# Plot CIT
plot(CIT)
```



The size of Conditional Inference Trees obtained with the *ctree* algorithm (CIT) can be obtained as follows:

```r
nodes_CIT <- length(CIT)                      # Number of overall nodes
```

```
## [1] 5
```

```r
term_nodes_CIT <- width(CIT)                  # Number of terminal nodes
```

```
## [1] 3
```

```r
n_splits_CIT <- nodes_CIT - term_nodes_CIT   # Number of splits
```

```
## [1] 2
```

The fitted CIT (as can be observed also through the plot) has 5 overall nodes; 3 of them are terminal nodes, determined by 2 splits.

To extract the splitting variables the structural change test (statistic test for building the tree) can be examined. It can be obtained through the *sctest* function in the *strucchange* package.

```r
library(strucchange)
SCTest <- sctest(CIT)
```

```
## $`1`
##                       X1        X2        X3         X4        X5           X6
## statistic 1.009557e+02 3.1213762 1.2541127 0.03599432 0.3448064 1.391055e+02
## p.value   9.406198e-23 0.5525553 0.9525715 0.99999999 0.9997093 4.176692e-31
##                  X7          X8       X9       X10
## statistic 0.2245846 0.001217413 2.635009 0.5981145
## p.value   0.9999587 1.000000000 0.668485 0.9969287
##
## $`2`
##                     X1       X2       X3        X4        X5         X7
## statistic 1.154079e+02 0.316541 1.603844 0.5965181 0.1103783 0.02653715
## p.value   5.766036e-26 0.999535 0.873661 0.9945760 0.9999945 0.99999999
##                  X8        X9       X10
## statistic 0.1076238 0.4704989 0.3152374
## p.value   0.9999951 0.9977770 0.9995427
##
## $`3`
##                   X1        X2        X3        X4        X5          X7
## statistic 0.02309543 0.0303257 0.5426397 0.9627379 0.7927230 0.001994507
## p.value   0.99999999 1.0000000 0.9961821 0.9714865 0.9850832 1.000000000
##                  X8        X9        X10
## statistic 0.2969120 0.3206509 6.50066478
## p.value   0.9996414 0.9995101 0.09296824
##
## $`4`
##                  X1        X2        X3        X4        X5          X7
## statistic 0.9939474 2.2488608 0.8112503 0.9552758 1.6853453 0.007226842
## p.value   0.9684071 0.7252378 0.9838566 0.9721957 0.8567914 1.000000000
##                   X8         X9       X10
## statistic 0.04947138 0.02608566 0.1722870
## p.value   0.99999984 0.99999999 0.9999629
##
## $`5`
##                  X1        X2          X3        X4         X5        X7
## statistic 0.0310211 0.8888394 0.003814942 0.8308227 0.02630194 1.3386237
## p.value   1.0000000 0.9780496 1.000000000 0.9824946 0.99999999 0.9224257
##                   X8       X9       X10
## statistic 0.03470195 5.357168 0.1375877
## p.value   0.99999997 0.171119 0.9999858
```

```
# Number of elements in the list SCTest, i.e. number of nodes of the tree
l_SCTest <- length(SCTest)
```

```
## [1] 5
```

```
# When the list is composed of only one element (tree with only 1 node),
# l_SCTest is equal to the length of the matrix (2 rows and 1 column for each
# variable) If this occurs l_SCTest should be fixed equal to 1.

if(l_SCTest==2*n_rv)  l_SCTest <- 1
```

The idea is to extract, for each node, the variable that is strongly associated to the outcome (in a significant way, i.e. $p \leq \alpha = 0.05$). Indeed, this is the way how the algorithm selects the splitting variables.

```
alpha <- 0.05

split_vars_CIT <- matrix(NA,nrow=1,ncol=l_SCTest)
for(j in 1:l_SCTest)
  {
    if(is.null(SCTest[[j]])) split_vars_CIT[1,j] <- "NULL"
    else
      {
        split_vars_CIT[1,j] <- min(SCTest[[j]]["p.value",]) #takes the minimum p
        if(as.numeric(split_vars_CIT[1,j])< alpha) # check significance
          split_vars_CIT[1,j] <- names(which.min(SCTest[[j]][2,]))
        else
          split_vars_CIT[1,j] <- "No sig"
      }
  }

split_vars_CIT
```

```
##      [,1] [,2] [,3]     [,4]     [,5]
## [1,] "X6" "X1" "No sig" "No sig" "No sig"
```

So, it can be seen, that the splitting variables used to grow the tree are $X6$ and $X1$.

For looking at the splitpoints the paths of the CIT can be considered. Then, treating it as a character we can split the string and identify the cut-off:

```
path_CIT <- partykit:::.list.rules.party((CIT))
```

```
##                                                3
## "X6 %in% c(\"0\") & X1 <= 0.00205327735949113"
##                                                4
##  "X6 %in% c(\"0\") & X1 > 0.00205327735949113"
##                                                5
##                               "X6 %in% c(\"1\")"
```

```
cutoff_CIT <- rep(NA,n_splits_CIT)

if (is.factor(data_train[,split_vars_CIT[1]]))
  {
    temp <- strsplit(path_CIT[1],"")
    cutoff_CIT[1] <- temp[[1]][12]
}

if (is.numeric(data_train[,split_vars_CIT[2]]))
  {
    temp <- strsplit(path_CIT[1],"=")
    cutoff_CIT[2] <- as.numeric(sapply(temp, `[`, 2))
  }
```

```
## [1] "0"                   "0.00205327735949113"
```

For extracting the outcome of CIT, i.e. the median survival time, the predict function can be used:

```
unique(predict(CIT,type="response"))
```

```
## [1] 0.2880490 0.6534732       Inf
```

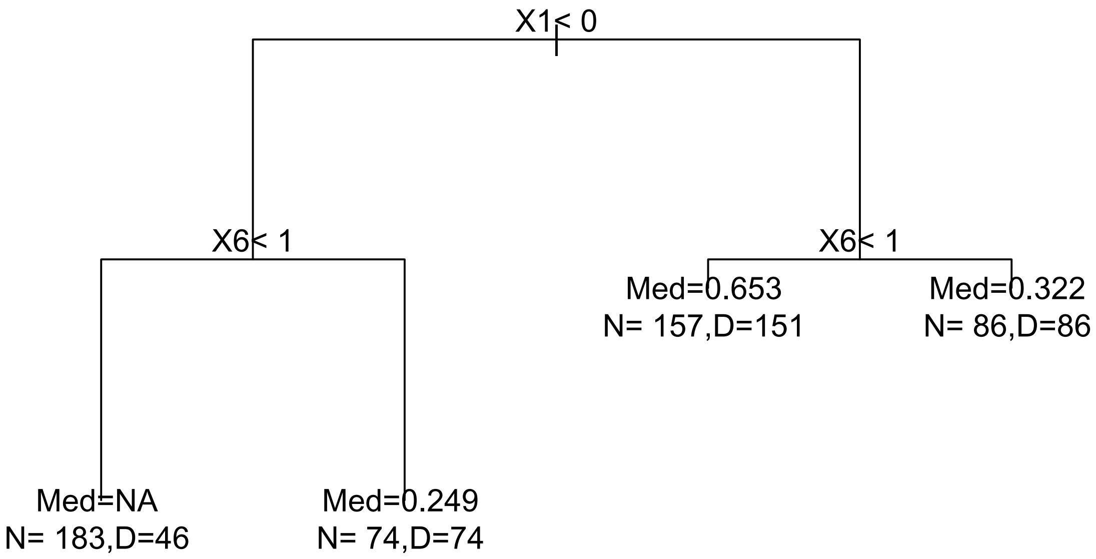
The estimated median survival times for the first two nodes are 0.2880490 and 0.6534732. The last estimated median survival time is set equal to Inf because the length of the follow-up is shorter than the median survival time (less than 50% of the entire sample experiences the event at the end of follow-up).

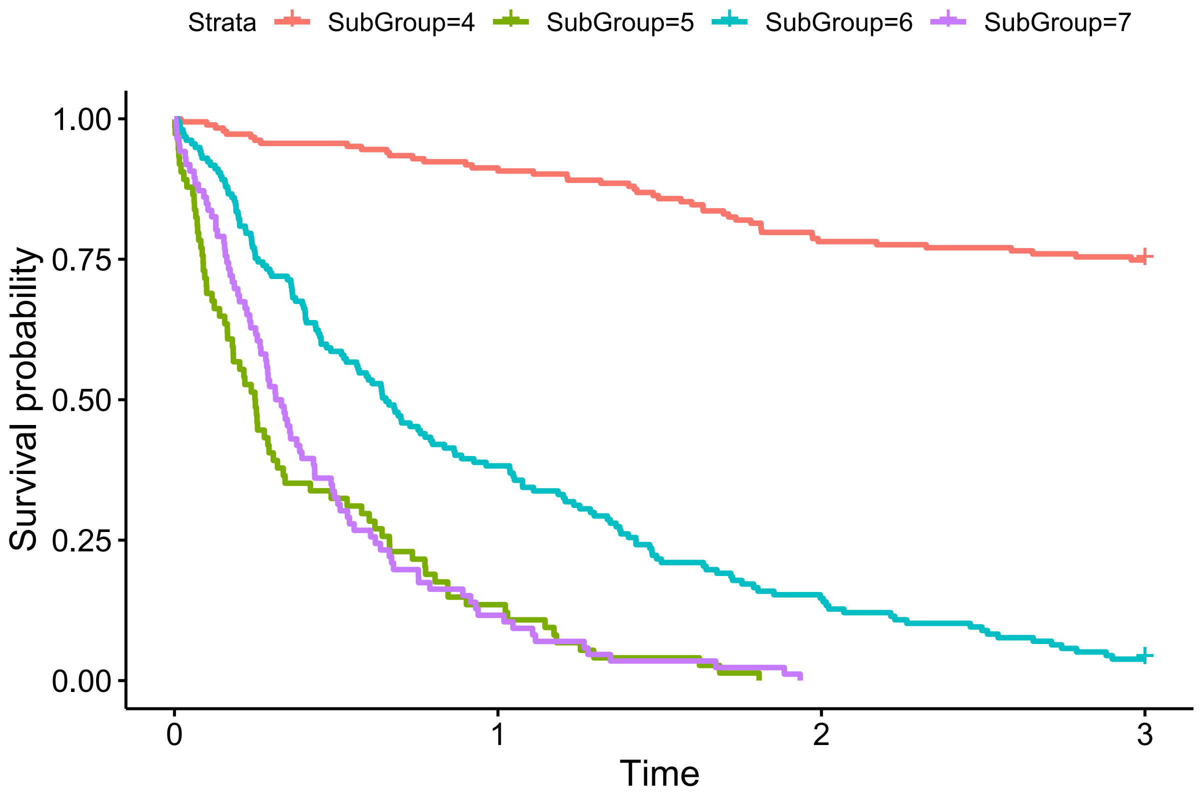
Conditional Inference Trees with the *SurvCART* algorithm

```
# 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

# Model Fitting
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)
```

```
##   ID   n   D median.T median.C loglik    AIC var index p (Instability) improve
## 1  1 500 357    0.928        3 -592.9 1187.8  X1     0           0.000   117.9
## 2  2 257 120       NA        3 -291.8  585.7  X6     1           0.000   129.1
## 3  4 183  46       NA        3 -153.0  308.1 X10    NA           0.068      NA
## 4  5  74  74    0.249       NA   -9.7   21.5  X2    NA           0.073      NA
## 5  3 243 237    0.488        3 -183.2  368.4  X6     1           0.000    15.3
## 6  6 157 151    0.653        3 -149.8  301.6  X1    NA           0.930      NA
## 7  7  86  86    0.322       NA  -18.1   38.3  X9    NA           0.837      NA
##   Terminal
## 1    FALSE
## 2    FALSE
## 3     TRUE
## 4     TRUE
## 5    FALSE
## 6     TRUE
## 7     TRUE
```

```
# Plot SCT
par(xpd = TRUE)
plot(SCT, compress = TRUE)
text(SCT, use.n = TRUE)

# Plot Kaplan-Meier curves estimated in the terminal nodes
KMPlot.SurvCART(x= SCT, type= 1)
```


X1< 0
X6< 1
X6< 1
Med=NA
N= 183,D=46
Med=0.249
N= 74,D=74
Med=0.653
N= 157,D=151
Med=0.322
N= 86,D=86



All the information about the structure of Conditional Inference Trees obtained with the *SurvCART* algorithm (SCT) are included in the *Treeout* dataframe. The *Treeout* matrix includes one row for each node. The columns include:

- *"ID"*, the identity number of the node;
- *"n"*, the number of observations in the node;
- *"D"*, the number of events in the node;
- *"median.T"*, the estimated median survival time in the node;
- *"median.C"*, the estimated median censoring time in the node;
- *"var"*, the splitting variable used for growing the tree;
- *"index"*, the cut-off that has determined the split;
- *"Terminal"*, the indicator of terminal node.

Further information about the structure of the dataframe *Treeout* can be found in the help function (?SurvCART).

```
SCT$Treeout
```

```
##   ID   n   D median.T median.C loglik    AIC var index p (Instability) improve
## 1  1 500 357    0.928        3 -592.9 1187.8  X1     0           0.000   117.9
## 2  2 257 120       NA        3 -291.8  585.7  X6     1           0.000   129.1
## 3  4 183  46       NA        3 -153.0  308.1 X10    NA           0.068      NA
## 4  5  74  74    0.249       NA   -9.7   21.5  X2    NA           0.073      NA
## 5  3 243 237    0.488        3 -183.2  368.4  X6     1           0.000    15.3
## 6  6 157 151    0.653        3 -149.8  301.6  X1    NA           0.930      NA
## 7  7  86  86    0.322       NA  -18.1   38.3  X9    NA           0.837      NA
##   Terminal
## 1    FALSE
## 2    FALSE
## 3     TRUE
## 4     TRUE
## 5    FALSE
## 6     TRUE
## 7     TRUE
```

The size of the tree can be obtained with the procedure below:

```
nodes_SCT <- nrow(SCT$Treeout)                            # Number of overall nodes
```

```
## [1] 7
```

```
term_nodes_SCT <- sum(SCT$Treeout[,"Terminal"]=="TRUE") # Number of terminal nodes
```

```
## [1] 4
```

```
nsplits_SCT <- nodes_SCT - term_nodes_SCT               # Number of splits
```

```
## [1] 3
```

The tree has 7 nodes, of which 4 are leaves. The number of splits needed to grow the tree is equal to 3.

The splitting variables and the cut-offs can be determined as follows:

```
vars_SCT <- matrix(NA,nrow=1,ncol=nsplits_SCT)
splits_SCT <- matrix(NA,nrow=1,ncol=nsplits_SCT)
for(j in 1:nsplits_SCT)
  {
   vars_SCT[1,j] <-
                 SCT$Treeout[which(SCT$Treeout[,"Terminal"]=="FALSE"),"var"][j]
   splits_SCT[1,j] <-
                SCT$Treeout[which(SCT$Treeout[,"Terminal"]=="FALSE"),"index"][j]
  }

vars_SCT
```

```
##      [,1] [,2] [,3]
## [1,] "X1" "X6" "X6"
```

```
splits_SCT
```

```
##      [,1] [,2] [,3]
## [1,]    0    1    1
```

So, the splitting variables are $X1$, $X6$ and $X6$. Ths cut-off values for the continuous variable $X1$ is 0.

The outcome, i.e. the median survival time, at each node can be retrieved as follows:

```
outcome_SCT <- matrix(NA,nrow=1,ncol=nsplits_SCT)
for(j in 1:nsplits_SCT)
  {
    outcome_SCT[1,j] <-
              SCT$Treeout[which(SCT$Treeout[,"Terminal"]=="TRUE"),"median.T"][j]
  }

outcome_SCT
```

```
##      [,1]  [,2]  [,3]
## [1,]   NA 0.249 0.653
```

The first node has a median survival time equal to NA because the follow-up duration is shorter than the median survival time. The other estimated median survival times are 0.249 and 0.653.

References

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

Y. Amit and D. Geman. Shape quantization and recognition with randomized trees. *Neural computation*, 9(7):1545–1588, 1997.

S. Bennett. Analysis of survival data by the proportional odds model. *Statistics in medicine*, 2(2):273–277, 1983a.

S. Bennett. Log-logistic regression models for survival data. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 32(2):165–171, 1983b.

B. Bieszk-Stolorz and K. Dmytrów. Application of the survival trees for estimation of the propensity to accepting a job and resignation from the labour office mediation by the long-term unemployed people. In *International Conference on Computational Methods in Experimental Economics*, pages 141–154. Springer, 2017.

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

I. Bou-Hamad, D. Larocque, and H. Ben-Ameur. Discrete-time survival trees and forests with time-varying covariates: application to bankruptcy data. *Statistical Modelling*, 11(5):429–446, 2011a.

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

L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001a.

L. Breiman. Statistical Modeling: The Two Cultures (with comments and a rejoinder by the author). *Statistical Science*, 16(3):199 – 231, 2001b. doi: 10.1214/ss/1009213726. URL `https://doi.org/10.1214/ss/1009213726`.

L. Breiman. Manual–Setting Up, Using, and Understanding Random Forests, v 4.0. *Using_random_forests_ V3. 0. pdf*, 2003. URL `ftp://ftp.stat.berkeley.edu/pub/users/breiman`.

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

N. Breslow. Covariance analysis of censored survival data. *Biometrics*, pages 89–99, 1974.

N. E. Breslow. Contribution to discussion of paper by DR Cox. *J. Roy. Statist. Soc., Ser. B*, 34:216–217, 1972.

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

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

J. Buckley and I. James. Linear regression with censored data. *Biometrika*, 66(3): 429–436, 1979.

L. Camilleri. History of survival analysis. 2019.

A. Ciampi, J. Thiffault, J.-P. Nakache, and B. Asselain. 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, 1986.

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

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

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

D. R. Cox. Regression Models and Life-Tables. *Journal of the Royal Statistical Society. Series B (Methodological)*, 34(2):187–220, 1972. ISSN 00359246. URL `http://www.jstor.org/stable/2985181`.

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

G. Csurilla and I. Fertő. How long does a medal win last? Survival analysis of the duration of Olympic success. *Applied Economics*, pages 1–15, 2022.

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

A. De Rose and A. Pallara. Survival trees: An alternative non-parametric multivariate technique for life history analysis. *European Journal of Population/Revue Européenne de Démographie*, 13(3):223–241, 1997.

J. Del Corral, C. P. Barros, and J. Prieto-Rodriguez. The determinants of soccer player substitutions: A survival analysis of the Spanish soccer league. *Journal of Sports Economics*, 9(2):160–172, 2008.

A. Ekman. Variable selection for the Cox proportional hazards model: A simulation study comparing the stepwise, lasso and bootstrap approach, 2017.

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

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

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

K. D. Fynn and M. Sonnenschein. An analysis of the career length of professional basketball players. *The Macalester Review*, 2(2):3, 2012.

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

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

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

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

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

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

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

T. Hastie, R. Tibshirani, and M. Wainwright. Statistical learning with sparsity. *Monographs on statistics and applied probability*, 143:143, 2015.

N. L. Hjort and A. Koning. Tests for constancy of model parameters over time. *Journal of Nonparametric Statistics*, 14(1-2):113–132, 2002.

T. Hothorn and A. Zeileis. partykit: A modular toolkit for recursive partytioning in R. *The Journal of Machine Learning Research*, 16(1):3905–3909, 2015.

T. Hothorn and A. Zeileis. Predictive distribution modeling using transformation forests. *Journal of Computational and Graphical Statistics*, 30(4):1181–1196, 2021.

T. Hothorn, P. Bühlmann, S. Dudoit, A. Molinaro, and M. J. Van Der Laan. Survival ensembles. *Biostatistics*, 7(3):355–373, 2006a.

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

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

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

T. Hothorn, L. Möst, and P. Bühlmann. Most likely transformations. *Scandinavian Journal of Statistics*, 45(1):110–134, 2018.

H. Ishwaran, E. H. Blackstone, C. E. Pothier, and M. S. Lauer. Relative risk forests for exercise heart rate recovery as a predictor of mortality. *Journal of the American Statistical Association*, 99(467):591–600, 2004.

H. Ishwaran, U. B. Kogalur, E. H. Blackstone, and M. S. Lauer. Random survival forests. *The annals of applied statistics*, 2(3):841–860, 2008.

H. Ishwaran, U. B. Kogalur, and M. U. B. Kogalur. Package 'randomforestsrc'. *breast*, 6:1, 2022.

R. A. Jack, K. R. Sochacki, T. Hirase, J. Vickery, P. C. McCulloch, D. M. Lintner, and J. D. Harris. Performance and return to sport after hip arthroscopic surgery in major league baseball players. *Orthopaedic Journal of Sports Medicine*, 7(2): 2325967119825835, 2019.

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

G. James, D. Witten, T. Hastie, and R. Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.

J. D. Kalbfleisch and R. L. Prentice. Marginal likelihoods based on Cox's regression and life model. *Biometrika*, 60(2):267–278, 1973.

E. l. Kaplan and M. Paul. Nonparametric estimation from incomplete observations. *Journal of the American Statistical Association*, 53(282):457–481, 1958. ISSN 01621459.

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

D. G. Kleinbaum, M. Klein, et al. *Survival analysis: a self-learning text*, volume 3. Springer, 2012.

J. Kropko and J. H. Jeffrey. coxed: An R package for Computing Duration-Based Quantities from the Cox Proportional Hazards Model. *R J.*, 11(2):38, 2019.

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

M. G. Kundu and S. Ghosh. 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, 2021.

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

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

R. A. Levine, J. Fan, X. Su, and M. E. Nunn. Bayesian survival trees for clustered observations, applied to tooth prognosis. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 7(2):111–124, 2014.

A. Liaw, M. Wiener, et al. Classification and regression by randomForest. *R news*, 2 (3):18–22, 2002.

A. R. Linero, P. Basak, Y. Li, and D. Sinha. Bayesian survival tree ensembles with submodel shrinkage. *Bayesian Analysis*, 17(3):997–1020, 2022.

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

A. Macis. Survival trees: a pathway among features and open issues of the main R packages. *Electronic Journal of Applied Statistical Analysis*, 2022a. (Forthcoming).

A. Macis. Recursive Partitioning for Survival Data. In *Book of the Short Papers - SIS 2022 51st Scientific Meeting of the Italian Statistical Society*, pages 1754—1759. Pearson, 2022b. URL `https://it.pearson.com/content/dam/region-core/italy/pearson-italy/pdf/Docenti/Universit%C3%A0/Sis-2022-4c-low.pdf`.

A. Macis, M. Manisera, M. Sandri, and P. Zuccolotto. A Survival Analysis Study to Discover Which Skills Determine a Higher Scoring in Basketball. *Statistica Applicata - Italian Journal of Applied Statistics*, 2022a. (Submitted).

A. Macis, M. Manisera, M. Sandri, and P. Zuccolotto. Which Achievements Are Associated With a Better Offensive Performance in NBA? A Survival Analysis Study. In *13th World Congress of Performance Analysis of Sport (WCPAS2022) the 13th International Symposium on Computer Science in Sport (IACSS2022), Proceedings*. Springer, 2022b. (Submitted).

H. T. Mai, D. S. Chun, A. D. Schneider, B. J. Erickson, R. D. Freshman, B. Kester, N. N. Verma, and W. K. Hsu. Performance-based outcomes after anterior cruciate ligament reconstruction in professional athletes differ between sports. *The American journal of sports medicine*, 45(10):2226–2232, 2017.

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

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

K. Moulds, S. Abbott, J. Pion, C. Brophy-Williams, M. Heathcote, and S. Cobley. Sink or swim? A survival analysis of sport dropout in Australian youth swimmers. *Scandinavian journal of medicine & science in sports*, 30(11):2222–2233, 2020.

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

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

S. Y. Park, J. E. Park, H. Kim, and S. H. Park. 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, 2021.

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

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

J. Pion, M. Lenoir, B. Vandorpe, and V. Segers. Talent in female gymnastics: a survival analysis based upon performance characteristics. *International journal of sports medicine*, 94(11):935–940, 2015.

S. Potapov, W. Adler, M. Schmid, and M. S. Potapov. Package 'survauc'. *Statistics in Medicine*, 25:3474–3486, 2012.

R Core Team. *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria, 2021. URL `https://www.R-project.org/`.

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

M. Schmid, H. Küchenhoff, A. Hoerauf, and G. Tutz. A survival tree method for the analysis of discrete event times in clinical and epidemiological studies. *Statistics in medicine*, 35(5):734–751, 2016.

M. Schumacher, G. Bastert, H. Bojar, K. Hübner, M. Olschewski, W. Sauerbrei, C. Schmoor, C. Beyerle, R. Neumann, and H. Rauschecker. Randomized 2 x 2 trial evaluating hormonal treatment and the duration of chemotherapy in node-positive breast cancer patients. German Breast Cancer Study Group. *Journal of Clinical Oncology*, 12(10):2086–2093, 1994.

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

N. Simon, J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for Cox's proportional hazards model via coordinate descent. *Journal of statistical software*, 39(5):1, 2011.

K. R. Sochacki, R. A. Jack, T. Hirase, J. Vickery, and J. D. Harris. Performance and return to sport after hip arthroscopy for femoracetabular impingement syndrome in National Hockey League players. *Journal of Hip Preservation Surgery*, 6(3):234–240, 2019.

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

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

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

F. Tang and H. Ishwaran. Random forest missing data algorithms. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 10(6):363–377, 2017.

K. Tay, N. Simon, J. Friedman, T. Hastie, R. Tibshirani, and B. Narasimhan. Regularized Cox Regression, 2022.

T. Therneau and E. Atkinson. The concordance statistic. 2020.

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

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

T. M. Therneau, E. J. Atkinson, et al. An introduction to recursive partitioning using the RPART routines. Technical report, Technical report Mayo Foundation, 2022.

R. Tibshirani. The lasso method for variable selection in the cox model. *Statistics in medicine*, 16(4):385–395, 1997.

J. Tobin. Estimation of relationships for limited dependent variables. *Econometrica: journal of the Econometric Society*, pages 24–36, 1958.

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

M. J. Van der Laan and J. M. Robins. *Unified methods for censored longitudinal data and causality*, volume 5. Springer, 2003.

M. J. van der Laan, S. Dudoit, and A. W. van der Vaart. The cross-validated adaptive epsilon-net estimator. *Statistics & Decisions*, 24(3):373–395, 2006.

P. J. Verweij and H. C. Van Houwelingen. Penalized likelihood in Cox regression. *Statistics in medicine*, 13(23-24):2427–2436, 1994.

B. Vinzamuri and C. K. Reddy. Cox regression with correlation based regularization for electronic health records. In *2013 IEEE 13th International Conference on Data Mining*, pages 757–766. IEEE, 2013.

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

D. B. Wangrow, D. J. Schepker, and V. L. Barker III. Power, performance, and expectations in the dismissal of NBA coaches: A survival analysis study. *Sport Management Review*, 21(4):333–346, 2018.

M. N. Wright and A. Ziegler. ranger: A Fast implementation of Random Forests for High Dimensional Data in C++ and R. *Journal of Statistical Software*, 77(1):1–17, 2017. doi: 10.18637/jss.v077.i01. URL `https://www.jstatsoft.org/index.php/jss/article/view/v077i01`.

A. Zeileis and K. Hornik. Generalized M-fluctuation tests for parameter instability. *Statistica Neerlandica*, 61(4):488–508, 2007.

A. Zeileis and T. Hothorn. Parties, Models, Mobsters: A New Implementation of Model-Based Recursive Partitioning in R, 2015.

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

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

H. Zhang and B. H. Singer. *Recursive Partitioning and Applications*. Springer Science & Business Media, 2010.

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

H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the royal statistical society: series B (statistical methodology)*, 67(2):301–320, 2005.

P. Zuccolotto and M. Manisera. *Basketball data science: with applications in R*. CRC Press, 2020.