MASTER THESIS
STATISTICS AND DATA SCIENCE
UNIVERSITEIT LEIDEN

A simulation study evaluating the effect of dependent censoring on survival curves and the performance of Inverse Probability Censoring Weights

Author:
Georgios Kalkantzis

Thesis Supervisor:
Dr.ir. N. (Nan) van Geloven
Medical statistics
Department of Biomedical Data Sciences
Leiden University Medical Center

Thesis Supervisor:
Dr. Sanne J.W. Willems
Methodology and Statistics
Institute of Psychology
Leiden University

June 2022

Abstract

Survival analysis studies time-to-event outcomes. One of the main characteristics of survival data is that some survival times are not observed, we call those observations censored. Standard methods to analyze censored data, like the Kaplan-Meier estimator or the Cox proportional hazards model, assume that censored observations are independent of the time to event and we call this type of censoring non-informative. In real life studies however, this is not always the case and the censoring may depend on time to event either directly or through covariates. In that case the censoring is called informative or dependent and using the standard methods can lead to biased results.

In this thesis we examined first how serious the issue of dependent censoring is by generating data with dependent censoring using two methods, one for two time-independent covariates and one for two time-independent and one time-dependent covariate, and studying how much bias is introduced if we assume independent censoring in the analysis.

Different approaches have been proposed in order to correct for the issue of dependent censoring, one of them is Inverse Probability Censoring Weighting (IPCW). In the second part, we will perform a simulation study to evaluate the performance of the IPCW method in the presence of dependent censoring, for each of the two methods we examined in the first part.

Results showed that the survival curves estimated by the traditional Kaplan-Meier method have only small bias in most cases. The bias increased when the dependent censoring gets stronger. The IPCW method overall performs well and corrects for the presence of dependent censoring but it is not able to correct the bias fully in case the dependency is too strong or when we introduced a time dependent covariate which is subject to measurement error.

Acknowledgements

I would like to express my deepest gratitude to my supervisors Dr. ir. N.(Nan) van Geloven and Dr. Sanne J.W. Willems for their support, encouragement and patience. I met with both of them almost in weekly basis. These meetings have been extremely useful for the progress of the thesis and were also helpful for other issues not directly involve with the thesis. Additionally, their reviews and extensive comments on various thesis drafts made this work possible.

Finally I want to thank my parents and friends for their unconditional support and encouragement through out my study.

Contents

Chapter 1

Introduction

Survival analysis focuses on time-to-event outcomes. Time can mean for example years, months or days from the beginning of the follow up until the event occurs. The event can be death, relapse, disease occurrence, recovery or any other experience of interest. The main issue that arises from studying time-to-event outcomes is that for some individuals the exact survival time is unknown although we have some information of their survival time. This problem is called censoring and the most frequent one and the one that will be the focus of this thesis is right censoring. Right censoring occurs when an individual leaves the study before the event occurs or when the study ends without the occurrence of the event of interest. Standard methods of analyzing survival data like the Kaplan-Meier method or the Cox proportional hazards model assume that censoring is independent or non-informative which means that the probability for an individual to be censored at any given time t is independent of the probability to experience the event at time t.

1.1 Motivation for this Thesis

However, with real data the assumption of independent censoring does not always hold. If we assume that censoring is independent in the analysis model, but censoring is in reality informative or dependent, this can lead to biased results. Dependent censoring can occur when time to censoring and time to event are dependent either directly or through covariates. Willems [12] conducted a simulation study to evaluate the performance of the inverse probability of censoring weighting (IPCW) method in the case of dependent censoring. The generated survival data had two independent covariates that influence both the time to event and time to censoring. Event and censoring times were both generated from the exponential distribution and constant hazard rates were assumed for both models. The results showed that the survival probabilities estimates were more accurate with the use of IPCW and that the standard Kaplan-Meier method overestimated the survival probabilities. However, authors had simulated extreme dependence of the censoring mechanism on the predictor variables in order to show notable differences in the estimates. Such extreme dependence is not expected to occur in real data. Furthermore, in another study by Matsuyama and Yamaguchi [5] an adjusted version of IPCW was developed to correct for dependent censoring in settings with more than one reason for censoring (competing risks). There are many examples where in applications the IPCW adjusted curves differs very little from the adjusted ones, which suggested that dependent censoring was not a real issue for this analysis.

1.2 Aims of this Thesis

The results from those two papers raise the question of how serious the threat of dependent censoring is and when it is necessary to account for it, or if the analysis without correction for it will give acceptable results. This will be the focus of the first part of this thesis, where we will perform a simulation study to answer this question. We will generate survival data that contain dependent censoring by using different approaches, for example assume constant hazard rates or include a time varying covariate. We will check to which extent standard survival analysis methods give biased results and determine when a correction method is required to account for it.

In the second part of the thesis, we will evaluate the performance of Inverse Probability of Censoring Weighting (IPCW). This method corrects for dependent censoring by giving extra weights to subjects that are not censored at time t and have similar characteristics as the subjects that were censored. We will use the ipw package for R in order to apply the IPCW method and explain how to prepare the data we generated in order to apply the ipw package. Then we will apply the IPCW method to the data we generated in the first part using different approaches, and run simulations studies to evaluate its performance compared to the standard method that assumes independent censoring.

1.3 Structure of this Thesis

In Chapter 2 some basic concepts of survival analysis will be introduced in order to help the reader with no or little experience in survival analysis to have a better understanding of the following chapters. Chapter 3 will consist of two parts. In the first part there will be a description of the methods used to generate survival times and censoring times for time independent covariates. In second part the results of the simulations will be presented evaluating how important the issue of dependent censoring can be. In Chapter 4 in the first part, the method to generate dependent censoring for time-independent covariates and one time-dependent covariate is described. In the second part the results of the simulations for time-independent covariates and one time-dependent covariate will be presented. In Chapter 5 the IPCW method will be described in detail, and the results of the simulation study to evaluate the performance of IPCW method in the presence of dependent censoring with only time-independent covariates. In Chapter 6 the results of the simulation study to evaluate the performance of IPCW method with time-independent covariates and one time-dependent covariate will be presented. Finally the R code that used for this thesis will be in the Appendix B.

Chapter 2

Survival analysis theory

In this chapter some basic concepts of survival analysis will be introduced. Notation and theory are based on Klein and Moeschberger [4].

2.1 Basic functions of survival analysis

Survival analysis is a branch of statistics that consists of methods of analyzing data in which the outcome variable of interest is time to an event. Event may be death, occurrence of a disease, relapse from remission, equipment failure and so forth, the event can be positive also, such as recovery from surgery or any other event of interest than can happen to the individual. Let X be the time until the even of interest, $X > 0$ is random variable and is called survival time. In the following paragraphs we will describe two functions that characterize the distribution of X, namely the *survival function* and the *hazard function.*

2.1.1 The Survival Function

The basic quantity of survival analysis is the survival function, which is the probability of an individual to survive to time x and it is defined as

$$S(x) = P(X > x). \tag{2.1}$$

If X is a continuous random variable, then $S(X)$ is a continuous strictly decreasing function. If $F(X)$ is the cumulative distribution function of X, where $F(x) = P(X \leq x)$, then $S(x) = 1 - F(x)$. The integral of the probability density function $f(x)$ is equal to the survival function,

$$S(x) = P(X > x) = \int_x^\infty f(t)dt. \tag{2.2}$$

Thus,

$$f(x) = -\frac{dS(x)}{dx}. \tag{2.3}$$

The survival curves are monotone, nonincreasing with probability equal to one at time zero and probability equal to zero as time goes to infinity.

2.1.2 The Hazard Function

The hazard function sometimes called *a conditional failure rate* gives the rate at which an individual will experience the event in the next instant of time given that he or she has survived up to time x. The hazard rate is defined as

$$h(x) = \lim_{\Delta x \to 0} \frac{P[x \leq X < x + \Delta x | X \geq x]}{\Delta x}. \tag{2.4}$$

For a continuous random variable X,

$$h(x) = \frac{f(x)}{S(x)} = -\frac{d\ln[S(x)]}{dx}. \tag{2.5}$$

From equation (2.4) we have the ratio of two quantities, the numerator is a conditional probability and the denominator denotes a small interval of time. The result of this division is probability per unit time, which is a rate and not a probability, thus it takes values in the range between zero and infinity. The hazard function curves are nonnegative $h(x) \geq 0$, do not have an upper bound and can have many different shapes.

The *cumulative hazard function* $H(x)$ can derive from (2.5) and it is defined as

$$H(x) = \int_0^x h(t)dt = -\ln[S(x)]. \tag{2.6}$$

Thus the survival function for continuous lifetimes is defined as

$$S(x) = \exp[-H(x)] = \exp\left[-\int_0^x h(t)dt\right]. \tag{2.7}$$

2.2 Censoring and Truncation

A problem that occurs frequently in survival data is that time to event for some individuals is not exactly known or it may have occured within a certain interval, this problem is called censoring. The three main categories of censoring are, right censoring, left censoring, and interval censoring. Another issue with survival data is truncation which occurs when only the event times of individuals that lie within a certain interval can be observed. There are two categories of truncation, right truncation and left truncation. In this section we will define briefly the different types of censoring and truncation.

2.2.1 Right Censoring

Right censoring, which is the most common type of censoring in survival time occurs when the event is not observed before a certain time point, which we call the *censoring time* C_r, in that case the individual for which the event has not observed is called *censored.*

For a subject in a study, if X is the lifetime and C_r is the censoring time, we assume the $X's$ are independent and identically distributed with probability density function $f(x)$ and survival function $S(x)$. For an individual in the study the event will be observed if and only if, X is less or equal to C_r, if X is bigger than C_r then the individual is censored at C_r.

Survival data can be represented by pairs of random variables (T, δ), where $\delta = 1$ if the event was observed and $\delta = 0$ for censored subjects. T represent the time to event and it equal to X if the event was observed, and to C_r if the event is not observed,i.e. $T = min(X, C_r)$.

2.2.2 Left Censoring

Left censoring occurs when the event of interest happened before a certain time point C_l, which means that the event of interest has occured for that person before his or her inclusion in the study at time C_l.

Survival data for left censored observations can be represented by pairs of random variables (T, δ), where $\delta = 1$ if the exact lifetime is observed and $\delta = 0$ if not. T is equal to X if the event is observed and equal to C_l if the event is not observed,i.e. $T = max(X, C_l)$.

Sometimes *left censoring* and *right censoring* can occur simultaneously in a study, in that case the lifetime are considered *double censored.* Again the survival data can be represented as a pair of variables (T, δ), where $T = max[min(X, C_r), C_l]$, and δ is 1 if the event time is observed, 0 if it is right censored and, -1 if it is left censored.

2.2.3 Interval Censoring

A more general type of censoring called *interval censoring* occurs when the lifetime is observed but we do not know the exact time of the event but only that it occured in a specific time interval.

This type of censoring often occurs in clinical trials or longitudinal studies where the subjects have follow-up times and the event of interest may happened between two follow-up times, in that case the exact life time is unknown but it is known to be into the time interval between the two visits $(L_i, R_i]$.

2.2.4 Truncation

Another issue that often occurs in survival studies is *truncation.* The issue arises when only subjects that have event times that are within a specific observational time frame (Y_L, Y_R) are observed. For a subject that the event time is outside this interval, his or her event time is not observed and there is no information available.

When Y_R is infinite *left truncation* occurs, in that case we only observe subjects that their event time is greater than Y_L. An example of *left truncation* is entry into a retirement home where subjects from a certain age, who have not experience the event of interest yet, entered the retirement home, subjects that have experience the event before this age can not enter the retirement home and there is no information about them.

Right truncation occurs when Y_L is zero, in that case only individuals that experience the event prior to right truncation time, Y_R, are included in the study. An example of *right truncation*

2.3 Estimation of the Survival and Cumulative Hazard Functions

The survival function and cumulative hazard function can be estimated by parametric and non-parametric methods. We will describe both of these approaches in this section.

2.3.1 Non-Parametric Methods

In a sample of right censored survival data each of the n subjects is represented by a pair of random variables (t_i, δ_i), where t_i represents event time or censoring time for each subject, and δ_i is the indicator of whether the time is an event time or a censoring time. If all the event times t_i occur at D distinct times $t_1 < t_2 < ... < t_D$, and d_i are the number of events. Let Y_i be the number of subjects that are at risk at time t_i. The quantity d_i/Y_i is an estimate of the conditional probability that a subject experiences the event of interest at time t_i, given that he or she survived until time t_i. This quantity will be the basic tool to construct the two non-parametric estimators described in this section for the survival and the cumulative hazard functions.

Product-Limit Estimator

An estimator proposed by Kaplan and Meier (1958) called Product-Limit estimator (also known as Kaplan-Meier estimator) is the standard estimator for the survival function, and it is defined as

$$\hat{S}(t) = \begin{cases} 1 & \text{if} \quad t < t_1, \\ \prod_{t_i \le t} \left[1 - \frac{d_i}{Y_i}\right], & \text{if} \quad t \ge t_1 \end{cases} . \tag{2.8}$$

The Product-Limit estimator is well defined for all time points t_i that are smaller than the largest observed time t_{max}. This estimator is a step function with jumps at each time point t_i, with the size of the jumps proportional to the number of events and the number of censored observations at time point t_i.

From the Product-Limit estimator we can estimate the cumulative hazard function. $H(t) = -\ln[S(t)]$,

$$\hat{H}(t) = -\ln[\hat{S}(t)]. \tag{2.9}$$

Nelson-Aalen Estimator

The second estimator that was suggested first by Nelson (1972) and rediscovered later by Aalen (1978), is called Nelson-Aalen estimator for cumulative hazard function and it is well defined up to the largest observed time as

$$\tilde{H}(t) = \begin{cases} 0 & \text{if} \quad t < t_1, \\ \sum_{t_i \le t} \frac{d_i}{Y_i} & \text{if} \quad t \ge t_1 \end{cases}. \tag{2.10}$$

From the Nelson-Aalen estimator for the cumulative hazard function we can derive an estimator for the survival function as follows,

$$\tilde{S}(t) = \exp[-\tilde{H}(t)]. \tag{2.11}$$

Both of these estimators are based on the assumption that censoring is non-informative. When this assumption does not hold then both of these estimators estimate the wrong function and the results can be misleading.

2.3.2 Parametric Methods

In the previous section we described two non-parametric methods to estimate the survival function and the cumulative hazard function. In this section we will briefly describe the most common parametric survival models that are used to estimate the survival function. In a parametric survival model the survival times are assumed to follow a known distribution, such as, the *exponential*, the *Weibull*, the *Gompertz*, and the *log-logistic*. Table 2.1 lists the basic survival functions for these distributions.

Distribution	Hazard Rate $h(x)$	Survival Function $S(x)$	Probability Density Function $f(x)$
Exponential $\lambda > 0, x \ge 0$	λ	$\exp[-\lambda x]$	$\lambda \exp[-\lambda x]$
Weibull $\alpha, \lambda > 0, x \ge 0$	$\alpha \lambda x^{\alpha-1}$	$\exp[-\lambda x^{\alpha}]$	$\alpha \lambda x^{\alpha-1} \exp[-\lambda x^{\alpha}]$
Gompertz $\alpha, \beta > 0, x \ge 0$	$\beta \exp^{\alpha x}$	$\exp\left[-\frac{\beta}{\alpha}(\exp^{\alpha x} - 1)\right]$	$\beta \exp^{\alpha x} \exp\left[-\frac{\beta}{\alpha}(\exp^{\alpha x} - 1)\right]$
Log-logistic $\alpha, \beta > 0, x \ge 0$	$\frac{(\alpha/\beta)(x/\beta)^{\alpha-1}}{1+(x/\beta)^{\alpha}}$	$\frac{1}{1+(x/\beta)^{\alpha}}$	$\frac{(\alpha/\beta)(x/\beta)^{\alpha-1}}{(1+(x/\beta)^{\alpha})^2}$

Table 2.1: Parametric distributions in survival analysis

The exponential and the Weibull are the most common distributions for modeling lifetime data. The exponential is the most simple distribution to model survival data based on the constant hazard rate and the memoryless property, although those two properties limit its applicability in survival analysis. The popularity of Weibull distribution arises from its flexibility to model survival data. Its hazard rate can be monotone increasing (for $\alpha > 1$), monotone decreasing (for $0 < \alpha < 1$), or constant (for $a = 1$).

2.4 Semi-Parametric Models

Often, subjects in a survival study have characteristics that may effect their time to event, those characteristics can range from age, gender, education to physical activity, smoking and drinking habits or can be a vast range of biomarkers. In order then to have more accurate results the survival function must adjust to account for these variables. A method to account for these covariates was introduced by Cox (1972) and is called Cox proportional hazards model.

2.4.1 Cox model with time-independent covariates

Let n be the size of a sample of survival data, we can describe each subject with the triple $(T_j, \delta_j, \mathbf{Z}_j)$, with $j = 1, 2, ..., n$, where T_j is the event time for the j subject, δ_j is the event indicator for the j subject and $\mathbf{Z} = (Z_1, Z_2, ..., Z_p)$ is the vector of p covariates for each subject j.

The hazard rate for time t for the proportional hazard model is defined as

$$h(t|\mathbf{Z}) = h_0(t)c(\boldsymbol{\beta}^t\mathbf{Z}), \tag{2.12}$$

where $h_0(t)$ is an unspecified baseline hazard rate, $\boldsymbol{\beta}^t = (\beta_1, \beta_2, ..., \beta_p)$ is a vector of p parameters, and $c(\boldsymbol{\beta}^t\mathbf{Z})$ is a known function linking the covariates with the parameters. Another advantage of this model besides the fact that can adjust for covariates is that the unspecified baseline hazard gives the model flexibility. In practice this means even without a specific form for the baseline hazard function the Cox model we can obtain good estimates for the survival curves, regression coefficients and hazard ratios, for a wide variety of survival data.

Since the hazard rate must be positive a common function to link the covariates with the parameters is the exponential

$$c(\boldsymbol{\beta}^t\mathbf{Z}) = \exp(\boldsymbol{\beta}^t\mathbf{Z}) = \exp\left[\sum_{k=1}^{p} \beta_k Z_k\right], \tag{2.13}$$

and the hazard rate $h(t|\mathbf{Z})$ is defined as

$$h(t|\mathbf{Z}) = h_0(t)\exp(\boldsymbol{\beta}^t\mathbf{Z}) = h_0(t)\exp\left[\sum_{k=1}^{p} \beta_k Z_k\right]. \tag{2.14}$$

If we take two subjects with covariate values $\mathbf{Z}$ and $\mathbf{Z}^*$, the ratio of their hazard rates is

$$\frac{h(t|\mathbf{Z})}{h(t|\mathbf{Z}^*)} = \frac{h_0(t)\exp\left[\sum_{k=1}^{p} \beta_k Z_k\right]}{h_0(t)\exp\left[\sum_{k=1}^{p} \beta_k Z_k^*\right]} = \exp\left[\sum_{k=1}^{p} \beta_k (Z_k - Z_k^*)\right], \tag{2.15}$$

which is constant. This hazard ratio describes the relative risk of a subject with risk factor $\mathbf{Z}$, to experience the event of interest compared to a subject with risk factor $\mathbf{Z}^*$ adjusting for the other covariates. If for example covariate $\mathbf{Z}_1$ describes a treatment effect one for treatment A and zero for treatment B, then the relative risk of treatment A versus treatment B adjusting for other covariates is, $h(t|\mathbf{Z})/h(t|\mathbf{Z}^*) = \exp(\beta_1)$.

2.4.2 Cox model with time-dependent covariates

Often in survival studies there are explanatory variables whose values change during the duration of the study. These variables can be either continuous or categorical and are called time-dependent covariates. The Cox proportional hazard model that was introduced in the previous section can be extended to include time-dependent covariates.

As before if we have a sample of survival data of size n, we can describe each subject with a triple $(T_j, \delta_j, [\mathbf{Z}_j(t), 0 \leq t \leq T_j])$, with $j = 1, 2, ..., n$, T_j is the event time for the j subject,

δ_j is the event indicator for the j subject and $\mathbf{Z}_j(t) = (Z_{j1}(t), Z_{j2}(t), ..., Z_{jp}(t)$ is the vector of p covariates for each subject j. The $\mathbf{Z}_j(t)$ can represent time-dependent covariates or time-independent covariates and we assume that their values are known at each time point t. The proportional hazards models that also includes time-dependent covariates is defined as

$$h(t|\mathbf{Z}(t)) = h_0(t)\exp(\boldsymbol{\beta}^t\mathbf{Z}(t)) = h_0(t)\exp\left[\sum_{k=1}^{p}\beta_k Z_k(t)\right]. \tag{2.16}$$

Time-dependent variables can also used to test the proportional hazard assumption of the Cox model. In that case the Cox model can be extended to contain an interaction term between a time-independent covariate that violates the proportional hazard assumption and some function of time. For example if we want to assess the proportional hazards assumption for *Gender*, we can extend the Cox model to include the interaction term *Gender* x t. If the coefficient of the interaction term is significant, the proportional hazards assumption for *Gender* is violated.

2.5 Censoring Assumptions

All the methods for analyzing survival data that we described in this chapter are based on the assumption that censoring is *non-informative*, *independent* or *random.* Although the definition for those three types of censoring is similar it is not identical, but the important thing they have in common is that censoring observations do not relate to the event of interest. For this reason these terms are often mixed up, and the term *non-informative censoring* is mostly used for all three types of censoring. If the assumption does not hold the censoring is called *informative* or *dependent*, and not accounting for it can lead to biased results. For example if subjects that are censored are more likely to experience the event of interest that the ones that are not censored, then the estimated survival curve at any time point t will overestimate the true survival probability.

The three definitions for types of censoring based on the book of Kleinbaum and Klein (2012) are as follows.

Random Censoring

Subjects that are censored at time t are representative of all subjects that remain at risk at time t with respect to their survival experience. More simple this means that the risk of experiencing the event for subjects who are censored is equal to the subjects that remain at risk in the study.

Independent Censoring

Subjects within a subgroup who are censored at time t are representative of all the subjects in that subgroup who remain at risk at time t with respect to their survival experience. In other words, censoring is independent as long as it is random within each subgroup of interest.

Censoring can be independent and not random, for example if we study the survival of two groups A and B of subjects, censoring can be random within each group thus be independent, but if the survival probabilities are different in each group the censoring is not random.

Non-Informative Censoring

Let T be the distribution of the time to event random variable and, C the distribution of time to censoring random variable. When the distribution of T provides no information about the distribution of C, and vice versa, then we say the censoring is non-informative.

Situations where the condition of non-informative censoring is valid is when subjects are lost to follow-up for reasons independent of the study, for example moving, illness or changing of lifestyle unrelated to the study, etc.

Chapter 3

Simulation study with time-independent covariates

In this chapter we will evaluate the assumption of independent censoring in survival analysis. As we discussed briefly in the previous chapter, methods of estimating the survival probability typically assume that censoring is independent or non-informative. In practice however this assumption is often violated and, if it is not taken into account, it may introduce bias or even can nullify the results from certain statistical methods for survival data.

In order to evaluate how the presence of dependent censoring influences the result of standard methods, we will perform a series of Monte Carlo simulations. In this chapter we will focus only on time-independent covariates and in the next chapter we will add time-dependent covariates.

We will create an artificial population with two time-independent covariates *Age* and *Treatment*. We will generate events and censoring times by using an algorithm proposed by Bender [2], and in order to introduce dependent censoring both the events and censoring times distributions will be dependent on the time-independent covariates *Age* and *Treatment*. We will compare different scenarios for weak, moderate and strong association between the time-independent covariates and the censoring model and we will use two distribution for the event and censoring times, Weibull and Exponential to evaluate how the presence of dependent censoring effects the survival curves estimated by the Kaplan-Meier method. Finally we will introduce two metrics in order to quantify the bias that is produced by dependent censoring.

3.1 Simulation set-up

In this section we will give a detailed description of the artificial population, the method to generate events and censoring times, the way we will introduce dependent censoring and the diagnostics we will use in order to quantify the bias.

3.1.1 Introduction to Monte Carlo simulations

Since statistical methods are based on a sample from the population there is uncertainty in the estimate of the true parameter θ in the population the inference is based on. If we assume the estimate of the true parameter $\hat{\theta}$ comes for a probability distribution, Monte Carlo simulations can be used to estimate the sampling distribution of $\hat{\theta}$, by resampling from a given sample or by repeatedly sampling from an artificial population. For each sample an estimate of the true parameter $\hat{\theta}$, is calculated. Then the estimates $\hat{\theta}_1, \hat{\theta}_2, ..., \hat{\theta}_M$ of the M samples that have been drawn can be used to estimate the sampling distribution of $\hat{\theta}$. In the simulations of this chapter since we do not need to estimate model parameters but just the survival curves, $M = 50$ will be sufficient.

3.1.2 Define the artificial population

For the simulation study of this chapter we will define a sample of size of $n = 500$. For each subject in the sample we will generate two time-independent covariates *Age* and *Treatment*. *Age* will be generated from a normal distribution with mean 50 and standard deviation 10, and for generation of the survival and censoring times the Z-values for *Age* will be used. For the variable *Treatment* subjects will randomly receive treatment A (*Treatment*=1) or treatment B (*Treatment*=0) with equal probability 0.5.

3.1.3 Generation of survival and censoring times

For each individual i in the sample, we have to generate a pair of variables (t_i, δ_i), with $t_i = \min(x_i, c_i)$, where x_i is time to event and c_i is the censoring time, and $\delta_i = 1_{[t_i = x_i]}$. In order to obtain the pair of (t_i, δ_i), we have to generate event time x_i and censoring time c_i for each individual and take the minimum value.

The Exponential and Weibull distribution are two of the most commonly used distributions to describe survival data, both of them satisfy the assumption of proportional hazards , when they are used as baseline hazards in a Cox proportional hazards model. In this section we will show how to simulate a Cox proportional hazards model by generating survival times following the exponential distribution, based on the work of Bender [2].

The hazard function for the Cox model is defined as

$$h(t|\mathbf{Z}) = h_0(t)\exp(\boldsymbol{\beta}^t\mathbf{Z}), \tag{3.1}$$

where t is the survival time, $\mathbf{Z}$ is a vector with the model time-invariant covariates, $\boldsymbol{\beta}$ is a vector with the regression coefficients and $h_0(t)$ is the baseline hazard function. The survival function for the Cox proportional hazards model is defined as

$$S(t|\mathbf{Z}) = \exp[-H_0(t)\exp(\boldsymbol{\beta}^t\mathbf{Z})], \tag{3.2}$$

with

$$H_0(t) = \int_0^t h_0(u)du \tag{3.3}$$

the cumulative baseline hazard function. The cumulative distribution of the Cox model can be defined as

$$F(t|\mathbf{Z}) = 1 - \exp[-H_0(t)\exp(\boldsymbol{\beta}^t\mathbf{Z})]. \tag{3.4}$$

If Y is a random variable following a distribution with function F, $U = F(Y)$ will follow a uniform distribution in $[0, 1]$ based on the theorem of probability integral transform, also if U follows a uniform distribution then the variable $1 - U$ will follow a uniform distribution at $[0, 1]$. Let T be the survival time of the Cox proportional hazards model of (3.1), then from (3.3) we have

$$U = \exp[-H_0(T)\exp(\boldsymbol{\beta}^t\mathbf{Z})]. \tag{3.5}$$

H_0 can be inverted, if $h_0(t) > 0$ for all t, and the survival time of the Cox proportional hazard model can be simulated from

$$T = H_0^{-1}[-\log(U)\exp(-\boldsymbol{\beta}^t\mathbf{Z})]. \tag{3.6}$$

where $U \sim Uni[0, 1]$. Because random numbers following a uniform distribution can easily be simulated from statistical programs, equation (3.6) is suitable to generate survival and censoring times.

Exponential distribution

The exponential distribution with scale parameter $\lambda > 0$ and constant baseline hazard $h_0(t) = \lambda$, has a cumulative baseline hazard function $H_0(t) = \lambda t$. Since $h_0(t) > 0$ for all t, the cumulative baseline hazard function can be inverted, giving

$$H_0^{-1}(t) = \lambda^{-1} t. \tag{3.7}$$

Thus, if we insert (3.7) into (3.6), we have

$$T = \lambda^{-1}[-\log(U)\exp(-\boldsymbol{\beta}^t\mathbf{Z})] = -\frac{\log(U)}{\lambda\exp(\boldsymbol{\beta}^t\mathbf{Z})}. \tag{3.8}$$

So, the Cox proportional hazard model $h(t|\mathbf{Z}) = \lambda\exp(\boldsymbol{\beta}^t\mathbf{Z})$, with constant baseline hazard will give exponentially distributed survival times with scale parameter $\lambda(z) = \lambda\exp(\boldsymbol{\beta}^t\mathbf{Z})$, that it depends on the regression coefficients and the time-invariant covariates of the model.

Weibull distribution

Weibull is another frequently used distribution to describe survival data, but compared to the exponential distribution it does not have a constant hazard function, which is more realistic for real survival data. The Weibull distribution is described by two positive parameters, the scale parameter $\lambda > 0$ and shape $\nu > 0$. For values of $\nu > 1$ the hazard function increases and for $0 < \nu < 1$ the hazard function decreases, and for the special case of $\nu = 1$, the Weibull distribution is reduced to the exponential distribution with constant hazard function.

The cumulative baseline hazard function of the Weibull distribution is $H_0(t) = \lambda t^\nu$, with $h_0(t) > 0$ for all t, so the inverse of the cumulative baseline hazard is defined as

$$H_0^{-1}(t) = (\lambda^{-1} t)^{1/\nu}. \tag{3.9}$$

Again, by inserting (3.9) in (3.6), we get an expression for survival time T with the baseline hazard function of a Weibull distribution

$$T = [\lambda^{-1} - \log(U)\exp(-\boldsymbol{\beta}^t\mathbf{Z})]^{1/\nu} = \left[-\frac{\log(U)}{\lambda\exp(\boldsymbol{\beta}^t\mathbf{Z})}\right]^{1/\nu}. \tag{3.10}$$

From equation (3.10) it follows that the survival times will follow a Weibull distribution with scale parameter $\lambda(z) = \lambda\exp(\boldsymbol{\beta}^t\mathbf{Z})$, that varies based on the regression coefficients and the time covariates, and a fixed shape parameter ν.

3.1.4 Dependent censoring

In order to introduce dependent censoring in our sample, both the event distribution and the censoring distribution must be dependent on the covariates. Based on methodology of this section, we can simulate one set of times from equation (3.8) for the exponential distribution, or equation (3.10) for the Weibull distribution, one for the event times and one for the censoring times for each subject. Then we take the minimum of those two values to obtain the observed survival time and the corresponding event indicator, one if the event is observed and zero if the event is not observed.

Thus, in case of exponential distribution to generate survival times, we have $X \sim \exp(\lambda_x)$ and $C \sim \exp(\lambda_c)$ for the event times and the censoring times respectively. The hazard rates for the time to event and censoring time are defined as

$$h_X(t|\mathbf{Z}) = h_{0_X}(t)\exp(\boldsymbol{\beta}^t\mathbf{Z}), \tag{3.11}$$

$$h_C(t|\mathbf{Z}) = h_{0_C}(t)\exp(\boldsymbol{\phi}^t\mathbf{Z}). \tag{3.12}$$

with h_{0_X} and h_{0_C} the baseline hazards, and β and ϕ the regression coefficients for time to event and time to censoring respectively.

From the above equations of the hazards rates for time to event and censoring time we can see that the strength of the dependency between event times X and censoring times C can be adjusted by varying the regression coefficients β and ϕ. And we can vary the censoring percentage by adjusting the baseline hazards h_{0_X} and h_{0_C}.

The event time x_i, censoring time c_i, and the observed pair of variables (t_i, δ_i) for each individual i with covariates Age_i and $Treatment_i$ will be generated using the following algorithm.

Generate x_i, c_i pair of variables (t_i, δ_i) and covariates for subject i :

Step 1: Genarate the baseline coviaraites *Age* and *Treatment*, $Age \sim N(50, 10)$ and for Treatment, subjects will randomly receive Treatment A (Treatment=1) or Treatment B (Treatment=0) with equal probability 0.5. $\mathbf{Z}_i = c(Age_i, Treatment_i)$, the vector of covariates.

Step 2: Generate $U_{1_i}, U_{2_i} \sim U[0, 1]$.

Step 3: Generate x_i and c_i using equation (3.8)

$$x_i = -\frac{\log(U_1)}{\lambda\exp(\beta^t Z_i)},$$

$$c_i = -\frac{\log(U_2)}{\lambda\exp(\phi^t Z_i)}.$$

Step 4: From x_i and c_i obtain the pair (t_i, δ_i) :

$$t_i = \min(x_i, c_i),$$

and

$$\delta_i = \begin{cases} 1 & \text{if} \quad x_i \le c_i, \\ 0 & \text{if} \quad x_i > c_i \end{cases}.$$

With the same algorithm we will generate survival times using the Weibull distribution but instead of using equation (3.8) we will use equation (3.10) to generate event times x_i and censoring times c_i.

3.1.5 Estimation of survival curves

The goal of these simulations is to compare how the presence of dependent censoring affects the estimation of the survival curve. This will be done by comparing the true survival curve with the curve estimated by the standard [Kaplan-Meier] method. The Kaplan-Meier estimator will be calculated at k predefined time points $\tau_1, \tau_2, ..., \tau_k$.

True Survival Curve

The true survival curve can be estimated either parametrically or from the Monte Carlo simulations, here will choose the latter method, because it can represent better real life situations where only the event times for subjects are observed and the model parameters are unknown.

For every Monte Carlo simulation j, with $j = 1, ..., M$ the survival model is fitted based only on the event times x_i and not on the $t_i = \min(x_i, c_i)$, in this case all the subjects in the study experience the event of interest. Next we calculate the survival probabilities with the Kaplan-Meier estimator at the predefined time points $\tau_1, \tau_2, ..., \tau_k$, and we get the survival probability estimates $\hat{S}_j(\tau_1)$, $\hat{S}_j(\tau_2)$, ..., $\hat{S}_j(\tau_k)$ for each Monte Carlo simulation j.

Survival Curve from censored data using the Kaplan-Meier Estimator

For each Monte Carlo simulation j, we estimate the survival curve based on the pair of variables (t_i, δ_i), and the survival probabilities are calculated again with the Kaplan-Meier estimator for the predefined time points $\tau_1, \tau_2, ..., \tau_k$. The survival probabilities are denoted as $\tilde{S}(\tau_k)$.

3.1.6 Diagnostics

If $\hat{S}(\tau_k)$ and $\tilde{S}(\tau_k)$ are the estimates of the survival probabilities estimated with the true (uncensored) survival data and the standard Kaplan-Meier method applied to the censored data respectively. For M Monte Carlo simulations the mean survival probability for each method at each time point τ_k is defined as

$$\bar{\hat{S}}(\tau_k) = \sum_{j=1}^{M} \frac{\hat{S}_j(\tau_k)}{M},$$
$$\bar{\tilde{S}}(\tau_k) = \sum_{j=1}^{M} \frac{\tilde{S}_j(\tau_k)}{M},$$

with $k = 1, ..., p$. With those means we can plot the survival curves estimated by each method for the predefined time points τ_k and compare them. We will introduce two methods in order to quantify the difference between the two survival curves.

Area between curves

The area of a region that is bound on $a \leq x \leq b$ by two continuous functions y_1, y_2 with $y_2(x) \geq y_1(x)$ is given by, $A = \int_a^b y_2(x) - y_1(x)\, dx$. For each simulation j will calculate the area

$$A_j = \int_a^b \tilde{S}_j(\tau_k) - \hat{S}_j(\tau_k)\, d\tau_k, \tag{3.13}$$

between the standard Kaplan-Meier curve and the true survival curve in the interval [0,b], where b is the last observed event time. Next we divide the area A by the last observed event time of the simulation j, in order to get the mean, and then average over the M Monte Carlo simulations. In our case we will calculate the area between the curves using the function area.between.curves from package geiger, which use the trapezoid rule to calculate the area.

In some of the simulations, particularly when the strength of the censoring mechanism is small, the two curves cross at some time points resulting in the area between curves being smaller since it subtracts the area that is below the true survival curve $\hat{S}(\tau_k)$ from the area that is above $\hat{S}(\tau_k)$. Since that issue occurs only in some simulations when the censoring mechanism is low and we mainly are interested in the average overestimation of the standard Kaplan-Meier curve $\tilde{S}(\tau_k)$, we opt to use equation (3.13) to calculate the area than the alternative,

$$A = \int_a^b |\tilde{S}(\tau_k) - \hat{S}(\tau_k)|\, d\tau_k, \tag{3.14}$$

which will calculate the sum of the area that is above and below the true survival curve $\hat{S}(\tau_k)$.

Maximum distance between curves

The maximum vertical distance between two continuous functions y_1, y_2 with $y_2(x) \geq y_1(x)$ in a region $[a, b]$ is defined as, $D = \max[y_2(x) - y_1(x)]$. For each simulation j the maximum distance between the curves will be calculated as,

$$D_j = \max[\tilde{S}_j(\tau_k) - \hat{S}_j(\tau_k)], \tag{3.15}$$

between the standard Kaplan-Meier curve and the true survival curve, where we take the maximum over all k's, i.e. over all specified time points. Then we average over the M Monte Carlo simulations to find the average maximum distance between the two curves.

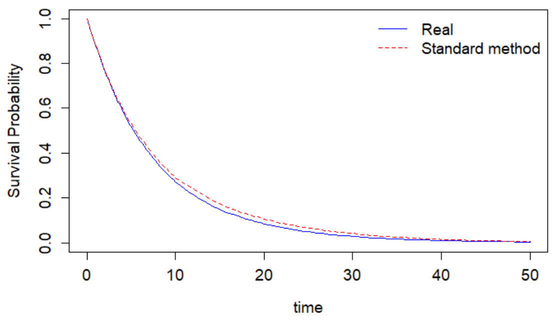
3.2 Simulation results for Exponentially distributed event and censoring times

In order to evaluate the influence of dependent censoring on survival curve estimation we will consider different scenarios, where we will vary the censoring percentage, the strength of the censoring and the number of subjects. For this part an exponential distribution will be assumed for the event and censoring times, and those times will be generated from equation (3.8) with $\lambda_x(t|\mathbf{Z}) = \lambda_{0_x} \exp(\boldsymbol{\beta}^t\mathbf{Z})$ for the event times, and $\lambda_c(t|\mathbf{Z}) = \lambda_{0_c} \exp(\boldsymbol{\phi}^t\mathbf{Z})$ for the censoring times, where $\mathbf{Z}_i = (Age_i, Treatment_i)$. For every simulation, time to event will be simulated with the same coefficients $\boldsymbol{\beta} = (0.5, 0.1)$ and baseline hazard $\lambda_{0_x} = 0.1$.

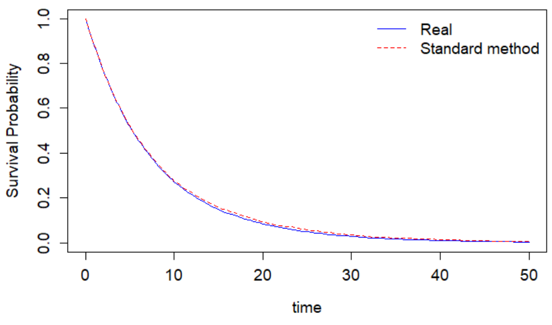
3.2.1 Strength of the dependency of the censoring mechanism on the time-invariant covariates

The strength of the dependency can be varied by adjusting the coefficients $\phi = (\phi_1, \phi_2)$. As the values of $\boldsymbol{\phi}$ get bigger the censoring is more dependent on the covariates $\mathbf{Z} = (Age, Treatment)$. For this simulation the sample size will be $n = 500$, the $\boldsymbol{\beta}$ coefficients and the baseline hazard λ_{0_x} for the time to event will be the same. The coefficients $\boldsymbol{\phi}$ will vary in order to simulate different levels of dependency of the censoring times on the covariates $\mathbf{Z}$. The baseline hazard λ_{0_c} will be adjusted in order to keep the censoring percentage at 33%.

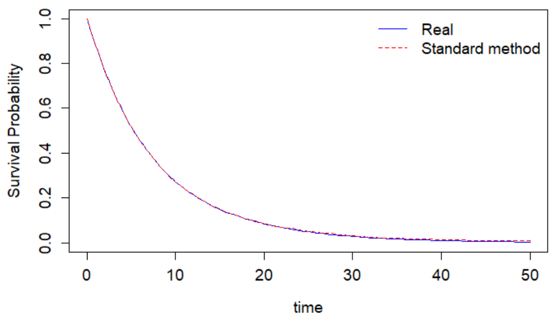
In Figure 3.1 we see the results for three different pairs of values for the $\boldsymbol{\phi}$ coefficients. As the $\boldsymbol{\phi}$ coefficients get smaller in absolute value the covariates have smaller effect on the hazard. Thus the probability of getting censored is correlated with the probability of experiencing the event, correlation between the generated event and censoring times is 0.119 for $(\phi_1, \phi_2) = (1.5, 0.5)$ and drops to 0.011 for $(\phi_1, \phi_2) = (0.1, 0)$. This leads to overestimation of the survival probabilities using the standard Kaplan-Meier since less event are observed. From the plots in figure 3.1 we see as the dependency gets weaker the fit of the Kaplan-Meier method gets better.



(a) $(\phi_1, \phi_2) = (1.5, 0.5), \lambda_{0_c} = 0.027$



(b) $(\phi_1, \phi_2) = (0.5, 0.1), \lambda_{0_c} = 0.05$



(c) $(\phi_1, \phi_2) = (0.1, 0), \lambda_{0_c} = 0.06$

Figure 3.1: *The real survival curve (blue) based on uncensored data and the survival curve estimated with the Kaplan-Meier method on censored data (red) for different values of $\boldsymbol{\phi}$ coefficients. These are the average curves over 50 simulations with $(\beta_1, \beta_2) = (0.5, 0.1)$ and $\lambda_{0_x} = 0.1$.*

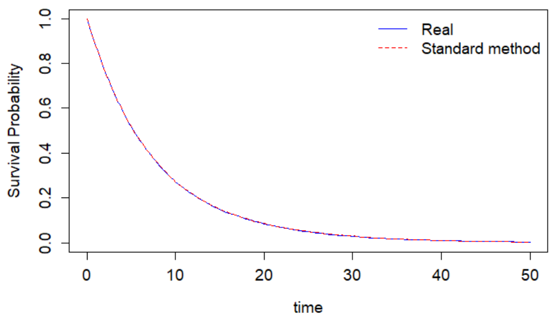
Table 3.1 shows the parameters that were used for this simulation. Besides the visual comparison of the two curves we introduced two functions to measure the dissimilarities between the two curves. In the Table 3.1 we see the results of the average area between the two curves for three different cases of coefficients $\boldsymbol{\phi}$, divided by time period of follow up. The max distance shows the maximum distance the curves have. The average area from Table 3.1 gets smaller as the strength of the dependency gets weaker. The max distance values as we see from Table 3.1 is not very reliable method by itself to evaluate the dissimilarities between the curves, although the have an decreasing trend.

	n	h_{0_x}	$\exp(\beta_1)$	$\exp(\beta_2)$	h_{0_c}	$\exp(\phi_1)$	$\exp(\phi_2)$	Average area	Max distance
(ϕ_1, ϕ_2)=(1.5,0.5)	500	0.1	1.65	1.10	0.027	4.50	1.65	0.013	0.038
(ϕ_1, ϕ_2)=(0.5,0.1)	500	0.1	1.65	1.10	0.05	1.65	1.10	0.007	0.031
(ϕ_1, ϕ_2)=(0.1,0)	500	0.1	1.65	1.10	0.06	1.10	1.00	0.002	0.028

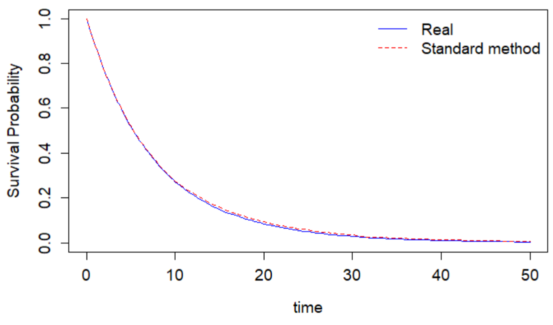
Table 3.1: Simulation parameters for event and censoring models, with $(\beta_1, \beta_2) = (0.5, 0.1)$,scale $\lambda_{0_x} = 0.01$, different values for $\boldsymbol{\phi}$ coefficients and λ_{0_c} respectively, and 33% censoring.

3.2.2 Percentage of censored subjects

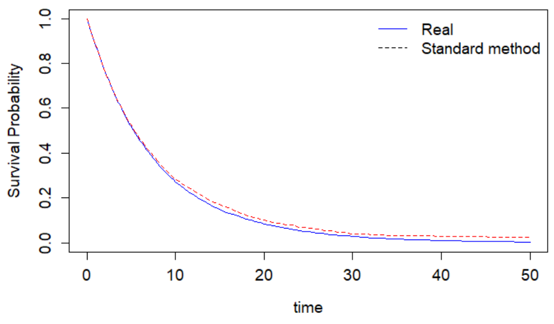
Next we will vary the percentage of censored subjects by adjusting the parameter λ_{0_c}. The ϕ parameters of the censoring model will be equal to (0.5,0.1) and for the events time model parameters β will again be equal to (0.5,0.1) and $\lambda_{0_x} = 0.1$.



(a) 10%, $\lambda_{0_c} = 0.012$



(b) 30%, $\lambda_{0_c} = 0.045$



(c) 50%, $\lambda_{0_c} = 0.1$

Figure 3.2: *The real survive curve (blue) based on uncensored data and the survival curve estimated with the Kaplan-Meier method on censored data (red) for different percentage of censored subjects. These are the average curves over 50 simulations with* $(\beta_1, \beta_2) = (0.5, 0.1)$ *and* $\lambda_{0_x} = 0.1$*, for the events model and* $(\phi_1, \phi_2) = (0.5, 0.1)$*, and different values* λ_{0_c} *for the censoring model.*

Results in Figure 3.2 show that the overestimation of the survival curves using the standard Kaplan-Meier method gets worse as the censoring percentage is increasing. This trend is more clear from Table 3.2 with the increase of Average area and Max distance as the censoring percentage is increasing.

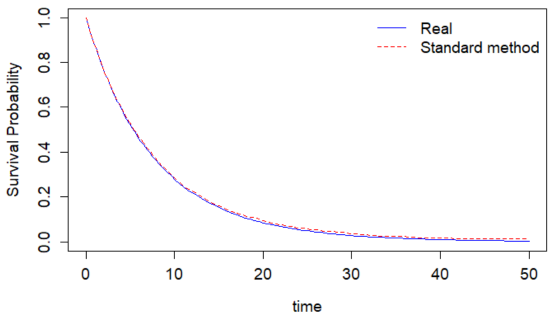
	n	h_{0_x}	$\exp(\beta_1)$	$\exp(\beta_2)$	h_{0_c}	$\exp(\phi_1)$	$\exp(\phi_2)$	Average area	Max distance
10%	500	0.1	1.65	1.10	0.012	1.65	1.10	0.0007	0.012
30%	500	0.1	1.65	1.10	0.045	1.65	1.10	0.006	0.028
50%	500	0.1	1.65	1.10	0.1	1.65	1.10	0.015	0.058

Table 3.2: Simulation parameters for event and censoring models, with $(\beta_1, \beta_2) = (0.5, 0.1)$, and $\lambda_{0_x} = 0.1$ for the event times model, and $(\phi_1, \phi_2) = (0.5, 0.1)$, for the censoring model.

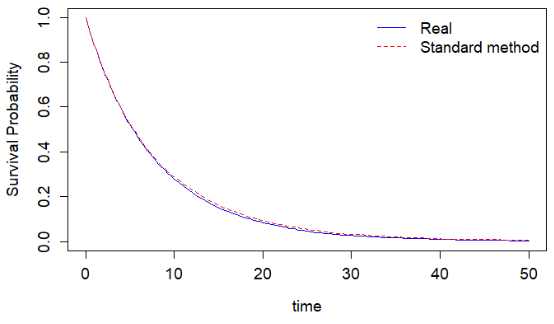
From Table 3.2 we can see clearly by looking the values of the Average area and Max distance the overestimation of the survival curve calculated by the standard method is increasing as the censoring percentage is increasing.

3.2.3 Size of sample

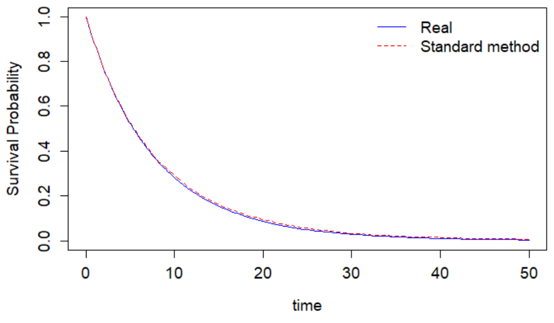
Finally we will vary the sample size while keeping the censoring percentage at 33%. The $\boldsymbol{\phi}$ parameters of the censoring model will be equal to $(0.5, 0.1)$ with $\lambda_{0_c} = 0.05$ and for events time model parameters $\boldsymbol{\beta}$ will be equal to $(0.5, 0.1)$ with $\lambda_{0_x} = 0.1$.



(a) $n = 250$



(b) $n = 500$



(c) $n = 1000$

Figure 3.3: *The real survival curve (blue) based on uncensored data and the survival curve estimated with the Kaplan-Meier method on censored data (red) for different sample sizes. These are the average curves over 50 simulations with* $(\beta_1, \beta_2) = (0.5, 0.1)$, $\lambda_{0_x} = 0.1$ *and* $(\phi_1, \phi_2) = (0.5, 0.1)$, $\lambda_{0_c} = 0.05$, *for the event and censoring model respectively.*

From Figure 3.3, results for the simulation shown that as the sample size increases the curve of the standard method improves slightly between different sample sizes.

	n	h_{0_x}	$\exp(\beta_1)$	$\exp(\beta_2)$	h_{0_c}	$\exp(\phi_1)$	$\exp(\phi_2)$	Average area	Max distance
(ϕ_1,ϕ_2)=(0.5,0.1)	250	0.1	1.65	1.10	0.05	1.65	1.10	0.008	0.046
(ϕ_1,ϕ_2)=(0.5,0.1)	500	0.1	1.65	1.10	0.05	1.65	1.10	0.007	0.032
(ϕ_1,ϕ_2)=(0.5,0.1)	1000	0.1	1.65	1.10	0.05	1.65	1.10	0.006	0.023

Table 3.3: Simulation parameters for event and censoring models, with $(\beta_1, \beta_2) = (0.5, 0.1)$, and $\lambda_{0_x} = 0.1$ for the event times model, and $(\phi_1, \phi_2) = (0.5, 0.1)$, $\lambda_{0_c} = 0.05$ for the censoring model, for different sample sizes and 33% censoring.

Table 3.3 shows that there is a very small improvement when the sample increases from 250 to 500, and 1000 and it is only evident by looking at the values of the Average area and Max distance.

3.3 Simulation results for Weibull distributed event and censoring times

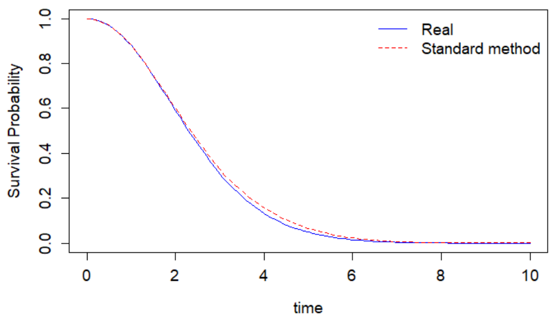
For this part a Weibull distribution will be assumed for the event and censoring times, and the times will be generated from equation (3.10) with $\lambda_x(t|\mathbf{Z}) = \lambda_{0_x}(t)\exp(\boldsymbol{\beta}^t\mathbf{Z})$ for the event times and $\lambda_c(t|\mathbf{Z}) = \lambda_{0_c}(t)\exp(\boldsymbol{\phi}^t\mathbf{Z})$ for the censoring times, where $\mathbf{Z}_i = (Age_i, Treatment_i)$. For every simulation, time to event will be simulated with the same coefficients $\beta = (0.5, 0.1)$

and scale parameter $\lambda_{0_x} = 0.1$. The shape parameter of the Weibull distribution will be equal to $\nu = 2$ for simulations for both the censoring times and events times.

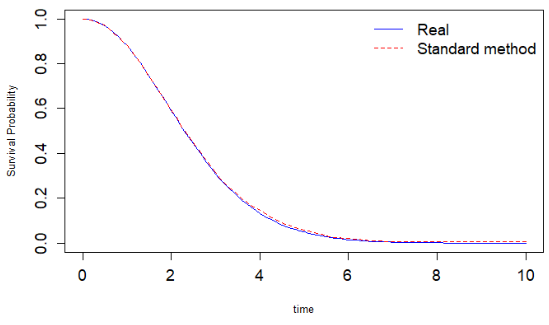
3.3.1 Strength of the dependency of the censoring mechanism on the time-invariant covariates

As the previous case of exponential distribution, first we will evaluate the strength of the censoring mechanism by adjusting the coefficients $\boldsymbol{\phi}$ of the censoring model for strong, medium and weak association, while keeping the coefficients $\boldsymbol{\beta}$ of the event times the same. The sample size for the simulations will be $n = 500$, the shape parameter of the Weibull distribution for both the censoring and event times will be $\nu = 2$, and the shape parameter λ_{0_c} of the censoring model will adjusted in order to keep the censoring percentage at 33%.

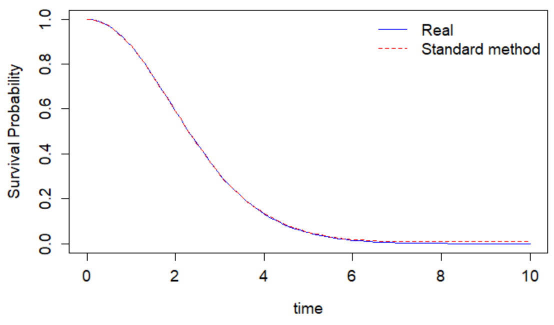
Figure 3.4 shows the results for the three different pairs of values for the $\boldsymbol{\phi}$ coefficients. As the $\boldsymbol{\phi}$ coefficients of the time-invariant covariates get bigger the covariates have bigger effect on the hazard. Thus the probability of getting censored is correlated with the probability of experiencing the event, correlation between the generated event and censoring times is 0.135 for $(\phi_1, \phi_2) = (1.5, 0.5)$ and drops to 0.007 for $(\phi_1, \phi_2) = (0.1, 0)$. This leads to overestimation of the survival probabilities using the standard Kaplan-Meier estimator since fewer events are observed specially at later points where the observations are censored. From the plots in Figure 3.4 we see as the dependency gets weaker the fit of the Kaplan-Meier method gets better, but still as we see from Figure 3.4(b) even for a moderate value of the $\phi = (0.5, 0.1)$ parameters there is a small difference between the two curves.



(a) $(\phi_1, \phi_2) = (1.5, 0.5), \lambda_{0_c} = 0.027$



(b) $(\phi_1, \phi_2) = (0.5, 0.1), \lambda_{0_c} = 0.05$



(c) $(\phi_1, \phi_2) = (0.1, 0), \lambda_{0_c} = 0.06$

Figure 3.4: *The real survival curve (blue) based on uncensored data and the survival curve estimated with the Kaplan-Meier method on censored data (red) for different values of ϕ coefficients. These are the average curves over 50 simulations with $(\beta_1, \beta_2) = (0.5, 0.1)$, $\lambda_{0_x} = 0.1$ and $\nu_x = 2$.*

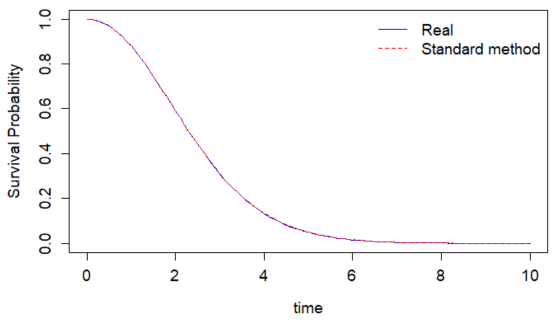
	n	λ_{0_x}	exp(β_1)	exp(β_2)	λ_{0_c}	exp(ϕ_1)	exp(ϕ_2)	Average area	Max distance
(ϕ_1, ϕ_2)=(1.5,0.5)	500	0.1	1.65	1.10	0.027	4.50	1.65	0.012	0.038
(ϕ_1, ϕ_2)=(0.5,0.1)	500	0.1	1.65	1.10	0.05	1.65	1.10	0.005	0.030
(ϕ_1, ϕ_2)=(0.1,0)	500	0.1	1.65	1.10	0.06	1.10	1.00	0.001	0.027

Table 3.4: Simulation parameters for event and censoring models, with $(\beta_1, \beta_2) = (0.5, 0.1)$, scale $\lambda_x = 0.1$, shape $\nu_x = 2$, for the event times model, shape $\nu_c = 2$ and different values for $\boldsymbol{\phi}$ coefficients and λ_{0_c}, for the censoring model, and 33% censoring.

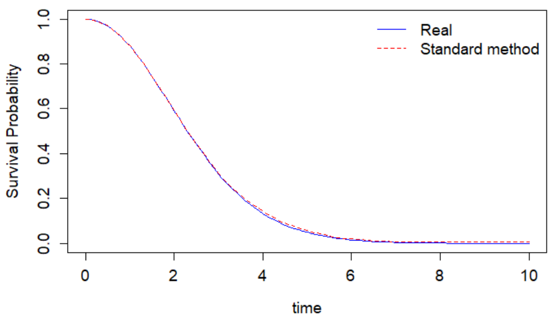
Table 3.4 shows the parameters that were used for this simulation and the results for the Average area and Max distance. We observe a similar trend as in the case of the exponentially distributed event and censoring times, with Average area and Max distance decreasing as the strength of the dependency gets weaker. The values for the Average area and Max distance are close to the values we observed in the case of the exponential distributed event and censoring times.

3.3.2 Percentage of censored subjects

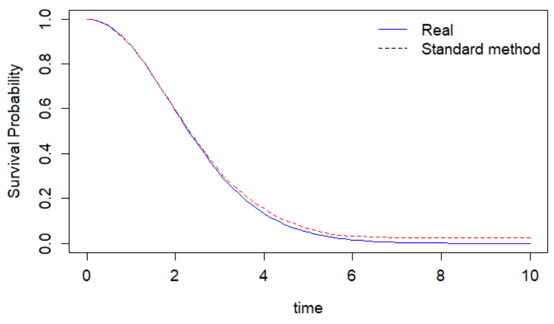
By adjusting the scale parameter of the censoring model λ_{0_c}, we can change the percentage of censored subjects. For the censoring model simulations will be performed with parameters $\boldsymbol{\phi}$ equal to (0.5,0.1). For the event times model parameters $\boldsymbol{\beta}$ will be equal to (0.5,0.1), scale $\lambda_{0_x} = 0.1$ and for the both models the shape parameter of the Weibull distribution will be $\nu_x = \nu_c = 2$.



(a) 10%, $\lambda_{0_c} = 0.012$



(b) 30%, $\lambda_{0_c} = 0.045$



(c) 50%, $\lambda_{0_c} = 0.1$

Figure 3.5: *The real survive curve (blue) based on uncensored data and the survival curve estimated with the Kaplan-Meier method on censored data (red) for different percentage of censored subjects. These are the average curves over 50 simulations with* $(\beta_1, \beta_2) = (0.5, 0.1)$, $\lambda_{0_x} = 0.1$ *and shape* $\nu_x = 2$, *for the events model and* $(\phi_1, \phi_2) = (0.5, 0.1)$, *shape* $\nu_c = 2$ *and different values* λ_{0_c} *for the censoring model.*

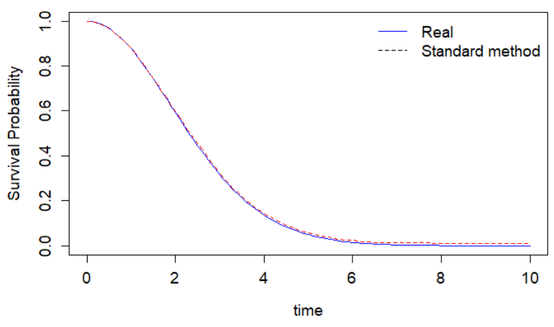
From Figure 3.5 we see that for 10% censoring the two curves has hardly any difference, for 30% there is some small difference between the two curves, and for 50% the difference is more clear, specially at the later time points ($t > 4$), where more observations are censored.

	n	λ_{0_x}	$\exp(\beta_1)$	$\exp(\beta_2)$	λ_{0_c}	$\exp(\phi_1)$	$\exp(\phi_2)$	Average area	Max distance
10%	500	0.1	1.65	1.10	0.012	1.65	1.10	0.0007	0.011
30%	500	0.1	1.65	1.10	0.045	1.65	1.10	0.004	0.028
50%	500	0.1	1.65	1.10	0.1	1.65	1.10	0.01	0.057

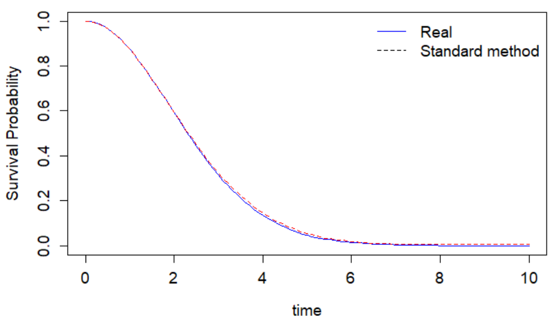
Table 3.5: Simulation parameters for event and censoring models, with $(\beta_1, \beta_2) = (0.5, 0.1)$, $\lambda_{0_x} = 0.1$ and shape $\nu_x = 2$ for the event times model, and $(\phi_1, \phi_2) = (0.5, 0.1)$, shape $\nu_c = 2$ for the censoring model.

3.3.3 Size of sample

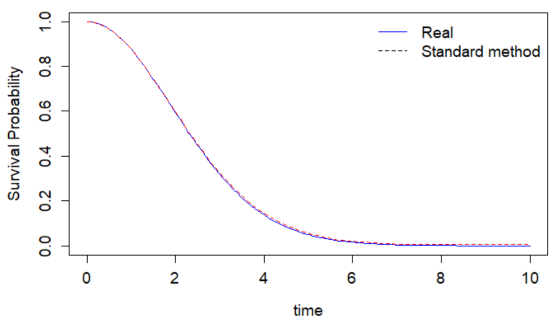
Finally we will vary the sample size while keeping the censoring percentage at 33%. The $\boldsymbol{\phi}$ parameters of the censoring model will be equal to $(0.5, 0.1)$ with $\lambda_{0_c} = 0.05$ and for events time model parameters β will be equal to $(0.5, 0.1)$ with $\lambda_{0_x} = 0.1$. The shape parameter for both models will be $\nu_x = \nu_c = 2$.



(a) $n = 250$



(b) $n = 500$



(c) $n = 1000$

Figure 3.6: *The real survival curve (blue) based on uncensored data and the survival curve estimated with the Kaplan-Meier method on censored data (red) for different sample sizes. These are the average curves over 50 simulations with* $(\beta_1, \beta_2) = (0.5, 0.1)$, $\lambda_{0_x} = 0.1$, $\nu_x = 2$ *and* $(\phi_1, \phi_2) = (0.5, 0.1)$, $\lambda_{0_c} = 0.05$, $\nu_c = 2$ *for the event and censoring model respectively.*

Again from Figure 3.6 and Table 3.6, we notice that increasing the sample size has a very small impact on the prediction of the survival curve with the standard method, the performance increases a little bit as we see from the average and the max distance but the differences are extremely small.

	n	λ_{0_x}	$\exp(\beta_1)$	$\exp(\beta_2)$	λ_{0_c}	$\exp(\phi_1)$	$\exp(\phi_2)$	Average area	Max distance
(ϕ_1,ϕ_2)=(0.5,0.1)	250	0.1	1.65	1.10	0.05	1.65	1.10	0.006	0.045
(ϕ_1,ϕ_2)=(0.5,0.1)	500	0.1	1.65	1.10	0.05	1.65	1.10	0.006	0.031
(ϕ_1,ϕ_2)=(0.5,0.1)	1000	0.1	1.65	1.10	0.05	1.65	1.10	0.005	0.022

Table 3.6: Simulation parameters for event and censoring models, with $(\beta_1, \beta_2) = (0.5, 0.1)$, $\lambda_{0_x} = 0.1$ and shape $\nu_x = 2$ for the event times model, and $(\phi_1, \phi_2) = (0.5, 0.1)$, $\lambda_{0_c} = 0.05$ and shape $\nu_c = 2$ for the censoring model, for different sample sizes.

3.4 Conclusions

The simulations that were carried out in this chapter with different scenarios for the strength of censoring mechanism, the censoring percentage, and the sample size show how the standard method to calculate the survival curve performed in the presence of dependent censoring. Results showed that although there is a difference between the true survival curve and the curve calculated from the standard method, these differences seem significant only in the case when censoring was around 50% and when the coefficients ϕ that were associated with censoring had values $(\phi_1, \phi_2) = (1.5, 0.5)$. For a moderate value of the coefficients ϕ and with a censoring percentage around 33%, which are closer to real life scenarios, the differences between the two curves are small and within the confidence intervals of a single survival curve for a chosen sample size as we can see from the graphs on the Appendix A.1 . From the simulations in this chapter we can conclude that dependent censoring does not seem as a significant problem as long as the values of the coefficients of the time-invariant covariates that can influence dependent censoring are not unusual high. In the next chapter we will explore if this conclusion is different if we add time-dependent covariates to the dataset.

Chapter 4

Simulation study with time-dependent Covariate

On the previous chapter we used the method introduced by Bender [2] to generate event and censoring times by using the Cox proportional model with exponential and Weibull distributions for the baseline survival. However, generating event and censoring times that depend on time-dependent covariates is more challenging. The reason is that an individual's outcome corresponds to multiple values of a covariate over time, and where the number of the observations of the covariate may vary across subjects, for example if the time-dependent covariate is a biomarker. Using the method introduced by Bender [2] in this case is only possible in special cases because it will require to invert the expression $H_0(t)\exp(\boldsymbol{\beta}^t\mathbf{Z}(\mathbf{t}))$. This inversion cannot be done easily and it is only possible if the changes over time for the time-dependent covariate can be described by a parametric function and are defined over the range of study time t, as shown by Sylvestre and Abrahamowicz [9].

Several methods have been proposed to generate event and censoring times with time-dependent covariates. Austin [1] extended the work of Bender [2] and presents closed form expressions to generate survival times that follow exponential, Weibull or Gompertz distributions for the baseline survival, with a vector of time-invariant covariates and one time-dependent covariate. However as stated in the previous paragraph this only works if changes over time of the time-dependent covariate can be described by a parametric function and are defined over the range of study time t.

Methods that do not require inverting the cumulative hazard functions have been proposed. Sylvestre and Abrahamowicz [9] proposed an algorithm that generates event times conditional on time-dependent covariates, which matches survival times with an independently generated vectors of covariates, based on a probability law derived from a partial likelihood of the Cox proportional hazards model. There is no closed form for the generated survival times but this method allows for the use of any number of time-dependent and time-invariant covariates. There is no need to specify a parametric form for the time-dependent covariates or how they vary over time, although the accuracy of the algorithm decreases with increasing rates of censoring. Hendry [3] expanding the work of Zhou [13] proposed an algorithm to generate event times with any number of time-invariant and time-dependent covariates, that relies on a transformation of a random variable according to a truncated piecewise exponential distribution. The bounds of the truncation allow flexibility in the number of measurements that are of interest in a given scenario, and the piecewise exponential distribution allows the covariates to vary as step functions over the time scale.

Finally Ngwa [6] proposed a method that builds on the work of Bender [2] and Austin [1] for generating event and censoring times by deriving a closed form expression that links a set of time-invariant covariates and one time-dependent covariate to the event and censoring times when the baseline survival follows an Exponential or a Weibull distribution using the Lambert

W Function. This method does not require the time-dependent covariate to be described by a parametric function as is required for the method introduced by Austin [1], thus allows us to have a more realistic scenario for the time-dependent variable. Also the method offers a closed form expression for the survival times and because of that is faster compared to the methods that do not offer a closed form, which can be computationally heavy. For the above reasons and because our goal is to evaluate the presence of dependent censoring, we will choose the method proposed by Ngwa [6], to generate event and censoring times. This method is based on the Two Step Approach by Tsiatis [10], in which a linear mixed effects model is used to describe the longitudinal measurements of the biomarker and then on the second stage the fitted values of the biomarker are inserted in the Cox proportional model as time-dependent covariates.

In this chapter, we will create an artificial population with two time-independent covariates *Age* and *Treatment*, and one time-dependent covariate with measurements taken on 6 different time points. The method proposed by Ngwa [6] to generate event and censoring times that depend on two time-independent covariates and one time-dependent covariate will be introduced. Finally we will compare different scenarios where we will vary the coefficients of the time-independent covariates while holding the value of the coefficient that links the time-dependent covariate with the survival model constant, scenarios that we vary the coefficient that links the time-dependent covariate while the coefficients of the time-independent covariates have the same value, and lastly scenarios with different censoring percentage.

4.1 Simulation set-up

4.1.1 Define the artificial population

For the simulation study of this chapter we will use the same sample size $n = 500$ and the same time-invariant covariates that we used in the previous chapter. For each subject in the sample we will generate two time-invariant covariates *Age* and *Treatment*. *Age* will be generated from a normal distribution with mean 50 and standard deviation 10, and for generation of the survival and censoring times the standardized Z-values for *Age* will be used. For the variable *Treatment* subjects will randomly receive treatment A (*Treatment*=1) or treatment B (*Treatment*=0) with equal probability 0.5. For the time-dependent covariate we will use a linear mixed effects model to model and generate the longitudinal measurements.

4.1.2 Generation of survival and censoring times

In this section we will show how to simulate a Cox proportional hazards model by generating survival times with baseline survival following the Weibull distribution[1], based on the work of Ngwa (2019). Before we derive the closed form expression to generate event and censoring times we have to define the linear mixed model that will be used to describe the longitudinal measurements of the time-dependent covariate.

Linear mixed model

For the time-dependent covariate we will assume a biomarker that we measure at six different time points t. Let $y_i(t)$ be the value of the biomarker at the time point t for the individual i. The value of the $y_i(t)$ is observed only at the time points at which the measurements are taken and not at any given time point. Thus the observed values of the biomarker for each individual at each time point j are $y_{ij} = [y_i(t_{ij}), j = 1, 2, .., 6]$. Also the values of $y_i(t)$ may be measured with error. In order to associate the true, unobserved value of the biomarker at time

[1]In this chapter we will present results only for the Weibull distribution, because the simulations with a time-dependent covariate are very time consuming and Weibull disribution is more close to real life scenarios.

t denoted by $y_i^*(t)$ with the event and censoring times we model the longitudinal values with a linear mixed model. So we have

$$\begin{aligned} y_i(t) &= y_i^*(t) + \varepsilon_i(t) \\ &= \mathbf{X}_i^T(t)\boldsymbol{\beta} + \mathbf{Z}_i^T(t)\boldsymbol{b}_i + \boldsymbol{\varepsilon}_i(t), \quad \boldsymbol{\varepsilon}_i(t) \sim \mathcal{N}(0, \sigma^2), \end{aligned} \tag{4.1}$$

where $\mathbf{X}_i^T(t)$ and $\mathbf{Z}_i^T(t)$ are the design matrices for the fixed and random effects, $\boldsymbol{\beta}$ is the vector of the fixed effects parameters, $\boldsymbol{b}_i$ is the vector of the random effects, and $\boldsymbol{\varepsilon}_i(t)$ is the vector of measurement errors, which is assumed independent and with variance σ^2.

For our simulation for the biomarker measurements we will use a simple linear growth model[2]. A linear growth model is defined as,

$$\begin{aligned} y_i(t) &= \xi_0 + \xi_1 t + b_{0i} + b_{1i} t + \varepsilon_i \\ &= (\xi_0 + b_{0i}) + (\xi_1 + b_{1i})t + \varepsilon_i \\ &= \alpha_{i0} + \alpha_{i1} t + \varepsilon_i \\ &= y_i^*(t) + \varepsilon_i, \end{aligned} \tag{4.2}$$

where $y_i(t)$ is the value of the biomarker at time point t, ξ_0 is the mean intercept, ξ_1 is the mean growth rate of the biomarker, b_{0i} and b_{1i} are the random intercept and slope respectively, that describe the individual deviation from the sample mean intercept and mean growth rate, and ε_i is the individual's deviation from the true value of the biomarker. The random intercept b_{0i}, random slope b_{1i} together with the mean intercept ξ_0 and mean slope ξ_1, describe the individuals true value of the biomarker $y_i^*(t)$.

Two step approach

The Cox proportional model including the time-dependent covariate is defined as

$$h(t|\mathbf{Z}) = h_0(t)\exp(\boldsymbol{\beta}^t\mathbf{Z} + \gamma y_i^*(t)), \tag{4.3}$$

where $h_0(t)$ is the baseline hazard function, $\mathbf{Z}$ is the vector with the model's time-invariant covariates, $\boldsymbol{\beta}$ the vector for the parameters for the time-invariant covariates, $y_i^*(t)$ are the true values for the biomarker, and γ is a parameter that links the time-dependent covariate to the Cox model. The Cox proportional hazards model of (4.3) and the linear growth model (4.2) are linked through the shared random effects b_{0i} and b_{1i}. To account for the dependency between the two models we will use the two step approach. In the first step we will fit the linear growth model to the longitudinal values of the biomarker, and estimate the values of the biomarker based on the fitted linear growth model. In the second step we will fit in the Cox proportional hazards model the estimates for the time-dependent covariate from the previous step in the place of the true, unobserved values of the biomarker.

Closed form expression to simulate event and censoring times

For each individual i in the sample, we have to generate a pair of variables (t_i, δ_i), with $t_i = \min(x_i, c_i)$, where x_i is time to event and c_i is the censoring time, and $\delta_i = 1_{[t_i = x_i]}$. In order to obtain the pair of (t_i, δ_i), we have to generate event time x_i and censoring time c_i for each individual and take the minimum value.

In this section we will show how to simulate a Cox proportional hazards model by generating survival times with baseline survival following the Weibull distribution with the help of Lambert W Function, based on the work of Ngwa (2019).

[2]The method we will use to generate the events times is not applicable for more complicated models. For example models that include interactions terms or polynomial regression terms.

Let T be the survival time of the Cox proportional model of (4.3), then from (3.5) we have

$$U = \exp\left[-H_0(t)\exp(\boldsymbol{\beta}^t\mathbf{Z} + \gamma y_i^*(t))\right]. \tag{4.4}$$

The cumulative baseline hazard function of the Weibull distribution is $H_0(t) = \lambda t^\nu$, with $h_0(t) > 0$ for all t, so H_0 can be inverted, and by inserting $y_i^*(t) = \alpha_{i0} + \alpha_{i1}T$ in (4.4) we have

$$\begin{aligned} U &= \exp\left[-\lambda T^\nu \exp(\boldsymbol{\beta}^t\mathbf{Z} + \gamma(\alpha_{i0} + \alpha_{i1}T))\right] \\ \log(U) &= -\lambda T^\nu \exp(\boldsymbol{\beta}^t\mathbf{Z} + \gamma(\alpha_{i0} + \alpha_{i1}T)) \\ \frac{-\log(U)}{\lambda\exp(\boldsymbol{\beta}^t\mathbf{Z} + \gamma(\alpha_{i0}))} &= T^\nu \exp(\gamma(\alpha_{i1}T)). \end{aligned} \tag{4.5}$$

Next we will implement the Lambert W Function to equation (4.5). For an equation of the form $X = Y\exp(Y)$, the Lambert W Function provides a solution for Y by inverting X, which has the form $Y = W(X)$, with W being the Lambert W Function. In order to implement the Lambert W Function to (4.5) we have to rewrite in the form $X = Y\exp(Y)$,

$$\begin{aligned} \left(\frac{-\log(U)}{\lambda\exp(\boldsymbol{\beta}^t\mathbf{Z} + \gamma(\alpha_{i0}))}\right)^{1/\nu} &= T\exp(\gamma(\alpha_{i1}T))^{1/\nu} \\ (\gamma\alpha_{i1}\frac{1}{\nu})\left(\frac{-\log(U)}{\lambda\exp(\boldsymbol{\beta}^t\mathbf{Z} + \gamma(\alpha_{i0}))}\right)^{1/\nu} &= (\gamma\alpha_{i1}\frac{1}{\nu})T\exp(\gamma(\alpha_{i1}T))^{1/\nu} \end{aligned} \tag{4.6}$$

A solution of (4.6) using the Lambert W Function is defined as

$$\gamma(\alpha_{i1}T\frac{1}{\nu}) = W\left(\gamma\left(\alpha_{i1}\frac{1}{\nu}\right)\left(\frac{-log(U)}{\lambda\exp(\boldsymbol{\beta}^t\mathbf{Z} + \gamma(\alpha_{i0}))}\right)^{1/\nu}\right), \tag{4.7}$$

and solving for T we have,

$$T = \frac{1}{\gamma(\alpha_{i1}\frac{1}{\nu})}W\left(\gamma\left(\alpha_{i1}\frac{1}{\nu}\right)\left(\frac{-log(U)}{\lambda\exp(\boldsymbol{\beta}^t\mathbf{Z} + \gamma(\alpha_{i0}))}\right)^{1/\nu}\right). \tag{4.8}$$

Dependent censoring

The methodology to introduce dependent censoring in our sample is the same as the one we introduced in chapter 3. We will simulate event and censoring times from equation (4.8). Then we take the minimum of those two values to obtain the observed survival time and the corresponding event indicator, one if the event is observed and zero if the event is censored. In our simulation we will use the Weibull distribution for the event times, so in that case the hazard functions for the time to event and censoring time are defined as

$$h_x(t|\mathbf{Z}) = h_{0_x}(t)\exp(\boldsymbol{\beta}^t\mathbf{Z} + \gamma y_i^*(t)), \tag{4.9}$$

$$h_c(t|\mathbf{Z}) = h_{0_c}(t)\exp(\boldsymbol{\phi}^t\mathbf{Z} + \gamma' y_i^*(t)), \tag{4.10}$$

with h_{0x} and h_{0c} the baseline hazard functions, $\boldsymbol{\beta}$ and $\boldsymbol{\phi}$ the coefficients for the time-fixed covariates relating to the event and censoring times respectively, and γ and γ' the parameters that links the 'true' longitudinal biomarker with the Cox proportional hazards model for time to event and time to censoring.

The dependency between event times and censoring times can be adjusted through the covariates by varying $\boldsymbol{\beta}$, $\boldsymbol{\phi}$, or the link parameters γ and γ'. The percentage of censoring can adjusted by varying the baseline hazards h_{0_x} and h_{0_c}.

The event time x_i, censoring time c_i and the observed pair of variables (t_i, δ_i) for each individual i with time-invariant covariates Age_i and $Treatment_i$ and a time-dependent covariate will be generated using the following algorithm.

Generate x_i, c_i pair of variables (t_i, δ_i) and covariates for subject i :

Step 1: Generate the baseline coviarates *Age* and *Treatment*, $Age \sim N(50, 10)$ and for Treatment, subjects will randomly receive Treatment A (Treatment=1) or Treatment B (Treatment=0) with equal probability 0.5. $\mathbf{Z}_i = (Age_i, Treatment_i)$, the vector of time-fixed covariates.

Step 2: Generate the expected longitudinal values of the biomarker $y_i^*(t)$ for individual i for each time point j using the linear growth model,

$$y_i^*(t_{ij}) = \alpha_{i0} + \alpha_{i1} t_{ij},$$

values α_{i0}, α_{i1}, are generated from a bivariate normal distribution with mean $\xi^T = (\xi_0, \xi_1)$, and variance G, which is the variance-covariance matrix of the random effects (b_{0i}, b_{1i}).

Step 3: Generate the observed values of the biomarker $y_i(t)$ from a multivariate normal distribution with mean $y_i^*(t)$ and variance $V = Z_i G Z_i^T + \Sigma_i$.

Step 4: Fit a linear mixed model to $y_i(t)$ to obtain the parameters $(\alpha_{i0}, \alpha_{i1})$. These parameters are used in equation (4.8) to generate the event and censoring times.

Step 5: Generate $U_{1_i}, U_{2_i} \sim U[0, 1]$.

Step 6: Generate x_i and c_i using equation (4.8)

$$x_i = \frac{1}{\gamma(\alpha_{i1}\frac{1}{\nu})} W\left(\gamma\left(\alpha_{i1}\frac{1}{\nu}\right)\left(\frac{-log(U_{i1})}{\lambda_x \exp(\boldsymbol{\beta}^t \mathbf{Z} + \gamma(\alpha_{i0}))}\right)^{1/\nu}\right),$$

$$c_i = \frac{1}{\gamma'(\alpha_{i1}\frac{1}{\nu'})} W\left(\gamma'\left(\alpha_{i1}\frac{1}{\nu'}\right)\left(\frac{-log(U_{i2})}{\lambda_c \exp(\boldsymbol{\phi}^t \mathbf{Z} + \gamma'(\alpha_{i0}))}\right)^{1/\nu'}\right).$$

Step 7: From x_i and c_i obtain the pair (t_i, δ_i) :

$$t_i = \min(x_i, c_i),$$

and

$$\delta_i = \begin{cases} 1 & \text{if} \quad x_i \le c_i, \\ 0 & \text{if} \quad x_i > c_i \end{cases}.$$

4.1.3 Estimation of survival curves

The methodology to compare how the presence of dependent censoring effects the estimation of the survival curve is the same as in chapter 3. We will compare the true survival curve with the curve estimated by the standard method (Kaplan-Meier) and we use the same methods as we did in chapter 3 to quantify the bias, i.e., the Average Area between the curves and the Maximum distance between curves.

4.1.4 Visualize the values of the biomarker

In Figure 4.1 we see the values of the biomarker for all 500 individuals of one simulated dataset for all 6 time points. For the generated values of the biomarker we used a linear growth model $y_i^*(t_{ij}) = \alpha_{i0} + \alpha_{i1} t_{ij}$, with $\alpha_{i0} = 2$ and $\alpha_{i1} = 1$ for the intercept and the slope respectively. Each line represents an individual and how the value of the biomarker increases through the time points.

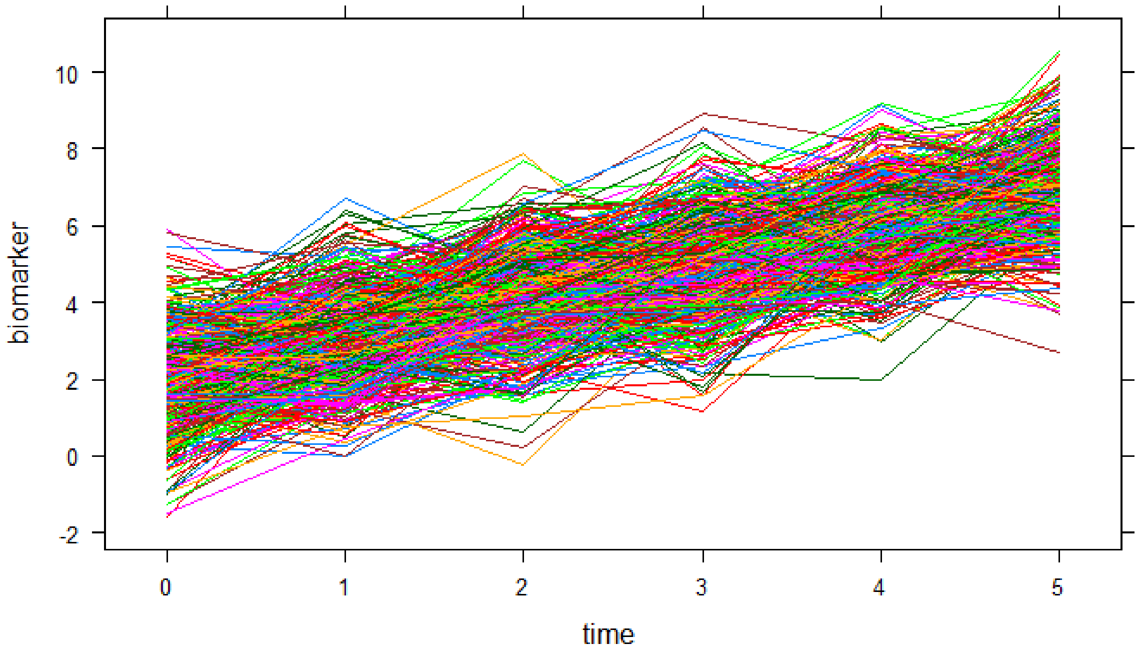


Figure 4.1: Individual profiles of the biomarker

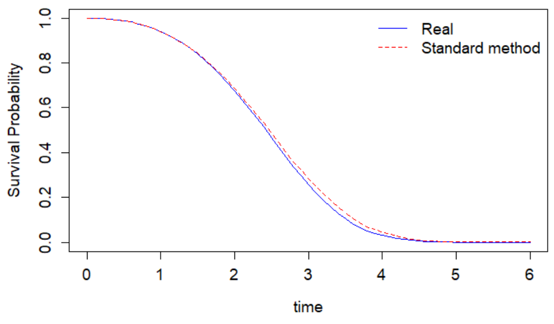
4.2 Simulation results for time-dependent covariate

To evaluate the influence of dependent censoring on survival curve estimation we will consider three different scenarios. Firstly we will vary the censoring percentage by keeping the coefficients ϕ and $\gamma^{'}$ of the censoring model the same and vary the scale parameter. Secondly we vary the strength of dependency by adjusting the coefficients ϕ of time-invariant covariates in the censoring model. Thirdly the strength of the parameter $\gamma^{'}$ that links the longitudinal model with the Cox proportional hazards model is varied. For this simulation a Weibull distribution will be assumed for baseline survival in the Cox model for event and censoring times, and we will use the algorithm we described in the previous section to generate event and censoring times. For every simulation scenario time to event will be simulated with the same coefficients $\beta = (0.1, 0.5)$, scale parameter $\lambda_X = 0.01$, and link parameter $\gamma = 0.5$. The shape parameter of the the Weibull distribution will be equal to $\nu = 2$ for all simulations for both the event and censoring times.

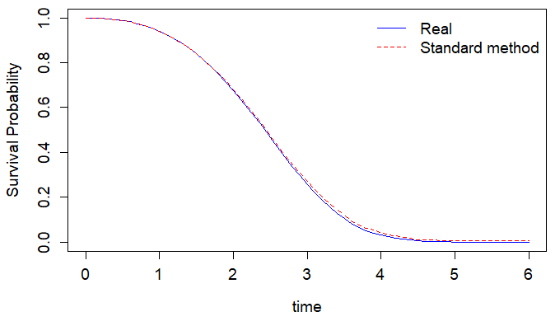
4.2.1 Strength of the dependency of the censoring mechanism on time-invariant covariates

The strength of the dependency in the time-invariant covariates can be varied by adjusting the coefficients $\phi = (\phi_1, \phi_2)$ of the censoring model. As the values of ϕ get bigger the censoring is more dependent on the covariates $\mathbf{Z} = (Age, Treatment)$. The values of ϕ will be $(0.5, 1.5)$ for strong dependency, $(0.1, 0.5)$ for moderate dependency, and $(0, 0.1)$ for weak dependency. For this simulation the sample size will be $n = 500$, the coefficients $(\beta_1, \beta_2) = (0.1, 0.5)$, the link parameter $\gamma = 0.5$, and the shape $\nu = 2$ and scale $\lambda_{0_x} = 0.01$ of the Weibull distribution for the event times model will have the same values for every simulation scenario. The link parameter $\gamma^{'} = 0.5$ and shape $\nu^{'} = 2$ parameter of the censoring model will also have the same value for every simulation and we will adjust the value of scale parameter λ_{0_c} to keep the censoring percentage at 33%.

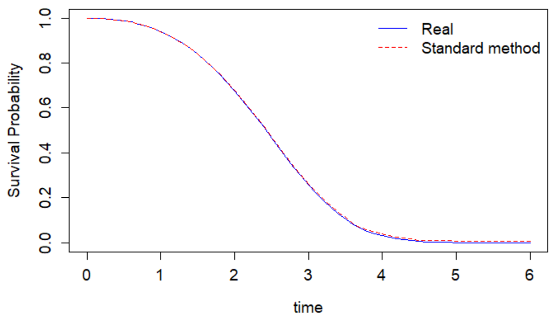
Figure 4.2 shows the results for the three different pairs of values for the ϕ coefficients. As the ϕ coefficients of the time-invariant covariates get bigger the covariates have bigger effect on the hazard. Thus the probability of getting censored is correlated with the probability of experiencing the event, correlation between the generated event and censoring times is 0.175 for $(\phi_1, \phi_2) = (0.5, 1.5)$ and drops to 0.061 for $(\phi_1, \phi_2) = (0, 0.1)$. This leads to overestimation of the survival probabilities using the standard Kaplan-Meier estimator since less events are observed specially at later points where more observations are censored. From the plots in Figure 4.2 we see as the dependency gets weaker as the fit of the Kaplan-Meier method gets better, but still as we see from Figure 4.2(b) even for a moderate value of $\phi = (0.1, 0.5)$ parameters there is a little difference between the two curves.



(a) $(\phi_1, \phi_2) = (0.5, 1.5), \lambda_c = 0.0028$



(b) $(\phi_1, \phi_2) = (0.1, 0.5), \lambda_c = 0.005$



(c) $(\phi_1, \phi_2) = (0, 0.5), \lambda_c = 0.0057$

Figure 4.2: *The real survival curve (blue) based on uncensored data and the survival curve estimated with the Kaplan-Meier method on censored data (red) for different values of ϕ coefficients. These are the average curves over 50 simulations with $(\beta_1, \beta_2) = (0.1, 0.5)$ and $\lambda_X = 0.01$. The link parameters γ and γ', shape parameter ν for both event and censoring times have the same values for all simulations, with $\gamma = \gamma' = 0.5$ and $\nu = 2$.*

Table 4.1 shows the parameters that were used for this simulation and the results of the area between the two curves[3] for three different cases of coefficients ϕ. The max distance shows the mean value of maximum distances in each simulated data the curves have. The average area gets smaller as the strength of dependency gets weaker. The max distance values as we see from table have a decreasing trend.

[3]Detailed description of area between the two curves and the max distance is given in chapter 3.

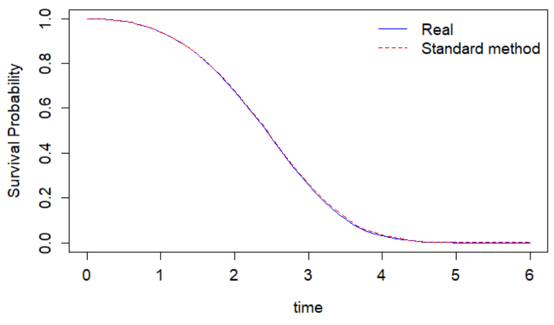
	n	λ_{0_x}	$\exp(\beta_1)$	$\exp(\beta_2)$	λ_{0_c}	$\exp(\phi_1)$	$\exp(\phi_2)$	Average area	Max distance
(ϕ_1,ϕ_2)=(0.5,1.5)	500	0.01	1.65	1.10	0.0028	1.65	4.50	0.012	0.041
(ϕ_1,ϕ_2)=(0.1,0.5)	500	0.01	1.65	1.10	0.005	1.10	1.65	0.007	0.030
(ϕ_1,ϕ_2)=(0,0.1)	500	0.01	1.65	1.10	0.0057	1.00	1.10	0.004	0.028

Table 4.1: Simulation parameters for event and censoring models, with $(\beta_1,\beta_2) = (0.1, 0.5)$, scale $\lambda_{0_x} = 0.01$, shape $\nu = 2$, link parameter $\gamma = 0.5$ for the event times model, shape $\nu^{'} = 2$ and link parameter $\gamma^{'} = 0.5$, for the censoring model and 33% censoring.

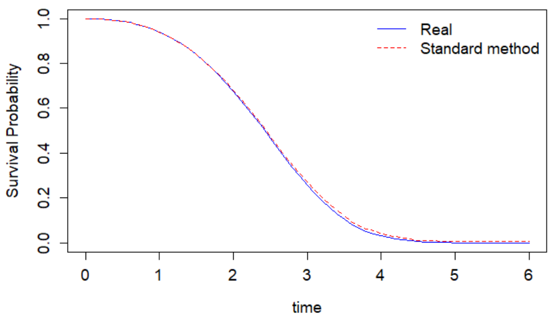
4.2.2 Strength of the link parameter γ of the time-dependent covariate

Next we will vary the strength of the link parameter $\gamma^{'}$ of the censoring model with values $(0.1, 0.5, 1, 1.5)$ for weak, moderate, strong and very strong association respectively. The sample size will be $n = 500$, the coefficients $(\beta_1, \beta_2) = (0.1, 0.5)$, link parameter $\gamma = 0.5$, shape $\nu = 2$ and scale $\lambda_{0_x} = 0.01$ for the events times model will have the same values for every simulation scenario. Also the coefficients $(\phi_1, \phi_2) = (0.1, 0.5)$, and shape $\nu^{'} = 2$ of the censoring model will have the same values in each scenario. We will adjust the value of the scale parameter λ_{0_c} to keep the censoring percentage at 33%.

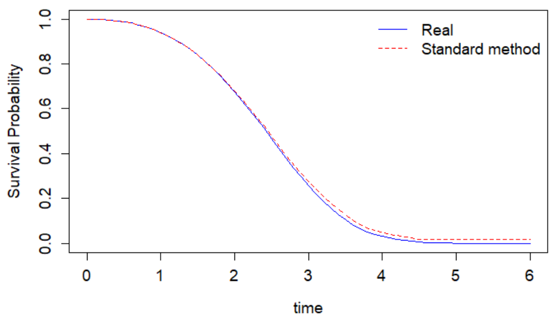
Figure 4.3 shows the results for the four different values for the link parameter $\gamma^{'}$ of the censoring model. As the link parameter $\gamma^{'}$ of the time-dependent covariate gets bigger the covariate has bigger effect on the hazard. Correlation between the generated event and censoring times is 0.174 for $\gamma^{'} = 1.5$ and drops to 0.058 for $\gamma^{'} = 0.1$, meaning individuals who have high probability experiencing the event have also a slightly higher probability of being censored. This leads to overestimation of the survival probabilities using the standard Kaplan-Meier estimator since fewer events are observed specially at later points where more observations are censored. From the plots in Figure 4.3 we see as the dependency gets stronger the fit of the Kaplan-Meier method gets worse, and the difference between the two curves is evident even for moderate association of the link parameter $\gamma^{'} = 0.5$.



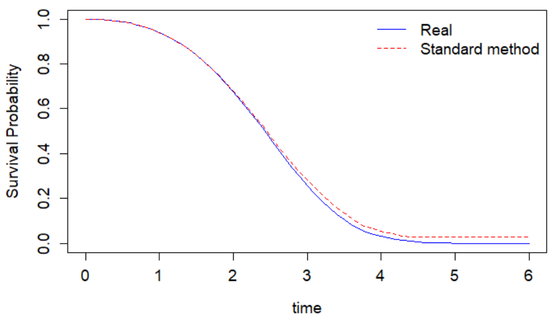
(a) $(\gamma, \gamma^{'}) = (0.5, 0.1), \lambda_{0_c} = 0.033$



(b) $(\gamma, \gamma^{'}) = (0.5, 0.5), \lambda_{0_c} = 0.005$



(c) $(\gamma, \gamma^{'}) = (0.5, 1), \lambda_{0_c} = 0.00045$



(d) $(\gamma, \gamma^{'}) = (0.5, 1.5), \lambda_{0_c} = 0.00004$

Figure 4.3: *The real survival curve (blue) based on uncensored data and the survival curve estimated with the Kaplan-Meier method on censored data (red) for different values of $\gamma^{'}$ link parameter of the censoring model.These are the average curves over 50 simulations with $(\beta_1, \beta_2) = (0.1, 0.5)$, $\gamma = 0.5$ and $\lambda_{0_x} = 0.01$ for time to event model. The coefficients ϕ_1,ϕ_2 for the censoring model are the same with values $(\phi_1, \phi_2) = (0.1, 0.5)$ and shape parameter ν for both event and censoring times have the same values for all simulations, with $\nu = 2$.*

	n	λ_{0_x}	exp(β_1)	exp(β_2)	λ_{0_c}	exp(ϕ_1)	exp(ϕ_2)	Average area	Max distance
$(\gamma,\gamma^{'})$=(0.5,0.1)	500	0.01	1.10	1.65	0.033	1.10	1.65	0.003	0.024
$(\gamma,\gamma^{'})$=(0.5,0.5)	500	0.01	1.10	1.65	0.005	1.10	1.65	0.007	0.030
$(\gamma,\gamma^{'})$=(0.5,1)	500	0.01	1.10	1.65	0.00045	1.10	1.65	0.009	0.042
$(\gamma,\gamma^{'})$=(0.5,1.5)	500	0.01	1.10	1.65	0.00004	1.10	1.65	0.011	0.053

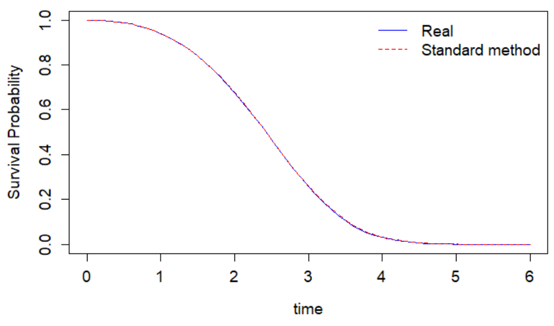
Table 4.2: Simulation parameters for event and censoring models, with $(\beta_1, \beta_2) = (0.1, 0.5)$,scale $\lambda_{0_x} = 0.01$, shape $\nu = 2$, link parameter $\gamma = 0.5$ for the event times model, and shape $\nu = 2$, $(\phi_1, \phi_2) = (0.1, 0.5)$, for the censoring model.

Results from Table 4.2 show that as we increase the value of the link parameter $\gamma^{'}$ of the censoring model the values of the Average area and Max distance have an increasing trend but a very small one suggesting that the impact of the link parameter $\gamma^{'}$ on censoring is small.

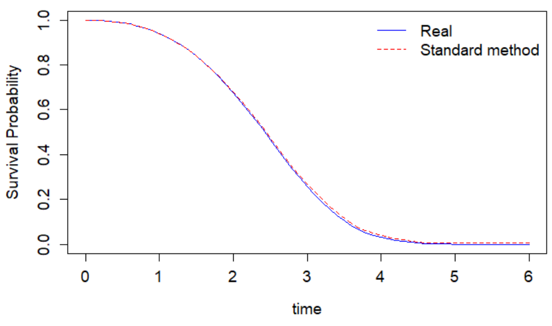
4.2.3 Percentage of censored subjects

Finally we will examine various percentages of censored subjects $(10\%, 30\%, 50\%)$, by adjusting the parameter λ_{0_c} of the censoring model. The sample size will be $n = 500$, the coefficients $(\beta_1, \beta_2) = (0.1, 0.5)$, link parameter $\gamma = 0.5$, shape $\nu = 2$ and scale $\lambda_x = 0.01$ for the events times model will have the same values for every simulation scenario. For the censoring model the coefficients $(\phi_1, \phi_2) = (0.1, 0.5)$, link parameter $\gamma^{'} = 0.5$, and shape $\nu^{'} = 2$ will have the same vale in each simulation scenario.

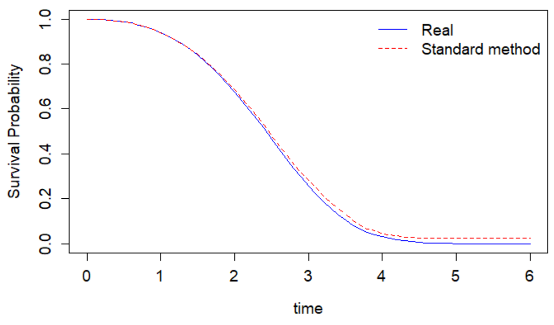
As expected, Figure 4.4 shows that as the percentage of censoring subjects increases, the difference between the real survival curve and the survival curve estimated on the censored data gets bigger.



(a) $(\phi_1, \phi_2) = (0.1, 0.5), \lambda_{0_c} = 0.0012$



(b) $(\phi_1, \phi_2) = (0.1, 0.5), \lambda_{0_c} = 0.0044$



(c) $(\phi_1, \phi_2) = (0.1, 0.5), \lambda_{0_c} = 0.01$

Figure 4.4: *The real survival curve (blue) based on uncensored data and the survival curve estimated with the Kaplan-Meier method on censored data (red) for different percentage of censored subjects.These are the average curves over 50 simulations with* $(\beta_1, \beta_2) = (0.1, 0.5)$, $(\phi_1, \phi_2) = (0.1, 0.5)$, $\lambda_{0_x} = 0.01$, *and varying scale parameter* λ_c. *The link parameters* γ,γ' *and shape parameter* ν *for both event and censoring times have the same values for all simulations, with* $\gamma = 0.5$ *and* $\nu = 2$.

	n	λ_x	$\exp(\beta_1)$	$\exp(\beta_2)$	λ_c	$\exp(\phi_1)$	$\exp(\phi_2)$	Average area	Max distance
10%	500	0.01	1.10	1.65	0.0012	1.10	1.65	0.002	0.012
30%	500	0.01	1.10	1.65	0.0044	1.10	1.65	0.005	0.027
50%	500	0.01	1.10	1.65	0.01	1.10	1.65	0.011	0.061

Table 4.3: Simulation parameters for event and censoring models, with $(\beta_1, \beta_2) = (0.1, 0.5)$, scale $\lambda_x = 0.01$, shape $\nu = 2$, and link parameter $\gamma = 0.5$ for the event times model, and $(\phi_1, \phi_2) = (0.1, 0.5)$, shape $\nu' = 2$ and link parameter $\gamma' = 0.5$, for the censoring model.

From Table 4.3 we can confirm from the values of Average area and Max distance that increasing the censoring percentage has a negative effect on the estimation of the survival curve.

4.3 Conclusions

The simulations that we carried out in this chapter for one time-dependent covariate and two time-invariant covariates, showed how the standard method to calculate survival curve performed in the presence of dependent censoring. The results are similar to the results from chapter 3 without the time-dependent covariate. Although there are differences between the curves they seem significant only when censoring percentage was 50%, or the coefficients ϕ of the time-independent covariates associated with the censoring model had values $(\phi_1, \phi_2) = (0.5, 1.5)$ or when the parameter $\gamma^{'}$ that links the time-dependent covariate with the survival model was larger than 1. For moderate values of the coefficients ϕ and link parameter $\gamma^{'}$, as we can see from Appendix A, the true survival curve is within the confidence interval of the estimated survival curve, and for more extreme values the true survival curve is outside the confidence intervals. Finally there is the issue of the impact the time-dependent covariate has on dependent censoring but we will explore this in chapter 6.

Chapter 5

IPCW with time-independent covariates

In the previous chapters we examined the issue of dependent censoring and how the Kaplan-Meier method can overestimate the survival probability in the case of dependent censoring. Although the differences between the true survival curve and the survival curve estimated with the standard Kaplan-Meier estimator were not very large, as we noticed from the metrics we used to quantify the difference for moderate values of censoring percentage and of the ϕ coefficients of the censoring model, the differences were more evident as the censoring percentage or the values of ϕ coefficients increased. In this chapter we will examine a method to correct for the presence of dependent censoring. Inverse Probability of Censoring Weighting (IPCW) was developed to correct for the issue of dependent censoring by giving extra weight to subjects who are not censored in order to compensate for censored subjects.

In this chapter we will give a brief introduction of the (IPCW) method, describe the algorithm that we will use to calculate the inverse probability of censoring weights and present the results of correction for the case of time-independent covariates.

5.1 IPCW theory

Inverse probability censoring weighting Robins [7], [8] corrects for dependent censoring by giving extra weight to subjects that are not censored. Thus these weights will be higher for subjects that the probability of remain uncensored is low, and small for subjects that the probability of remain uncensored is high. At each observed time point t, each subject is given a weight that is inversely proportional to the conditional probability of having remained uncensored until the time point t. The conditional probability is estimated from a Cox proportional hazards model or from the Product-Limit estimator, for censoring times. Thus the IPCW weights have to be calculated again for each subject that is at risk at each censoring time.

Let the weight for each subject j at a time point t be denoted as $W_j(t)$. The estimated probability to remained uncensored until that time point t is estimated either from the Cox proportional hazards model and denoted as $\hat{K}_j^{\mathbf{Z}}(t)$, where $\mathbf{Z}$ are the time-dependent and time independent covariates of the model, or from the Product-Limit estimator and denoted as $\hat{K}_j^0(t)$.

The estimator $W_j(t) = 1/\hat{K}_j^{\mathbf{Z}}(t)$ is sufficient to produce consistent weights. However in the case of dependent censoring and if the censoring percentage is big, the weights will be very large. In order to counter that Robins [7] suggested using $W_j(t) = \hat{K}_j^0(t)/\hat{K}_j^{\mathbf{Z}}(t)$. In case that the covariates $\mathbf{Z}$ do not influence the hazard of censoring at time t, the weights $W_j(t)$ will be close to one.

We can summarise the procedure to calculate the IPCW weights and the resulting survival probabilities in the following steps.

Calculate IPCW weights and survival probabilities S_{IPCW} :

Step 1: Fit a model for the censoring times with all the covariates that influence time to event and time to censoring.

Step 2: Estimate the probability to remain uncensored at each time point t for all subjects that are at risk at that time point, $\hat{K}_J^{\mathbf{Z}}(t)$ from the Cox proportional hazards model and $\hat{K}_J^0(t)$ from the Product-Limit estimator.

Step 3: Calculate the IPCW weights for each subject j at each time point t as, $W_j^{unstab}(t) = 1/\hat{K}_J^{\mathbf{Z}}(t)$ for the unstabilized weights and $W_j^{stab}(t) = \hat{K}_J^0(t)/\hat{K}_J^{\mathbf{Z}}(t)$ for the stabilized weights.

Step 4: Estimate the survival probabilities S_{IPCW} from the Cox proportional hazards model for time to event with subjects IPCW weights in the absence of censoring.

Next we will give a brief explanation of the steps to calculate the IPCW weights.

5.1.1 Step 1: Fit a model for the censoring times

For modelling the censoring times we will use the standard survival analysis but instead of time to event we will use time to censoring as the outcome. Hence since the event of interest is now time to censoring, subjects that were censored will have the event indicator and subjects that experienced the event are now considered censored and will have the censored indicator. Thus for a subject j the data representation for time to censoring is $(T_j, \delta_{C_j}, Z_j(t))$, where δ_{C_j} is the inverse of the event indicator δ, $\delta_{C_j} = 1 - \delta_j$. Using this data representation for time to censoring we can apply the Cox model with time-dependent covariates $h(t|\mathbf{Z}(t)) = h_0(t)\exp(\boldsymbol{\beta}^t\mathbf{Z}(t))$, from chapter 2 in order to calculate the probability of being censored.

One fundamental assumption of the IPCW method is that there are no unmeasured covariates that influence the censoring time. To evaluate the influence of the covariates on the probability of being censored, we will include all of them in the model. In order to estimate the probability of being censored based on some known covariates $\mathbf{Z}_j(t)$, we will use a Cox proportional hazards model.

The Cox proportional hazards model for the censoring times is defined as

$$h_C(t|\mathbf{Z}(t)) = h_{C_0}(t)\exp(\boldsymbol{\beta}^t{}_C\mathbf{Z}(t)), \tag{5.1}$$

where h_{C_0} is the baseline hazard for time to censoring, $\boldsymbol{\beta}_C$ is the vector with the regression coefficients and $\mathbf{Z}(t)$ is the vector with model time-dependent and time independent covariates.

5.1.2 Step 2: Estimate the probability to remain uncensored

From the equation (5.1), using the estimated hazard $h_C(t|\mathbf{Z}(t))$, we can estimate the probabilities of remaining uncensored for each subject j at each time point t. A Kaplan-Meier estimator for censoring that also includes the time-dependent covariates $Z_j(t)$, is defined as :

$$\hat{K}_j^{\mathbf{Z}}(t) = \prod_{[i;t_i<t,\delta_i=0]} [1 - \hat{h}_{C_0}(t_i)\exp(\boldsymbol{\beta}^t{}_C\mathbf{Z}_j(t_i))]. \tag{5.2}$$

The $\mathbf{Z}$ index of the $\hat{K}_j^{\mathbf{Z}}(t)$ is used in order to highlight the dependence of the estimator on the covariates $Z_j(t)$. Where the $\hat{K}_j^0(t)$ is the standard Kaplan-Meier estimator, that it does not acount for the covaraties $Z_j(t)$. In case the vector β_C is zero then $\hat{K}_j^{\mathbf{Z}}(t)$ and $\hat{K}_j^0(t)$ are equal.

5.1.3 Step 3: Calculate the IPCW weights

The weights for each subject j are calculate as $W_j^{unstab}(t) = 1/\hat{K}_j^{\mathbf{Z}}(t)$ for the unstabilized weights and $W_j^{stab}(t) = \hat{K}_j^0(t)/\hat{K}_j^{\mathbf{Z}}(t)$ for the stabilized weights. They are inversed to the conditional probability $\hat{K}_j^{\mathbf{Z}}(t)$ for a subject to remain uncensored until a time t. The weights increase in value as the percentage of censoring gets bigger, in the presence of dependent censoring, or near then end of the study when most of the subjects have already experienced the event of interest or they have been censored. In order to correct for this issue Robins proposed the use of the stabilized weights, which they will stay close to 1 if the covariates $\mathbf{Z}(t)$ do not influence the probability of being censored for any subject j.

5.1.4 Step 4: Estimate the survival probabilities S_{IPCW}

The survival probabilities can be estimated from the Product-Limit estimator introduced in chapter 2

$$\hat{S}(t) = \begin{cases} 1 & \text{if} \quad t < t_1, \\ \prod_{t_i \le t} \left[1 - \frac{d_i}{Y_i}\right], & \text{if} \quad t \ge t_1 \end{cases}, \tag{5.3}$$

where we substitute d_i the number of subjects that experience the event at time t_i with $\delta_i \hat{W}_i(t_i)$ which is a subject who experiences the event at time t_i multiplied by its weight, in the the absence of censoring. The denominator of (5.3) Y_i which represents the number of subjects at risk at time t_i is substituted by $\sum_{k=1}^n R_k(t_i)\hat{W}_k(t_i)$ which is the sum of all weights for subjects who are at risk at time t_i.

$$\hat{S}_{IPCW}(t) = \begin{cases} 1 & \text{if} \quad t < t_1, \\ \prod_{[i;t_i \le t]} \left[1 - \frac{\delta_i \hat{W}_i(t_i)}{\sum_{k=1}^n R_k(t_i)\hat{W}_k(t_i)}\right], & \text{if} \quad t \ge t_1 \end{cases}. \tag{5.4}$$

From equation (5.4) we notice that in case of stabilized weights the $\hat{K}_J^0(t)$ term cancels out both in numerator and the denominator resulting in the same survival probabilities for stabilized and unstabilized weights.

Note equation (5.4) holds only in datasets without ties. In case of ties in the dataset the Product-Limit estimator can be modified to account for ties. The term $\delta_i \hat{W}_i(t_i)$ from equation (5.4) can be substitute with the term $\sum_{[j;\delta_j(t_i)=1]} \hat{W}_j(t_i)$, which is the sum of weights of all subjects that experience the event at time point t_i. For this situation the survival probability is defined as

$$\hat{S}_{IPCW}(t) = \begin{cases} 1 & \text{if} \quad t < t_1, \\ \prod_{[i;t_i \le t]} \left[1 - \frac{\sum\limits_{[j;\delta_j(t_i)=1]} \hat{W}_j(t_i)}{\sum_{k=1}^n R_k(t_i)\hat{W}_k(t_i)}\right], & \text{if} \quad t \ge t_1 \end{cases}. \tag{5.5}$$

5.2 IPCW algorithm

To implement the IPCW algorithm in R we test two algorithms to calculate the weights, one proposed by Willems [12] and one proposed by M. van de Wal [11] and is implement in ipw package. We ran simulations with both of them and although the resulting weights were very similar some small differences were observed. The differences arise from the fact that the ipw package calculates weights at the end of the intervals compared to the method proposed by Willems [12], that calculates weights at the beginning of the interval. This issue arises from the way the intervals are interpreted from the two methods and whether weights should be calculate at the end or at the beginning of the interval. However figuring out the exact details of how weights are calculated is beyond the scope of this thesis and we decide to use the ipw method since it is a little less computational heavy than the method proposed by Willems [12].

5.3 Results for time-independent covariate

In the remainder of this chapter we will implement the ipw package to the data from chapter 3 and present and interpret the results.

5.3.1 Transform the data from wide to long format

In order to use the ipw package we need to transform the data from wide format to a long format. For each subject in the dataset the time to event or time to censoring will be divived in subintervals. The number of subintervals can be predetermined or be the number of unique event and censoring times for all subjects in the dataset, in our case we will choose the latter. Each subject j will require a number of rows r_j, each row will represent a time interval that a specific subject is at risk until the time the subject j experiences the event of interest or being censored. For time-fixed covariates, their value will be repeated in each row r_j, since their values do not change over time. For time-dependent covariates things are more complicated and we will explain it in the next chapter where we present the analysis for time-dependent covariates. The status indicators δ and δ_C will take the value 0 in every subinterval until the last one, where it will change to 1 in case of an event or 0 in case the subject is censored.

For transforming the data from wide to long format we will use the function **survSplit** from the library **survival**. The function splits each time interval in subintervals with boundaries at all the event and the censoring times, while also adjusting for the event status δ. However it does not adjust for the censoring status δ_C, so we will have to apply the function a second time by using the δ_C as the event indicator. Further explanation of how the transformation occurs will be in the Appendix B, where the code in R with comments will be presented.

5.4 Simulation for time-independent covariates

For the simulation in this chapter we will use the same artificial population as in chapter 3, the same method to generate survival and censoring times and introduce dependent censoring.

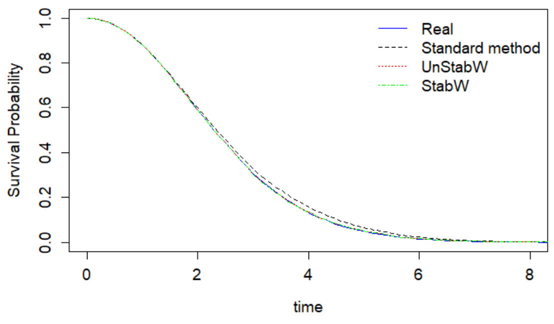
To evaluate the performance of the IPCW weights on the estimation of the survival curve we will consider five scenarios. Firstly we will evaluate the performance of IPCW method with a high value for the ϕ coefficients. Secondly we will check how the IPCW method performs when the ϕ coefficients have high values, thirdly we will check the performance of IPCW method when the percentage of censoring is very low and for moderate value of the ϕ coefficients. The reason we choose those 3 scenarios is to check how the IPCW method performs in these extreme cases, either for unusual high values of the ϕ coefficients or in the case of moderate value of the ϕ coefficients and low censoring. Lastly we will calculated the weights using the IPCW method, but only include one of the covariates that influence the censoring in the calculation of the IPCW weights, each time to see how the algorithm performs when we do not include all the covariates that influence the censoring. We will run these simulations once with Exponential distribution for the baseline survival in the Cox model for the event and censoring times and once with Weibull distribution, and with two time-independent covariates $\mathbf{Z} = (Age, Treatment)$. For every simulation scenario time to event will be simulated with the same coefficients as we did in chapter 3, $\beta = (0.5, 0.1)$, $\lambda_{0x} = 0.1$ for the Exponential distribution and scale parameter $\lambda_{0x} = 0.1$ and shape parameter $\nu = 2$ for the Weibull distribution.

5.4.1 Simulation results

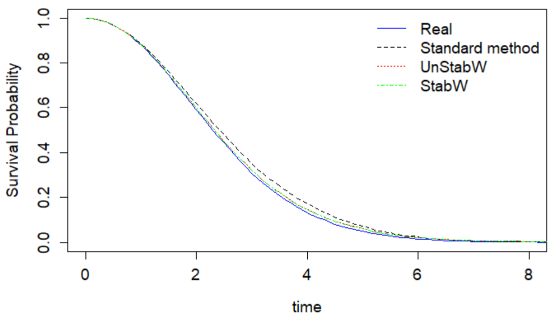
Next we will present results only for the Weibull distribution. Figures and tables for the Exponential distribution can be found in the Appendix A.3.

Weibull

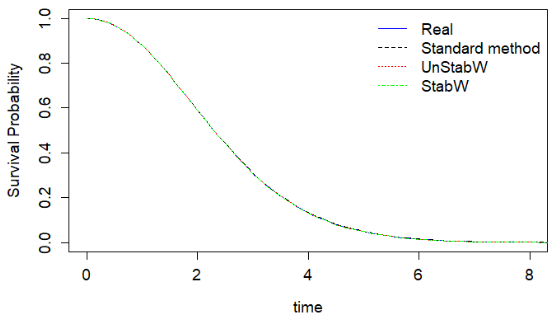
In the first scenario the ϕ coefficients of the censoring model have values $(1.5, 0.5)$ with $\lambda_{0c} = 0.027$ and censoring percentage (33%). As we see from Figure 5.1(a), the survival curves calculated with IPCW weights are very close to the survival curve based on the uncensored data, meaning that IPCW weights correct for the presence of dependent censoring. In the second scenario as the coefficients ϕ have extreme values $(4.5, 1.5)$, the IPCW weights do not completely correct for the dependent censoring but still perform better than the standard method, Figure 5.1(b). In the third we checked the stability of the IPCW by introducing a low censoring percentage (10%), as we see from Figure 5.1(c) the ICPW gives stable results even with a low percentage of censoring.



(a) $(\phi_1, \phi_2) = (1.5, 0.5), \lambda_{0_c} = 0.027$



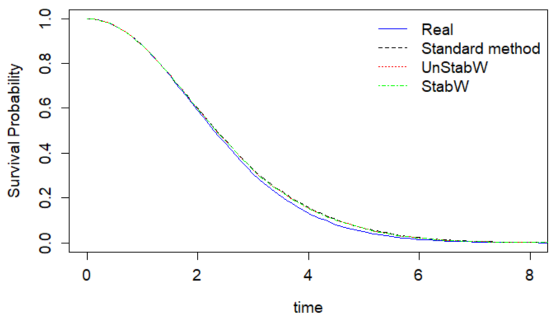
(b) $(\phi_1, \phi_2) = (4.5, 1.5), \lambda_{0_c} = 0.003$



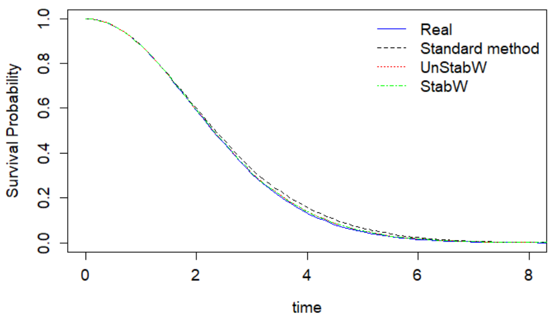
(c) $(\phi_1, \phi_2) = (0.5, 0.1), \lambda_{0_c} = 0.012$

Figure 5.1: *The real survival curve (blue) based on uncensored data, the survival curve estimated with the Kaplan-Meier method on censored data (black), and the survival curves estimated with IPCW weights, with stabilized weights (green) and unstabilized weights (red). These are the average curves over 50 simulations with* $(\beta_1, \beta_2) = (0.5, 0.1)$ *and* $\lambda_{0_x} = 0.1$. *The shape parameter* ν *for both event and censoring times have the same values for all simulations, with* $\nu = 2$.

The next two plots in Figure 5.2 show how the IPCW method performs if we omit one of the two time-independent covariates than influence censoring for the calculation of the IPCW weights. Figure 5.2(a) show the results of calculating the IPCW weights with only the *Age* covariate, and Figure 5.2(b) with only the *Treatment* covariate. The IPCW method performs better when we include in the model the *Treatment* covariate since the Age covariate is not very influential $\phi_2 = 0.5$ compare to $\phi_1 = 1.5$. Results of the survival curves for the IPCW weights from Figure 5.2(b) are very similar to Figure 5.1(a).



(a) $(\phi_1, \phi_2) = (1.5, 0.5), \lambda_{0_c} = 0.027$



(b) $(\phi_1, \phi_2) = (1.5, 0.5), \lambda_{0_c} = 0.027$

Figure 5.2: *The real survival curve (blue) based on uncensored data, the survival curve estimated with the Kaplan-Meier method on censored data (black), and the survival curves estimated with IPCW weights, with stabilized weights (green) and unstabilized weights (red). On figure (a) the weights are calculated with only Age as the time-invariant covariate, and figure (b) with only Treatment.*

	Fig 5.1			Fig 5.2	
	(a)	(b)	(c)	(a)	(b)
Standard method	0.012	0.02	0.0006	0.012	0.012
Unstabilized weights	0.0005	0.006	-0.0003	0.01	0.003
Stabilized weights	0.0005	0.006	-0.0003	0.01	0.003

Table 5.1: Area under the curve for all five simulations scenarios

Table 5.1 shows the area between the real survival curve the standard method, and the stabilized and unstabilized weights method, we also see how the IPCW weights perform better at estimating the survival curve than the standard method, even in case of column Fig 5.1(b) where the coefficients ϕ have extreme values. Finally we see from column Fig 5.2(b) that omitting the covariate with the smaller coefficient $\phi_2 = 0.5$ gives results relative close to scenario column Fig 5.1(a) with Area under the curve being 0.003 compare to 0.0005, where if we omit the covariate with coefficient $\phi_2 = 1.5$ the IPCW method does not perform well with Area under curve 0.01.

5.5 Discussion

In this chapter we carried out simulations to evaluate the performance of IPCW weights for different scenarios, where we consider different values for the ϕ coefficients, different censoring percentages and also with only one of the two coefficients for calculating the weights. From the first three scenarios we can conclude that the IPCW weights give very good correction, but in the case of unusual high values for the coefficients of the time-invariant covariates it only partly corrects. In the case of low censoring percentage gives reliable results similar to the ones from the standard method. Lastly seeing the performance of the IPCW weights when we omit one of the two covariates that influence the censoring from the calculation of the IPCW weights, we can conclude that even if there is an unmeasured covariate that influence the censoring the IPCW method will still perform slightly better than the standard method.

Chapter 6

IPCW with time-dependent covariate

In the previous chapter we examined the performance of the IPCW method on correcting the issue of dependent censoring in a dataset with only two time-independent covariates. In this chapter we will add one time-dependent covariate and we will examine the IPCW method to correct for the presence of dependent censoring. As in the previous chapter it is expected that the IPCW weights will perform better than the standard method.

We will apply the ipw package to the data from chapter 4 for the Weibull distribution but adjusted for the time-dependent covariate. We will use the same methodology we used in chapter 5 to calculate the inverse probability weights.

6.1 Results for time-dependent covariate

6.1.1 Transform the data from wide to long format

The transformation of the data set from wide to long format is more complicated in this chapter compared to the previous due to the presence of the time-dependent covariate. In the previous chapter the transformation from the wide to long format involved determining the number of intervals in which the dataset would be divided. Those intervals were based on the unique event and censoring times for all subjects in the dataset and then each subject j will be represent by a number of rows r_j, where each row will represent a time interval that a specific subject is at risk until the time subject j experiences the event or being censored. The value of the time-fixed covariates will be repeated in each row r_j. The value of the time-dependent covariates needs to be adjusted to match the observed value in the corresponding time interval.

In order to implement the correct values of the time-dependent covariate in the long format we will use to functions, **reshape** and **tmerge**. The **reshape** function will transform the dataset with the time-dependent covariate from a wide to long format and the tmerge function will merge the dataset with the survival times for each subject j with the values of the time-dependent covariates, creating a new dataset in a semi-long format where each person can have multiple rows but only rows that are relevant for that subject j. Next in this semi-long dataset we will apply the function **survSplit** two times, one to adjust for the event status and one to adjust for the censoring status.

6.1.2 Simulation set-up

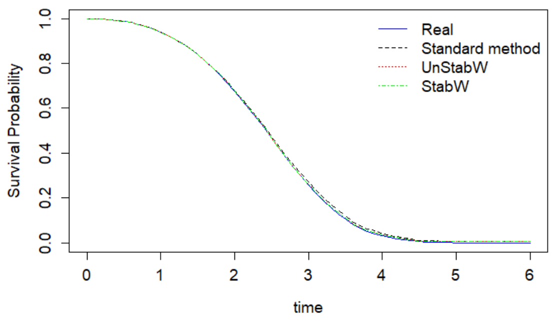
For the simulation in this chapter, we will use the data from Chapter 4 for the Weibull case, using the same method introduced in Chapter 4 for generating event and censoring times, and introducing dependent censoring.

To evaluate the performance of the IPCW weights on the estimation of the survival curve, we will consider six scenarios. First we will evaluate the IPCW method for 3 different values of the link parameter γ' of the censoring model, for weak, moderate and strong association, between the time-dependent covariate and the censoring model. In this chapter we will use more extreme values for the link parameter γ' compared to chapter 4. The reason is that we want to see how the IPCW method performs when the difference between the standard method to calculate the survival curve and the true survival curve is evident. Secondly we will check how stable the results are from the IPCW method for a small percentage of censoring and moderate value of the link parameter γ'. Lastly we will calculated the weights using the IPCW method when the censoring model is misspecified, in order to see how the algorithm performs when we do not include all the covariates that influence the censoring. This can be done by include only one of the time-independent covariates that influence the censoring in the calculation of IPCW weights each time.

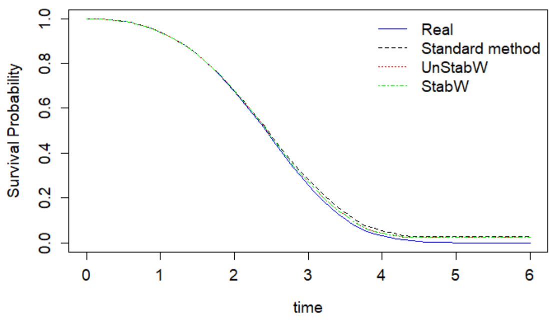
We will run these simulations with a Weibull distribution for the baseline survival in the Cox model for event and censoring times, with two time-independent covariates $\mathbf{Z} = (Age, Treatment)$ and with one time-dependent covariate. For every simulation scenario time to event will be simulated with the same coefficients as we did in chapter 4, $\beta = (0.5, 0.1)$, $\lambda_{0x} = 0.01$ and link parameter $\gamma = 0.5$ for the events model, and $\phi = (0.5, 0.1)$ for the censoring model. The shape parameter of the the Weibull distribution will be equal to $\nu = 2$ for all simulations for both the event and censoring times.

6.1.3 Simulations results

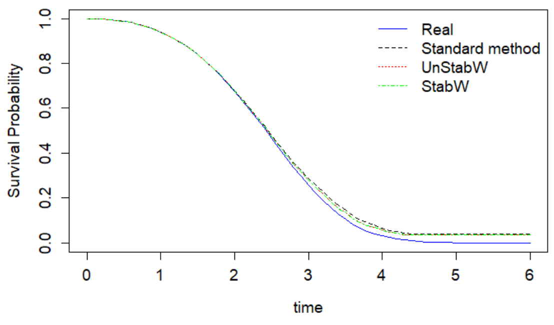
In the first scenario the link parameter γ' of censoring model has value 0.5 with $\lambda_{0c} = 0.005$ and censoring percentage 33%. As we see from Figure 6.1(a), the survival curves calculated with IPCW weights are very close to the true survival curve, meaning that the IPCW weights correct for the presence of dependent censoring. In the second scenario Figure 6.1(b), with a value of link parameter γ' equal to 1.5, the IPCW weights perform better than the standard method but their performance is getting worse and eventually is the same as the standard method at higher time points. Since less events occur at later time points, this leads to overestimation of the survival probabilities, which causes the IPCW method to perform worse than the previous scenario but still slightly better than the standard method. The link parameter γ' of the censoring model is not estimated very accurate, which in return leads to inaccurate IPCW weights. In the third scenario, with the value of the link parameter γ' equal to 3, the survival curve calculated with the IPCW weights is very close to the survival curve calculated with the standard method, meaning the IPCW weights do a poor job correcting for the presence of depending censoring. In the fourth scenario Figure 6.1(d), with a value of link parameter γ' equal to 1.5 and censoring percentage 10%, the IPCW weights give stable results.



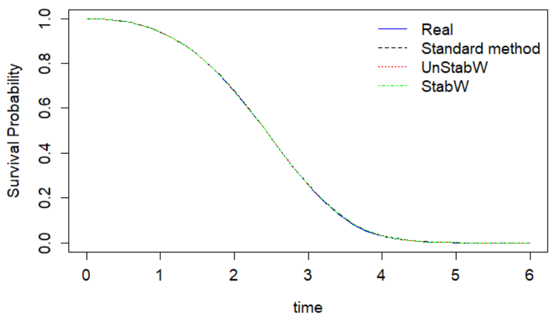
(a) $\gamma' = 0.5, \lambda_c = 0.005$



(b) $\gamma' = 1.5, \lambda_c = 0.00004$

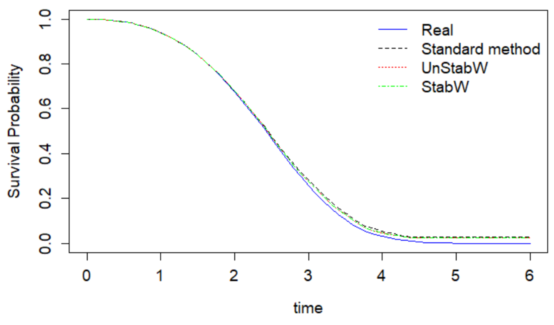


(c) $\gamma' = 3, \lambda_c = 0.00000003$

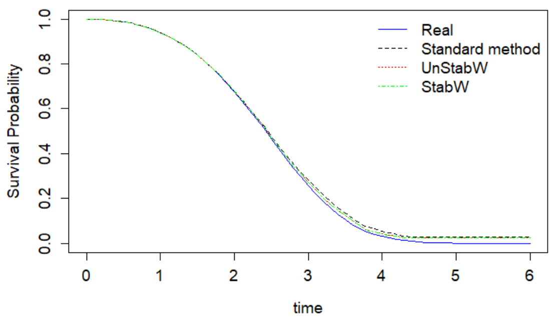


(d) $\gamma' = 0.5, \lambda_c = 0.0012$

Figure 6.1: *The real survival curve (blue) based on uncensored data, the survival curve estimated with the Kaplan-Meier method on censored data (black), and the survival curves estimated with IPCW weights, with stabilized weights (green) and unstabilized weights (red). These are the average curves over 50 simulations with* $(\beta_1, \beta_2) = (0.5, 0.1)$, $\lambda_{0x} = 0.01$,$\gamma = 0.5$ *and* $(\phi_1, \phi_2) = (0.5, 0.1)$ *for the censoring model. The shape parameter* ν *for both event and censoring times have the same values for all simulations, with* $\nu = 2$. *Censoring percentage is* 33% *for (a),(b) and (c) and* 10% *for (d).*



(a) $\gamma' = 1.5, \lambda_c = 0.005$



(b) $\gamma' = 1.5, \lambda_c = 0.0012$

Figure 6.2: *The real survival curve (blue) based on uncensored data, the survival curve estimated with the Kaplan-Meier method on censored data (black), and the survival curves estimated with IPCW weights, with stabilized weights (green) and unstabilized weights (red). On figure (a) the weights are calculated with only Age as the time-invariant covariate, and figure (b) with only Treatment.*

The additional two plots in Figure 6.2 show the performance of the IPCW weights if we omit from the calculation of the IPCW weights one of the two time-independent covariates.

Figure 6.2(a) shows the results of calculating the IPCW weights with only the *Age* covariate, and Figure 6.2(b) with only the *Treatment* covariate. The IPCW method performs better when we include in the calculation of the IPCW weights the *Treatment* covariate since the *Age* covariate is not very influential $\phi_2 = 0.1$ compared to $\phi_1 = 0.5$. The performance of IPCW weights is better than the standard curve but still they do not correct completely for the presence of dependent censoring,

	Fig 6.1				Fig 6.2	
	(a)	(b)	(c)	(d)	(a)	(b)
Standard method	0.0065	0.011	0.013	0.0017	0.011	0.011
Unstabilized weights	0.0024	0.0057	0.009	0.0008	0.008	0.0059
Stabilized weights	0.0024	0.0057	0.009	0.0008	0.008	0.0059

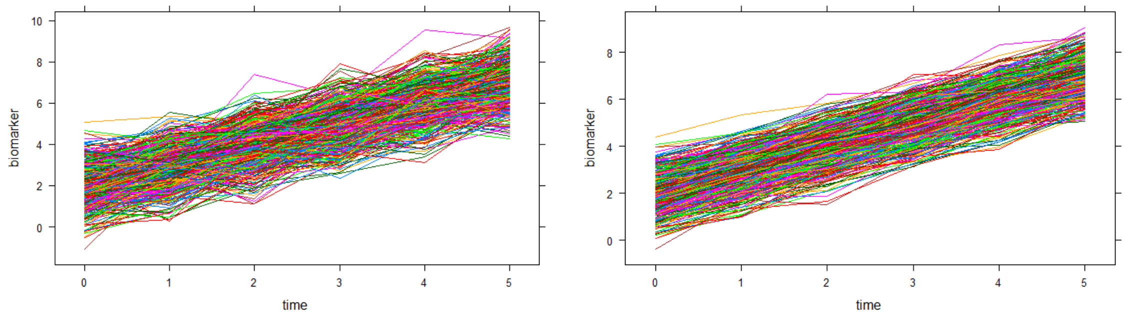
Table 6.1: Area under the curve for all five simulations scenarios

Table 6.1 shows the area between the real survival curve the standard method, and the stabilized and unstabilized weights method. We see that although the IPCW weights perform better than the standard method, the correction does not seem very significant. Finally if we omit one of the two time-independent covariates, the IPCW method performs worse in case we omit the covariate with the bigger influence $\phi_1 = 0.5$, as we see from column Fig 6.2(a) compared to column Fig 6.1(b), and almost the same if we omit the covariate with the smaller influence $\phi_2 = 0.1$, column Fig 6.2(b). This confirms our previous observation from the graphs.

6.2 Estimated value of the link parameter γ' of the censoring model

In the previous section we mention that one of the reasons that the IPCW method fails to produce accurate results is the bad estimation of the parameters that influence censoring and more specific the link parameter γ'. In this section we will try to find why this is the case.

In Chapter 4 we describe in detail the algorithm to generate event and censoring times for a time-dependent covariate. Step 2 through step 4 describe how we generate the expected longitudinal values of the biomarker $y_i^*(t)$, then how we generate the observed values of the biomarker $y_i(t)$ from a multivariate normal distribution with mean $y_i^*(t)$ and variance $V = Z_i G Z_i^T + \Sigma_i$, and finally fit a linear mixed model to $y_i(t)$ to obtain the parameters $(\alpha_{i0}, \alpha_{i1})$ that we used to generate the event and censoring times. The issue lies with the variance-covariance matrix $V = Z_i G Z_i^T + \Sigma_i$ and the error we introduced with the diagonal matrix Σ . The bigger the values of the diagonal elements of diagonal matrix Σ are, which means there is bigger variation for the observed values of the biomarker for each subject between time points, the worse is the estimation for the parameters of the censoring model and mainly for the link parameter γ'. In order to illustrate the problem we run simulations with reduced values of the diagonal matrix Σ, $diag(\Sigma) = 0.5$ and $diag(\Sigma) = 0.1$ instead of $diag(\Sigma) = 1$, and the estimation of the link parameter γ' was getting better as values of the diagonal elements of matrix Σ were smaller. Below we present the spaghetti plots that show how the longitudinal measurements for each subject look when we reduce the value of matrix Σ in the variance-covariance matrix.



(a) $diag(\Sigma) = 0.5$ (b) $diag(\Sigma) = 0.1$

Figure 6.3: *Individual profiles of the biomarker for different values of the diagonal matrix* Σ.

In Table 6.2 below we show the estimated values of the parameters of the censoring values for different values of the diagonal matrix Σ. The values we used to generate the event and censoring times were $(\phi_1, \phi_2) = (0.5, 0.1)$ and $\gamma' = 0.5$.

	$diag(\Sigma) = 1$	$diag(\Sigma) = 0.5$	$diag(\Sigma) = 0.1$	true values
ϕ_1	0.494	0.503	0.510	0.5
ϕ_2	0.109	0.108	0.109	0.1
γ'	0.120	0.232	0.387	0.5

Table 6.2: Values of the estimated parameters of the censoring model for different values of the diagonal matrix Σ.

We noticed from Table 6.2 that as the value of the diagonal matrix Σ is reduced the estimated value of the link parameter γ' is getting better but also the variation on the individual profiles of the subjects decreases. In the Appendix A.4 we present results of simulations with the value of the diagonal matrix Σ equal to 0.1, and it is evident that the IPCW weights are more accurate and the IPCW method corrects better for the presence of dependent censoring in that case.

6.3 Discussion

In this chapter we carried out simulations to evaluate the performance of the IPCW method in the presence of a time-dependent covariate. We considered six scenarios, in which we vary the value of the link parameter γ' of the censoring model, the censoring percentage and with only one of the two time-independent covariates. Increasing the value of the link parameter γ' made the performance of the IPCW weights worse than expected, since less events are observed and survival probabilities are overestimated leading in return to overestimation from the IPCW method. In theory the IPCW weights should completely correct for the presence of dependent censoring, but since the Cox model does not produce a perfect fit for the time to censoring model, it leads to not very accurate weights for the IPCW. Although the IPCW method performs better in all scenarios that we examine compared to the standard method, the difference in performance does not seem very significant to justify the extra work of calculating the IPCW weights.

Chapter 7

Discussion

The aim of this thesis was firstly to examine the effect that the presence of dependent censoring can have on estimated survival curves, and secondly examine a way that can correct for the presence of dependent censoring. To address the first issue we ran simulations by using different approaches, using time-independent covariates and a time-dependent covariate, and the event and censoring times were generated from an Exponential and a Weibull distribution. In order to correct for the presence of dependent censoring we used the Inverse Probability of Censoring Weighting (IPCW) method.

Simulation studies were carried out in the first part of this thesis to evaluate the bias that dependent censoring can introduce to the estimates for survival probabilities, if we assume independent censoring. We included two main scenarios, one in which the generated survival data included two time independent covariates and one that included two time independent covariates and one time dependent covariate. Then for each of the two main scenarios we generated event and censoring times from the exponential distribution with constant baseline hazards rates and from the Weibull distribution with no constant baseline hazards rates. Next we evaluated different scenarios for each case of the exponential and Weibull distribution by varying the sample size, the percentage of censored subjects and the strength of the dependency of the censoring mechanism on the covariates. Results showed that the survival probabilities estimated by the standard Kaplan-Meier estimator that assumes independent censoring were too high, but this overestimation did not seem significant in most scenarios. Only in scenarios with high censoring percentage (50%) or for high values of the coefficients ϕ and the link paramater $\gamma^{'}$ that link the censoring and time to event process, the difference seemed significant enough that if we do not account for the dependent censoring can lead to seriously biased estimates of survival probabilities.

In the second part of this thesis we carried out two simulations to evaluate the performance of the IPCW method in the presence of dependent censoring. For the simulation with the two time independent covariates, we generated event and censoring times both with the exponential and the Weibull distribution, and for the simulation with the one time dependent covariate and two time independent covariates only with the Weibull distribution. In both simulations we examined scenarios that vary the strength of the dependency of the censoring mechanism on the covariates, the ϕ coefficients in the case of only time independent covariates and only link parameter $\gamma^{'}$ in the case of the time dependent covariate. We also included one extra scenario were we omit one of the two time independent covariates from the calculation of the IPCW weights. The IPCW method performed very well compared to the standard method in the case of only time independent covariates, although it loses a lot of its efficiency for unusual high values of the coeffiecients ϕ due to imprecise fit of the censoring model, but still performs better than the standard method in those cases. In the case of two independent covariates and one time dependent covariate the IPCW method is less accurate than the previous simulation mainly due to imprecise estimate of the link parameter $\gamma^{'}$ of censoring model, but still performs slightly better overall than the standard method. Finally the results from the scenarios where we omitted one of the two time independent covariates from the calculation of the weights seems to suggest that the IPCW can perform better than the standard method even in the case we have an unknown covariate that influences the censoring mechanism.

Overall the IPCW method performed well and corrected for the presence of dependent censoring. Specially in scenarios were the coefficients that influence the censoring had moderate and more realistic values. However in those scenarios the difference in performance was not significantly different from the standard method as bias introduced by the dependent censoring was relatively limited. In scenarios were the dependence was strong either due the high value of the censoring coefficients or a high percentage of censoring, the IPCW method does not perform so well but still performed better than the standard method. Results showed that a good fit for the censoring model is very important in order for the IPCW method to perform well and this is specially the case when we included a longitudinal covariate in the model. In this latter case measurement error in the observed values of the longitudinal covariate led to inaccurate results from the censoring model and thereby only partial reduction of the bias introduced by the dependent censoring.

One limitation of this thesis is that we examine a simple case of a time dependent covariate and our methodology is not applicable in the case of more complicated models that, e.g., include interaction terms between the time dependent and time independent covariate. Another limitation is we did not consider the issue of missing data that is common in real life longitudinal measurements, since subjects may miss measurement visits, leading to missing data on the time dependent covariate. Further work is needed to examine the issue of dependent censoring in the case of more complicated models that, e.g., include more than one time dependent covariate, competing risks outcomes and missing data.

Bibliography

[1] P. C. Austin. “Generating survival times to simulate Cox proportional hazards models with time-varying covariates”. In: *Stat Med* 31.29 (Dec. 2012), pp. 3946–3958.

[2] R. Bender, T. Augustin, and M. Blettner. “Generating survival times to simulate Cox proportional hazards models”. In: *Stat Med* 24.11 (June 2005), pp. 1713–1723.

[3] D. J. Hendry. “Data generation for the Cox proportional hazards model with time-dependent covariates: a method for medical researchers”. In: *Stat Med* 33.3 (Feb. 2014), pp. 436–454.

[4] John P Klein and Melvin L Moeschberger. *Survival analysis: techniques for censored and truncated data*. Vol. 2. Springer, 2003.

[5] Y. Matsuyama and T. Yamaguchi. “Estimation of the marginal survival time in the presence of dependent competing risks using inverse probability of censoring weighted (IPCW) methods”. In: *Pharm Stat* 7.3 (2008), pp. 202–214.

[6] Julius S Ngwa et al. “Generating survival times with time-varying covariates using the Lambert W function”. In: *Communications in Statistics-Simulation and Computation* (2019), pp. 1–19.

[7] James M Robins and Dianne M Finkelstein. “Correcting for noncompliance and dependent censoring in an AIDS clinical trial with inverse probability of censoring weighted (IPCW) log-rank tests”. In: *Biometrics* 56.3 (2000), pp. 779–788.

[8] James M. Robins and Andrea Rotnitzky. “Recovery of Information and Adjustment for Dependent Censoring Using Surrogate Markers”. In: *AIDS Epidemiology: Methodological Issues*. Ed. by Nicholas P. Jewell, Klaus Dietz, and Vernon T. Farewell. Boston, MA: Birkhäuser Boston, 1992, pp. 297–331.

[9] M. P. Sylvestre and M. Abrahamowicz. “Comparison of algorithms to generate event times conditional on time-dependent covariates”. In: *Stat Med* 27.14 (June 2008), pp. 2618–2634.

[10] Anastasios A Tsiatis, Victor Degruttola, and Michael S Wulfsohn. “Modeling the relationship of survival to longitudinal data measured with error. Applications to survival and CD4 counts in patients with AIDS”. In: *Journal of the American Statistical Association* 90.429 (1995), pp. 27–37.

[11] Willem M van der Wal and Ronald B Geskus. “ipw: an R package for inverse probability weighting”. In: *Journal of Statistical Software* 43 (2011), pp. 1–23.

[12] S. Willems et al. “Correcting for dependent censoring in routine outcome monitoring data by applying the inverse probability censoring weighted estimator”. In: *Stat Methods Med Res* 27.2 (Feb. 2018), pp. 323–335.

[13] Mai Zhou. “Understanding the Cox regression models with time-change covariates”. In: *The American Statistician* 55.2 (2001), pp. 153–155.

Appendices

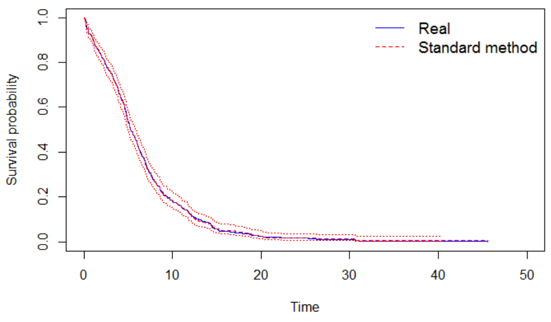
Appendix A

Confidence intervals and simulations

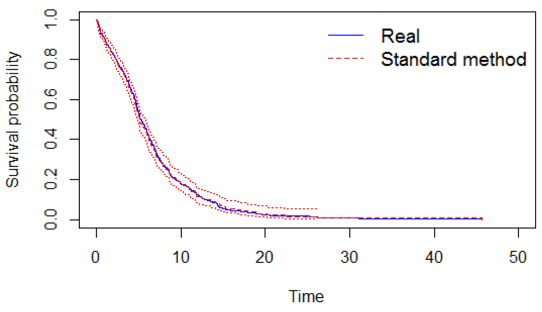
In Appendix A we will present some plots with the true survival curve and the curve from the standard method with confidence intervals for the standard method.

A.1 Chapter 3

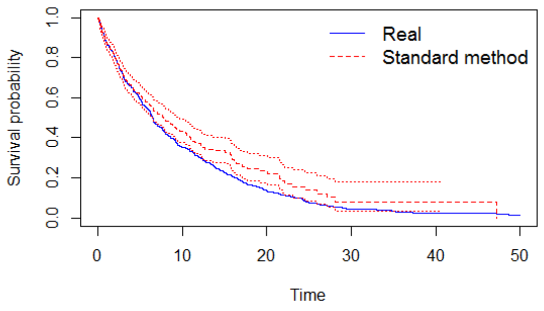
A.1.1 Confidence intervals for exponential distributed event and censoring times



(a) 10%, $\lambda_{0_C} = 0.027$

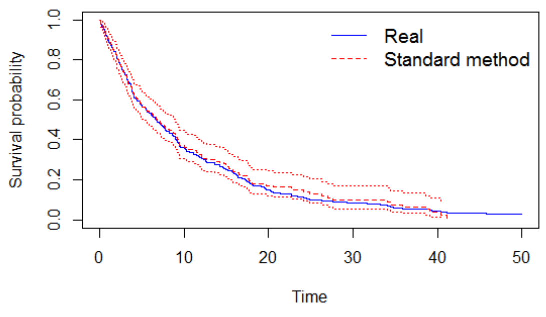


(b) 30%, $\lambda_{0_C} = 0.05$

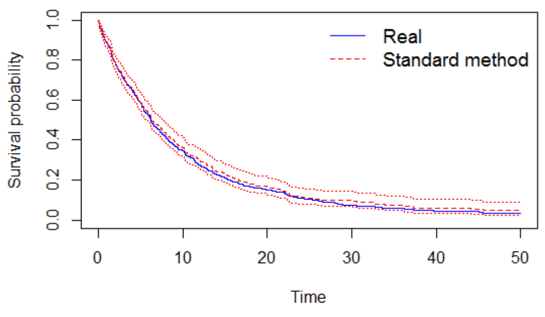


(c) 50%, $\lambda_{0_C} = 0.058$

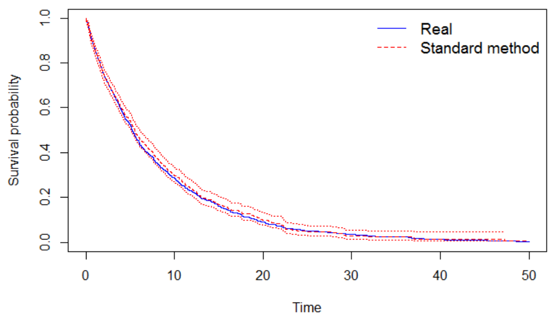
Figure A.1: *The real survival curve (blue) based on uncensored data and the survival curve estimated with the Kaplan-Meier method on censored data (red) with confidence interval for different percentage of censored subjects. These are the curves over 1 simulation with* $(\beta_1, \beta_2) = (0.5, 0.1)$ *and* $\lambda_{0_X} = 0.1$*, for the events model and* $(\phi_1, \phi_2) = (0.5, 0.1)$*, and different values* λ_{0_C} *for the censoring model, and sample size* $n = 500$*.*



(a) $n = 250$



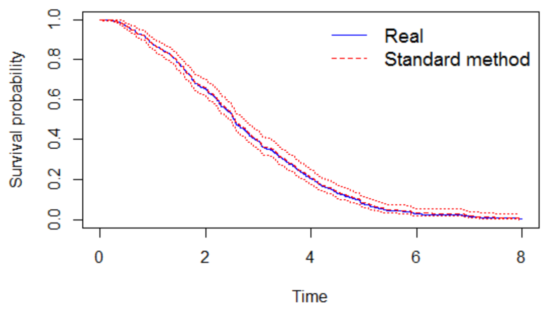
(b) $n = 500$



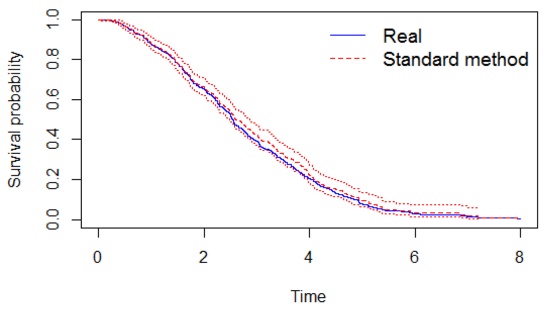
(c) $n = 1000$

Figure A.2: *The real survival curve (blue) based on uncensored data and the survival curve estimated with the Kaplan-Meier method on censored data (red) with confidence interval for different sample sizes. These are the average curves over 1 simulation with* $(\beta_1, \beta_2) = (0.5, 0.1)$, $\lambda_{0_X} = 0.1$ *and* $(\phi_1, \phi_2) = (0.5, 0.1)$, $\lambda_{0_C} = 0.05$, *for the event and censoring model respectively.*

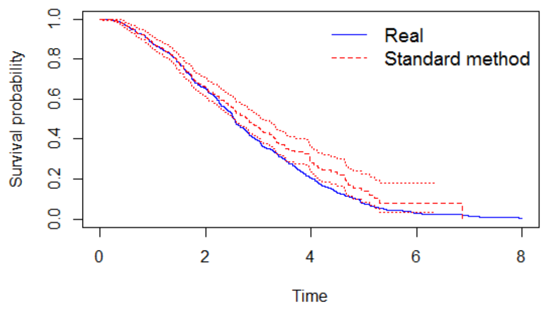
A.1.2 Confidence intervals for Weibull distributed event and censoring times



(a) 10%, $\lambda_C = 0.012$

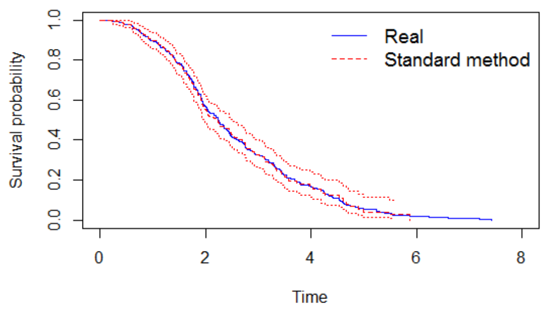


(b) 30%, $\lambda_C = 0.045$

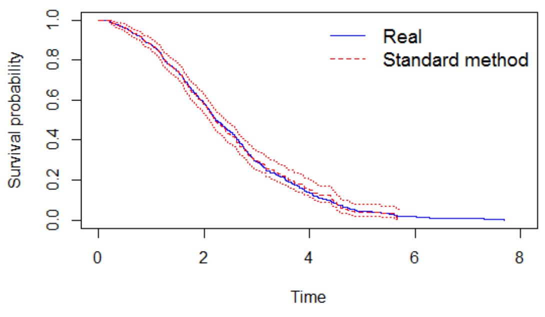


(c) 50%, $\lambda_C = 0.1$

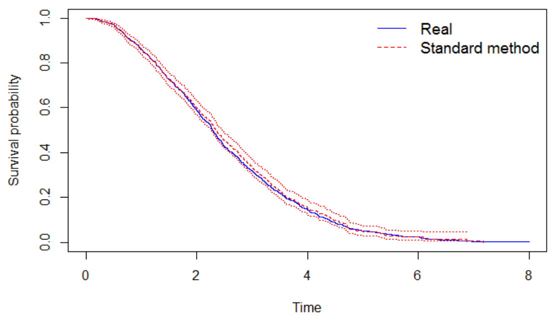
Figure A.3: *The real survive curve (blue) based on uncensored data and the survival curve estimated with the Kaplan-Meier method on censored data (red) with confidence interval for different percentage of censored subjects.These are the average curves over 1 simulation with* $(\beta_1, \beta_2) = (0.5, 0.1)$, $\lambda_{0_X} = 0.1$ *and shape* $\nu_x = 2$, *for the events model and* $(\phi_1, \phi_2) = (0.5, 0.1)$, *shape* $\nu_c = 2$ *and different values* λ_{0_C} *for the censoring model, and sample size* $n = 500$.



(a) $n = 250$



(b) $n = 500$

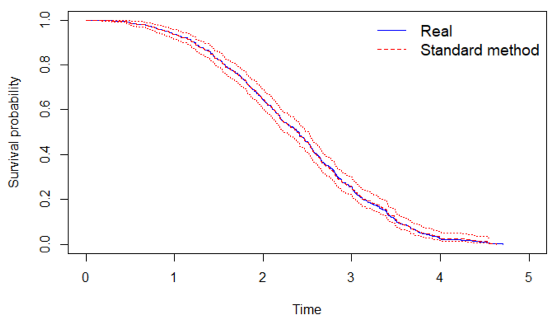


(c) $n = 1000$

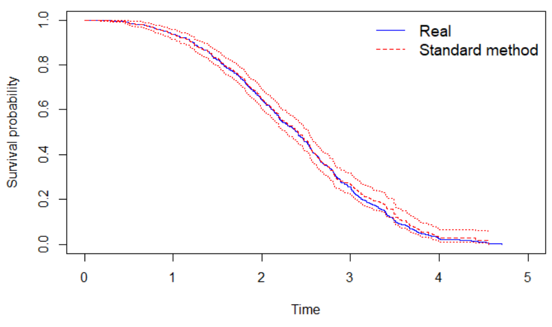
Figure A.4: *The real survival curve (blue) based on uncensored data and the survival curve estimated with the Kaplan-Meier method on censored data (red) with confidence intervals for different sample sizes. These are the average curves over 1 simulation with* $(\beta_1, \beta_2) = (0.5, 0.1)$, $\lambda_{0_X} = 0.1$, $\nu_x = 2$ *and* $(\phi_1, \phi_2) = (0.5, 0.1)$, $\lambda_{0_C} = 0.05$, $\nu_c = 2$ *for the event and censoring model respectively.*

A.2 Chapter 4

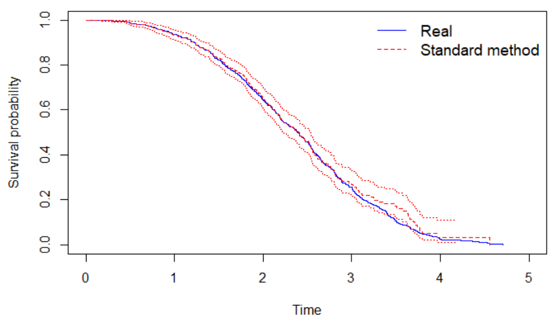
Confidence intervals for Weibull distributed event and censoring times with time-dependent covariate



(a) 10%, $\lambda_{0_C} = 0.0012$

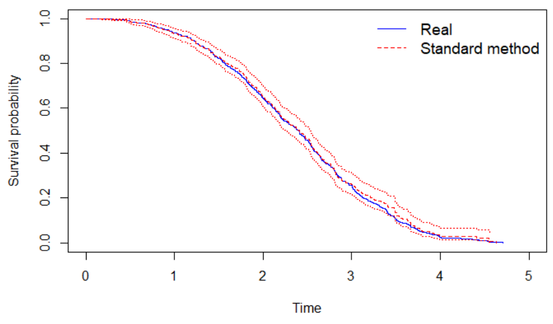


(b) 30%, $\lambda_{0_C} = 0.0044$

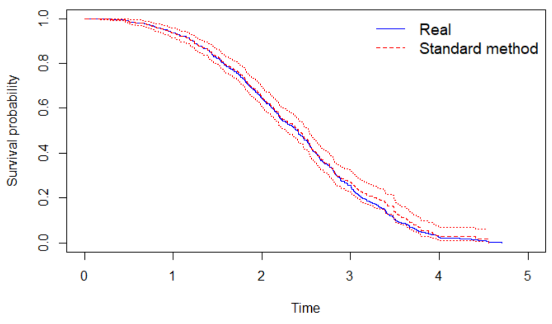


(c) 50%, $\lambda_{0_C} = 0.01$

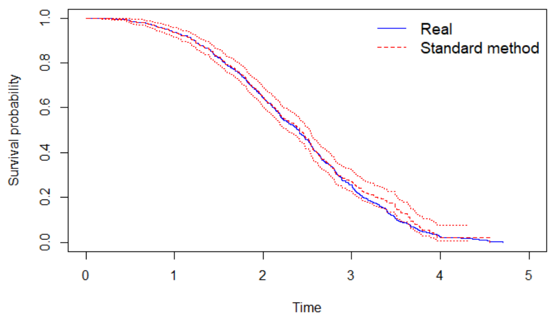
Figure A.5: *The real survival curve (blue) based on uncensored data and the survival curve estimated with the Kaplan-Meier method on censored data (red) with confidence interval for different percentage of censored subjects.These are the curves over 1 simulation with* $(\beta_1, \beta_2) = (0.1, 0.5)$ *and* $\lambda_x = 0.01$*, for the events model and* $(\phi_1, \phi_2) = (0.1, 0.5)$*, and different values* λ_c *for the censoring model. The link parameters* γ,$\gamma^{'}$ *and shape parameter* ν *for both event and censoring times have the same values for all simulations, with* $\gamma = 0.5$ *and* $\nu = 2$.



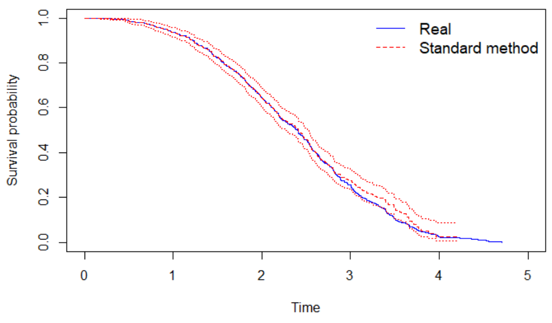
(a) $(\gamma, \gamma^{'}) = (0.5, 0.1), \lambda_c = 0.033$



(b) $(\gamma, \gamma^{'}) = (0.5, 0.5), \lambda_c = 0.005$



(c) $(\gamma, \gamma^{'}) = (0.5, 1), \lambda_c = 0.00045$

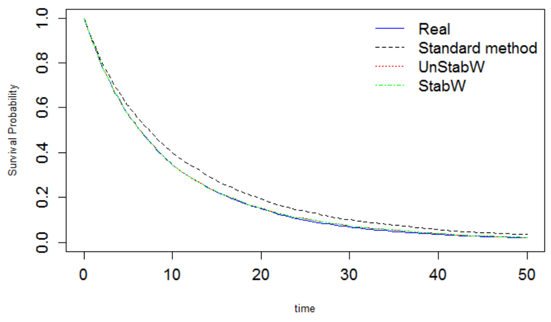


(d) $(\gamma, \gamma^{'}) = (0.5, 1.5), \lambda_c = 0.00004$

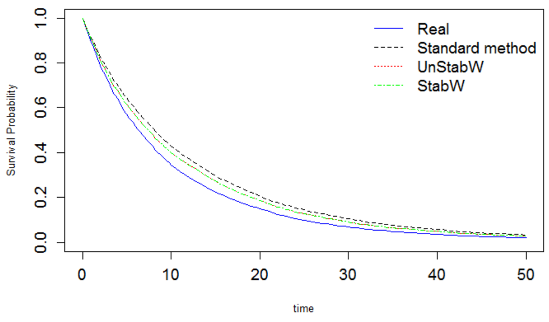
Figure A.6: *The real survival curve (blue) based on uncensored data and the survival curve estimated with the Kaplan-Meier method on censored data (red) for different values of* $\gamma^{'}$ *link parameter of the censoring model.These are the curves over 1 simulations with* $(\beta_1, \beta_2) = (0.1, 0.5)$, $\gamma = 0.5$ *and* $\lambda_x = 0.01$ *for time to event model. The coefficients* ϕ_1,ϕ_2 *for the censoring model are the same with values* $(\phi_1, \phi_2) = (0.1, 0.5)$ *and shape parameter* ν *for both event and censoring times have the same values for all simulations, with* $\nu = 2$.

A.3 Chapter 5

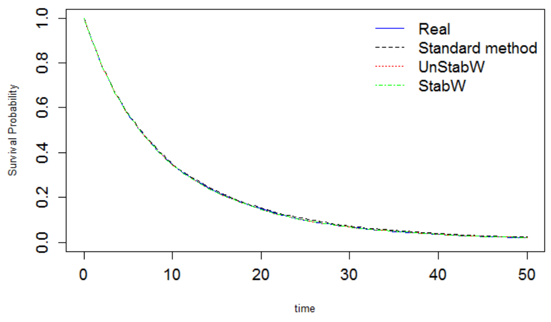
Exponential



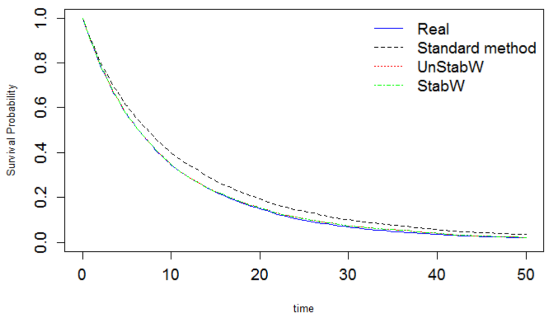
(a) $(\phi_1, \phi_2) = (1.5, 0.5), \lambda_c = 0.034$



(b) $(\phi_1, \phi_2) = (4.5, 1.5), \lambda_c = 0.007$



(c) $(\phi_1, \phi_2) = (0.5, 0.1), \lambda_c = 0.012$

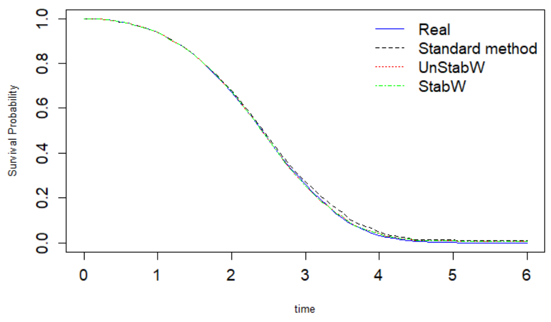


(d) $(\phi_1, \phi_2) = (1.5, 0.5), \lambda_c = 0.034$

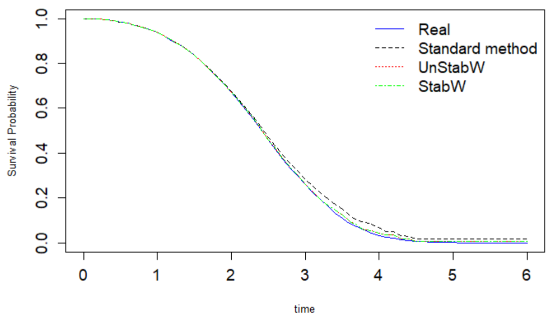
Figure A.7: *The real survival curve (blue) based on uncensored data and the survival curve estimated with the Kaplan-Meier method on censored data (red) for different values of ϕ coefficients.These are the average curves over 50 simulations with $(\beta_1, \beta_2) = (0.1, 0.5)$ and $\lambda_X = 0.01$. The link parameters γ and $\gamma^{'}$, shape parameter ν for both event and censoring times have the same values for all simulations, with $\gamma = \gamma^{'} = 0.5$ and $\nu = 2$.*

A.4 Chapter 6

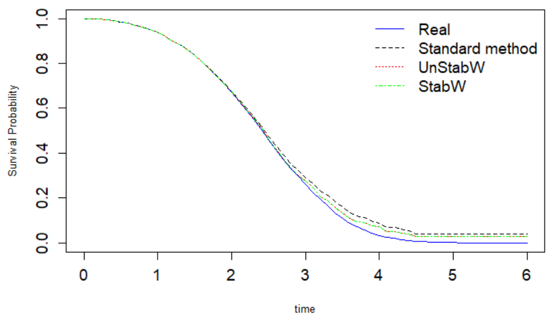
In this section we include simulations with a value of the diagonal matrix Σ equal to 0.1.



(a) $\gamma' = 0.5, \lambda_c = 0.005$



(b) $\gamma' = 1.5, \lambda_c = 0.00004$



(c) $\gamma' = 3, \lambda_c = 0.00000003$

Figure A.8: *The real survival curve (blue) based on uncensored data, the survival curve estimated with the Kaplan-Meier method on censored data (black), and the survival curves estimated with IPCW weights, with stabilized weights (green) and unstabilized weights (red). These are the average curves over 10 simulations with* $(\beta_1, \beta_2) = (0.5, 0.1)$*,* $\lambda_{0x} = 0.01$*,*$\gamma = 0.5$ *and* $(\phi_1, \phi_2) = (0.5, 0.1)$ *for the censoring model. The shape parameter* ν *for both event and censoring times have the same values for all simulations, with* $\nu = 2$*. Censoring percentage is* (33%)*.*

	(a)	(b)	(c)
Standard method	0.0081	0.0134	0.0172
Unstabilized weights	0.0011	0.0033	0.0079
Stabilized weights	0.0011	0.0033	0.0079

Table A.1: Area under the curve for all three simulations scenarios

Appendix B

R Code

B.1 R code chapter 3

```
## Load packages
  library(survival)
  library(ranger)
  library(ggplot2)
  library(dplyr)
  library(survminer)
  library(kableExtra)


## Functions

## simexp function to generate survival and censoring times

simexp<-function (N,data,lamdax,lamdac,beta1,beta2,phi1,phi2)
{

  u1<-runif(N)
  u2<-runif(N)
  Tx<-(-log(u1)/(lamdax*exp(data$x1*beta1+data$x2*beta2))) # event times
  Tc<-(-log(u2)/(lamdac*exp(data$x1*phi1+data$x2*phi2))) # censoring times
  time<-pmin(Tx,Tc)
  status<-as.numeric(Tx<=Tc)
  data.frame(id=1:N,Tx=Tx,Tc=Tc,time=time,status=status,
  x1=data$x1,x2=data$x2)
}

## Simulation iterations

simIterations <- function(# Parameters simulation:
                          tt, # time grid
                          N, # sample size
                          M, # ...
                          # Parameters survival model
                          lamdax, # ... .
                          beta1, # ...
```

```
                              beta2,
                              # Parameters censoring model
                              lamdac,
                              phi1,
                              phi2,
                              data

  )
{

surv.real <- matrix(nrow = M, ncol = length(tt))
surv.cens <- matrix(nrow = M, ncol = length(tt))
# Save censoring percentage in:
cens.perc <- vector(mode = "numeric", length = M)
# Save correlation between X and C in:
correlationXC <- vector(mode = "numeric", length = M)
#save the distance between the curves
area_between_curves<-vector(mode = "numeric", length = M)
#save the max distance between the curves
max_dist<-vector(mode = "numeric", length = M)
index_max<-vector(mode = "numeric", length = M)

for(j in 1:M)
{

  data.sim1<-simexp(N,data,lamdax,lamdac,beta1,beta2,phi1,phi2)
  data.sim2<-data.sim1[data.sim1$status==1,]
  data.sim1$one <- 1
  survtrue <- survfit(Surv(Tx, one) ~ 1, data = data.sim1)
  surv<-survfit(Surv(time,status)~1,data=data.sim1)
  ssf <- summary( survtrue, times=tt,extend = TRUE) ## estimate
  survival probabilities at time points tt
  ssf1<- summary(surv, times=tt,extend = TRUE)
  surv.real[j,]<-ssf$surv
  surv.cens[j,]<-ssf1$surv
  cens.perc[j] <- 1 - sum(data.sim1$status)/N
  correlationXC[j] <- cor(data.sim1$Tx, data.sim1$Tc)
  max_dist[j]<-max(abs(surv.real[j,]-surv.cens[j,]))
  index_max[j]<-which.max(abs(surv.real[j,]-surv.cens[j,]))
  area_between_curves[j]<-(geiger:::.area.between.curves(tt,surv.real[j,],
  surv.cens[j,],xrange = c(0,max(data.sim2$time))))/max(data.sim2$time)
}

return(list(surv.real=surv.real,
              surv.cens=surv.cens,
              cens.perc=cens.perc,
              max_dist=max_dist,
              area_between_curves=area_between_curves,
              correlationXC=correlationXC,
            index_max=index_max
           ))
```

```
}

# Fit models for different scenarios

## Define parameters that should be the same in each scenario
set.seed(123)
M <- 50 # number of Monte Carlo repetitions
N<-500
x1<-sample(c(0,1),size = N,replace=TRUE,prob = c(0.5,0.5)) #  treatment
x2<-round(rnorm(N,mean = 50,sd=10)) #  age
data<-as.data.frame(cbind(x1,x2))
data$x2 <- (data$x2 - 50)/10

# times at which survival prob are calculated
tt <- c(seq(0,5,by=0.1),
        seq(5.25,10,by=0.25),
        seq(10.5,20,by=0.5),
        seq(21,50,by=0.5))

## Scenario 1:
set.seed(123)
scenario1Fit  <- simIterations(M=M, #number of iterations
                               tt=tt, # time grid
                               N=N, # sample size
                               # Parameters survival model
                               lamdax=0.1, # ... .
                               beta1=0.5, # ...
                               beta2=0.1,
                               # Parameters censoring model
                               lamdac=0.027,
                               phi1=1.5,
                               phi2=0.5,
                               data=data

  )

surv.real.MC_scenario1 <- colMeans(scenario1Fit[["surv.real"]])
surv.cens.MC_scenario1 <- colMeans(scenario1Fit[["surv.cens"]])
cens.perc.MC<-mean(scenario1Fit[["cens.perc"]])
cens.perc.MC # percentage of censoring
area_between_curves.MC1<-mean(scenario1Fit[["area_between_curves"]])
max_dist.MC1<-mean(scenario1Fit[["max_dist"]])
area_between_curves.MC1 ## area
max_dist.MC1 ## max distance
correlationXC.MC<-mean(scenario1Fit[["correlationXC"]])
correlationXC.MC ## cor

plot(tt, surv.real.MC_scenario1,
     xlab = "time", ylab = "Survival␣Probability",
     type="l", xlim=c(0,50), ylim=c(0,1), col = "blue",
     cex = 1.2,
     cex.main = 1.2,
     cex.axis = 1.2,
     cex.lab=1.2
)
lines(tt, surv.cens.MC_scenario1, type = "l", col = "red", lwd = 1.5,lty=2)
legend(legend = c("Real",
                  "Standard␣method"),
       col = c("blue", "red"),
       lty = c(1,2),
       x = "topright",
       cex = 1.2,
       bty = "n"
)


## Scenario 2:

set.seed(123)
scenario2Fit  <- simIterations(M=M, #number of iterations
                          tt=tt, # time grid
                          N=N, # sample size
                          # Parameters survival model
                          lamdax=0.1, # ... .
                          beta1=0.5, # ...
                          beta2=0.1,
                          # Parameters censoring model
                          lamdac=0.05,
                          phi1=0.5,
                          phi2=0.1,
                          data=data

  )

surv.real.MC_scenario2 <- colMeans(scenario2Fit[["surv.real"]])
surv.cens.MC_scenario2 <- colMeans(scenario2Fit[["surv.cens"]])
# etc, code below should still be changed:
cens.perc.MC<-mean(scenario2Fit[["cens.perc"]])
cens.perc.MC # percentage of censoring
area_between_curves.MC2<-mean(scenario2Fit[["area_between_curves"]])
max_dist.MC2<-mean(scenario2Fit[["max_dist"]])
area_between_curves.MC2
max_dist.MC2
correlationXC.MC<-mean(scenario2Fit[["correlationXC"]])
correlationXC.MC

plot(tt, surv.real.MC_scenario2,
     xlab = "time", ylab = "Survival␣Probability",
     type="l", xlim=c(0,50), ylim=c(0,1), col = "blue",
     cex = 1.2,
     cex.main = 1.2,
     cex.axis = 1.2,
     cex.lab=0.8
)
lines(tt, surv.cens.MC_scenario2, type = "l", col = "red", lwd = 1.5,lty=2)
legend(legend = c("Real",
                  "Standard␣method"),
       col = c("blue", "red"),
       lty = c(1,2),
       x = "topright",
       cex = 1.2,
       bty = "n"
)



## Scenario 3:

set.seed(123)
scenario3Fit  <- simIterations(M=M, #number of iterations
                         tt=tt, # time grid
                         N=N, # sample size
                         # Parameters survival model
                         lamdax=0.1, # ... .
                         beta1=0.5, # ...
                         beta2=0.1,
                         # Parameters censoring model
                         lamdac=0.06,
                         phi1=0.1,
                         phi2=0,
                         data=data

  )

surv.real.MC_scenario3 <- colMeans(scenario3Fit[["surv.real"]])
surv.cens.MC_scenario3 <- colMeans(scenario3Fit[["surv.cens"]])
cens.perc.MC<-mean(scenario3Fit[["cens.perc"]])
cens.perc.MC # percentage of censoring
area_between_curves.MC3<-mean(scenario3Fit[["area_between_curves"]])
max_dist.MC3<-mean(scenario3Fit[["max_dist"]])
area_between_curves.MC3
max_dist.MC3
correlationXC.MC<-mean(scenario3Fit[["correlationXC"]])
correlationXC.MC

plot(tt, surv.real.MC_scenario3,
     xlab = "time", ylab = "Survival␣Probability",
     type="l", xlim=c(0,50), ylim=c(0,1), col = "blue",
     cex = 1.2,
     cex.main = 1.2,
     cex.axis = 1.2,
     cex.lab=1.2
)
lines(tt, surv.cens.MC_scenario3, type = "l", col = "red", lwd = 1.5,lty=2)
legend(legend = c("Real",
                  "Standard␣method"),
       col = c("blue", "red"),
       lty = c(1,2),
       x = "topright",
       cex = 1.2,
       bty = "n"
)



# Varying the censoring percentage

## Scenario 1:
set.seed(123)
scenario1Fit  <- simIterations(M=M, #number of iterations
                           tt=tt, # time grid
                           N=N, # sample size
                           # Parameters survival model
                           lamdax=0.1, # ... .
                           beta1=0.5, # ...
                           beta2=0.1,
                           # Parameters censoring model
                           lamdac=0.012,
                           phi1=0.5,
                           phi2=0.1,
                           data=data

  )

surv.real.MC_scenario1 <- colMeans(scenario1Fit[["surv.real"]])
surv.cens.MC_scenario1 <- colMeans(scenario1Fit[["surv.cens"]])
# etc, code below should still be changed:
cens.perc.MC<-mean(scenario1Fit[["cens.perc"]])
cens.perc.MC # percentage of censoring
area_between_curves.MC1<-mean(scenario1Fit[["area_between_curves"]])
max_dist.MC1<-mean(scenario1Fit[["max_dist"]])
area_between_curves.MC1 ## area
max_dist.MC1 ## max distance
correlationXC.MC1<-mean(scenario1Fit[["correlationXC"]])
correlationXC.MC1 ## cor

plot(tt, surv.real.MC_scenario1,
     xlab = "time", ylab = "Survival␣Probability",
     type="l", xlim=c(0,50), ylim=c(0,1), col = "blue",
     cex = 1.2,
     cex.main = 1.2,
     cex.axis = 1.2,
     cex.lab=1.2
)
lines(tt, surv.cens.MC_scenario1, type = "l",lty=2,col = "red", lwd = 1.5)
legend(legend = c("Real",
                  "Standard␣method"),
       col = c("blue", "red"),
       lty = c(1,2),
       x = "topright",
       cex = 1.2,
       bty = "n"
)

df <- data.frame(
  id=seq(1,50,by=1),
  time=tt[scenario1Fit[["index_max"]]]
)

ggplot(df, aes(x=time)) +
 geom_histogram(aes(y=..density..),binwidth = 3,colour="black",fill="white")
 geom_density(alpha=.2, fill="#FF6666") +
 labs(title = "Density␣plot␣of␣time␣points␣of␣maximum␣difference")




set.seed(123)
data.sim1<-simexp(500,data=data,0.1,0.012,0.5,0.1,0.5,0.1)
data.sim2<-data.sim1[data.sim1$status==1,]
data.sim1$one <- 1
survtrue <- survfit(Surv(Tx, one) ~ 1, data = data.sim1)
surv<-survfit(Surv(time,status)~1,data=data.sim1)
ssf <- summary( survtrue, times=tt,extend = TRUE)
ssf1<- summary(surv, times=tt,extend = TRUE)
surv.real<-ssf$surv
surv.cens<-ssf1$surv
cens.perc <- 1 - sum(data.sim1$status)/500
correlationXC <- cor(data.sim1$Tx, data.sim1$Tc)
max_dist<-max(abs(surv.real-surv.cens))
area_between_curves<-(geiger:::.area.between.curves(tt,surv.real,surv.cens,
xrange = c(0,50)))/max(data.sim2$time))

plot(survtrue, conf.int=F, xlab = "Time", xlim=c(0,50), ylim=c(0,1),
ylab = "Survival␣probability",col = "blue")
lines(surv, conf.int=T,lty = 2,xlab = "Time",ylab = "Survival␣probability",
col = "red")
legend(legend = c("Real",
                  "Standard␣method"),
       col = c("blue", "red"),
       lty = c(1,2),
       x = "topright",
       cex = 1.2,
       bty = "n"
)


## Scenario 2:
set.seed(123)
scenario2Fit  <- simIterations(M=M, #number of iterations
                           tt=tt, # time grid
                           N=N, # sample size
                           # Parameters survival model
                           lamdax=0.1, # ... .
                           beta1=0.5, # ...
                           beta2=0.1,
                           # Parameters censoring model
                           lamdac=0.045,
                           phi1=0.5,
                           phi2=0.1,
                           data=data

  )

surv.real.MC_scenario2 <- colMeans(scenario2Fit[["surv.real"]])
surv.cens.MC_scenario2 <- colMeans(scenario2Fit[["surv.cens"]])
# etc, code below should still be changed:
cens.perc.MC<-mean(scenario2Fit[["cens.perc"]])
cens.perc.MC # percentage of censoring
area_between_curves.MC2<-mean(scenario2Fit[["area_between_curves"]])
max_dist.MC2<-mean(scenario2Fit[["max_dist"]])
area_between_curves.MC2 ## area
max_dist.MC2 ## max distance
correlationXC.MC2<-mean(scenario2Fit[["correlationXC"]])
correlationXC.MC2 ## cor

plot(tt, surv.real.MC_scenario2,
     xlab = "time", ylab = "Survival␣Probability",
     type="l", xlim=c(0,50), ylim=c(0,1), col = "blue",
     cex = 1.2,
     cex.main = 1.2,
     cex.axis = 1.2,
     cex.lab=0.8
)
lines(tt, surv.cens.MC_scenario2, type = "l",lty=2, col = "red", lwd = 1.5)
legend(legend = c("Real",
                 "Standard␣method"),
       col = c("blue", "red"),
       lty = c(1,2),
       x = "topright",
       cex = 1.2,
       bty = "n"
)
```

```
df <- data.frame(
  id=seq(1,50,by=1),
  time=tt[scenario2Fit[["index_max"]]]
)

ggplot(df, aes(x=time)) +
  geom_histogram(aes(y=..density..),binwidth =3,colour="black",fill="white")
  geom_density(alpha=.2, fill="#FF6666") +
  labs(title = "Density␣plot␣of␣time␣points␣of␣maximum␣difference")


set.seed(123)
data.sim1<-simexp(500,data=data,0.1,0.045,0.5,0.1,0.5,0.1)
data.sim2<-data.sim1[data.sim1$status==1,]
data.sim1$one <- 1
survtrue <- survfit(Surv(Tx, one) ~ 1, data = data.sim1)
surv<-survfit(Surv(time,status)~1,data=data.sim1)
ssf <- summary( survtrue, times=tt,extend = TRUE)
ssf1<- summary(surv, times=tt,extend = TRUE)
surv.real<-ssf$surv
surv.cens<-ssf1$surv
cens.perc <- 1 - sum(data.sim1$status)/500
correlationXC <- cor(data.sim1$Tx, data.sim1$Tc)
max_dist<-max(abs(surv.real-surv.cens))
area_between_curves<-(geiger:::.area.between.curves(tt,surv.real,surv.cens,
xrange = c(0,50)))/50

plot(survtrue, conf.int=F, xlab = "Time", xlim=c(0,50), ylim=c(0,1),
ylab = "Survival␣probability",col = "blue")
lines(surv, conf.int=T,lty = 2, xlab = "Time", ylab = "Survival␣probability"
col = "red")
legend(legend = c("Real",
                  "Standard␣method"),
       col = c("blue", "red"),
       lty = c(1,2),
       x = "topright",
       cex = 1.2,
       bty = "n"
)

## Scenario 3:

set.seed(123)
scenario3Fit  <- simIterations(M=M, #number of iterations
                               tt=tt, # time grid
                               N=N, # sample size
                               # Parameters survival model
                               lamdax=0.1, # ... .
                               beta1=0.5, # ...
                               beta2=0.1,
                          # Parameters censoring model
                          lamdac=0.1,
                          phi1=0.5,
                          phi2=0.1,
                          data=data

  )

surv.real.MC_scenario3 <- colMeans(scenario3Fit[["surv.real"]])
surv.cens.MC_scenario3 <- colMeans(scenario3Fit[["surv.cens"]])
cens.perc.MC<-mean(scenario3Fit[["cens.perc"]])
cens.perc.MC # percentage of censoring
area_between_curves.MC3<-mean(scenario3Fit[["area_between_curves"]])
max_dist.MC3<-mean(scenario3Fit[["max_dist"]])
area_between_curves.MC3 ## area
max_dist.MC3 ## max distance
correlationXC.MC3<-mean(scenario3Fit[["correlationXC"]])
correlationXC.MC3 ## cor

plot(tt, surv.real.MC_scenario3,
     xlab = "time", ylab = "Survival␣Probability",
     type="l", xlim=c(0,50), ylim=c(0,1), col = "blue",
     cex = 1.2,
     cex.main = 1.2,
     cex.axis = 1.2,
     cex.lab=0.8
)
lines(tt, surv.cens.MC_scenario3, type = "l",lty=2, col = "red", lwd = 1.5)
legend(legend = c("Real",
                 "Standard␣method"),
       col = c("blue", "black"),
       lty = c(1,2),
       x = "topright",
       cex = 1.2,
       bty = "n"
)

df <- data.frame(
  id=seq(1,50,by=1),
  time=tt[scenario3Fit[["index_max"]]]
)

ggplot(df, aes(x=time)) +
  geom_histogram(aes(y=..density..),binwidth = 3,colour="black",fill="white"
  geom_density(alpha=.2, fill="#FF6666")+
  labs(title = "Density␣plot␣of␣time␣points␣of␣maximum␣difference")



set.seed(123)
data.sim1<-simexp(500,data=data,0.1,0.1,0.5,0.1,0.5,0.1)
data.sim1$one <- 1
survtrue <- survfit(Surv(Tx, one) ~ 1, data = data.sim1)
surv<-survfit(Surv(time,status)~1,data=data.sim1)
ssf <- summary( survtrue, times=tt,extend = TRUE)
ssf1<- summary(surv, times=tt,extend = TRUE)
surv.real<-ssf$surv
surv.cens<-ssf1$surv
cens.perc <- 1 - sum(data.sim1$status)/500
correlationXC <- cor(data.sim1$Tx, data.sim1$Tc)
max_dist<-max(abs(surv.real-surv.cens))
area_between_curves<-(geiger:::.area.between.curves(tt,surv.real,surv.cens,
xrange = c(0,50)))/50

plot(survtrue, conf.int=F, xlab = "Time", xlim=c(0,50), ylim=c(0,1),
ylab = "Survival␣probability",col = "blue")
lines(surv, conf.int=T,lty = 2, xlab = "Time",ylab = "Survival␣probability",
col = "red")
legend(legend = c("Real",
                  "Standard␣method"),
       col = c("blue", "red"),
       lty = c(1,2),
       x = "topright",
       cex = 1.2,
       bty = "n"
)


# Varying the sample size

## Scenario 1
set.seed(123)
N<-250
x1<-sample(c(0,1),size = N,replace=TRUE,prob = c(0.5,0.5)) # treatment
x2<-round(rnorm(N,mean = 50,sd=10))
data<-as.data.frame(cbind(x1,x2))
data$x2 <- (data$x2 - 50)/10
scenario1Fit  <- simIterations(M=M, #number of iterations
                              tt=tt, # time grid
                              N=N, # sample size
                              # Parameters survival model
                              lamdax=0.1, # ... .
                              beta1=0.5, # ...
                              beta2=0.1,
                              # Parameters censoring model
                              lamdac=0.05,
                              phi1=0.5,
                              phi2=0.1,
                              data=data

  )
```

```
surv.real.MC_scenario1 <- colMeans(scenario1Fit[["surv.real"]])
surv.cens.MC_scenario1 <- colMeans(scenario1Fit[["surv.cens"]])
cens.perc.MC<-mean(scenario1Fit[["cens.perc"]])
cens.perc.MC # percentage of censoring
area_between_curves.MC1<-mean(scenario1Fit[["area_between_curves"]])
max_dist.MC1<-mean(scenario1Fit[["max_dist"]])
area_between_curves.MC1 ## area
max_dist.MC1 ## max distance
correlationXC.MC1<-mean(scenario1Fit[["correlationXC"]])
correlationXC.MC1 ## cor

plot(tt, surv.real.MC_scenario1,
     xlab = "time", ylab = "Survival␣Probability",
     type="l", xlim=c(0,50), ylim=c(0,1), col = "blue",
     cex = 1.2,
     cex.main = 1.2,
     cex.axis = 1.2,
     cex.lab=1.2
)
lines(tt, surv.cens.MC_scenario1, type = "l",lty=2, col = "red", lwd = 1.5)
legend(legend = c("Real",
                  "Standard␣method"),
       col = c("blue", "red"),
       lty = c(1,2),
       x = "topright",
       cex = 1.2,
       bty = "n"
)

df <- data.frame(
  id=seq(1,50,by=1),
  time=tt[scenario1Fit[["index_max"]]]
)

ggplot(df, aes(x=time)) +
  geom_histogram(aes(y=..density..),binwidth = 3,colour="black",fill="white"
  geom_density(alpha=.2, fill="#FF6666")+
  labs(title = "Density␣plot␣of␣time␣points␣of␣maximum␣difference")



set.seed(123)
N<-250
x2<-sample(c(0,1),size = N,replace=TRUE,prob = c(0.5,0.5)) # treatment
x1<-round(rnorm(N,mean = 50,sd=10)) # age
data<-as.data.frame(cbind(x1,x2))
data$x1 <- (data$x1 - 50)/10
data.sim1<-simexp(250,data=data,0.1,0.05,0.5,0.1,0.5,0.1)
data.sim1$one <- 1
survtrue <- survfit(Surv(Tx, one) ~ 1, data = data.sim1)
surv<-survfit(Surv(time,status)~1,data=data.sim1)
```

```
ssf <- summary( survtrue, times=tt,extend = TRUE)
ssf1<- summary(surv, times=tt,extend = TRUE)
surv.real<-ssf$surv
surv.cens<-ssf1$surv
cens.perc <- 1 - sum(data.sim1$status)/250
correlationXC <- cor(data.sim1$Tx, data.sim1$Tc)
max_dist<-max(abs(surv.real-surv.cens))
area_between_curves<-(geiger:::.area.between.curves(tt,surv.real,surv.cens,
xrange = c(0,50)))/50

plot(survtrue, conf.int=F, xlab = "Time", xlim=c(0,50), ylim=c(0,1),
ylab = "Survival␣probability",col = "blue")
lines(surv, conf.int=T,lty = 2, xlab = "Time", ylab = "Survival␣probability"
col = "red")
legend(legend = c("Real",
                  "Standard␣method"),
       col = c("blue", "red"),
       lty = c(1,2),
       x = "topright",
       cex = 1.2,
       bty = "n"
)


## Scenario 2:
set.seed(123)
N<-500
x1<-sample(c(0,1),size = N,replace=TRUE,prob = c(0.5,0.5)) # treatment
x2<-round(rnorm(N,mean = 50,sd=10)) # age
data<-as.data.frame(cbind(x1,x2))
data$x2 <- (data$x2 - 50)/10
scenario2Fit  <- simIterations(M=M, #number of iterations
                          tt=tt, # time grid
                          N=N, # sample size
                          # Parameters survival model
                          lamdax=0.1, # ... .
                          beta1=0.5, # ...
                          beta2=0.1,
                          # Parameters censoring model
                          lamdac=0.05,
                          phi1=0.5,
                          phi2=0.1,
                          data=data

  )

surv.real.MC_scenario2 <- colMeans(scenario2Fit[["surv.real"]])
surv.cens.MC_scenario2 <- colMeans(scenario2Fit[["surv.cens"]])
cens.perc.MC<-mean(scenario2Fit[["cens.perc"]])
cens.perc.MC # percentage of censoring
area_between_curves.MC2<-mean(scenario2Fit[["area_between_curves"]])
```

```
max_dist.MC2<-mean(scenario2Fit[["max_dist"]])
area_between_curves.MC2 ## area
max_dist.MC2 ## max distance
correlationXC.MC2<-mean(scenario2Fit[["correlationXC"]])
correlationXC.MC2 ## cor

plot(tt, surv.real.MC_scenario2,
     xlab = "time", ylab = "Survival␣Probability",
     type="l", xlim=c(0,50), ylim=c(0,1), col = "blue",
     cex = 1.2,
     cex.main = 1.2,
     cex.axis = 1.2,
     cex.lab=0.8
)
lines(tt, surv.cens.MC_scenario2, type = "l",lty=2, col = "red", lwd = 1.5)
legend(legend = c("Real",
                 "Standard␣method"),
       col = c("blue", "red"),
       lty = c(1,2),
       x = "topright",
       cex = 1.2,
       bty = "n"
)

df <- data.frame(
  id=seq(1,50,by=1),
  time=tt[scenario2Fit[["index_max"]]]
)

ggplot(df, aes(x=time)) +
  geom_histogram(aes(y=..density..),binwidth = 3,colour="black",fill="white"
  geom_density(alpha=.2, fill="#FF6666")+
  labs(title = "Density␣plot␣of␣time␣points␣of␣maximum␣difference")




set.seed(123)
N<-500
x1<-sample(c(0,1),size = N,replace=TRUE,prob = c(0.5,0.5)) # treatment
x2<-round(rnorm(N,mean = 50,sd=10)) # age
data<-as.data.frame(cbind(x1,x2))
data$x2 <- (data$x2 - 50)/10
data.sim1<-simexp(500,data=data,0.1,0.05,0.5,0.1,0.5,0.1)
data.sim1$one <- 1
survtrue <- survfit(Surv(Tx, one) ~ 1, data = data.sim1)
surv<-survfit(Surv(time,status)~1,data=data.sim1)
ssf <- summary( survtrue, times=tt,extend = TRUE)
ssf1<- summary(surv, times=tt,extend = TRUE)
surv.real<-ssf$surv
surv.cens<-ssf1$surv
```

```
cens.perc <- 1 - sum(data.sim1$status)/500
correlationXC <- cor(data.sim1$Tx, data.sim1$Tc)
max_dist<-max(abs(surv.real-surv.cens))
area_between_curves<-(geiger:::.area.between.curves(tt,surv.real,surv.cens,
xrange = c(0,50)))/50

plot(survtrue, conf.int=F, xlab = "Time", xlim=c(0,50), ylim=c(0,1),
ylab = "Survival␣probability",col = "blue")
lines(surv, conf.int=T,lty = 2, xlab = "Time", ylab = "Survival␣probability"
col = "red")
legend(legend = c("Real",
                  "Standard␣method"),
       col = c("blue", "red"),
       lty = c(1,2),
       x = "topright",
       cex = 1.2,
       bty = "n"
)


## Scenario 3:
set.seed(123)
N<-1000
x1<-sample(c(0,1),size = N,replace=TRUE,prob = c(0.5,0.5)) #  treatment
x2<-round(rnorm(N,mean = 50,sd=10)) #age
data<-as.data.frame(cbind(x1,x2))
data$x2 <- (data$x2 - 50)/10
scenario3Fit  <- simIterations(M=M, #number of iterations
                            tt=tt, # time grid
                            N=N, # sample size
                            # Parameters survival model
                            lamdax=0.1, # ... .
                            beta1=0.5, # ...
                            beta2=0.1,
                            # Parameters censoring model
                            lamdac=0.05,
                            phi1=0.5,
                            phi2=0.1,
                            data=data

  )

surv.real.MC_scenario3 <- colMeans(scenario3Fit[["surv.real"]])
surv.cens.MC_scenario3 <- colMeans(scenario3Fit[["surv.cens"]])
cens.perc.MC<-mean(scenario3Fit[["cens.perc"]])
cens.perc.MC # percentage of censoring
area_between_curves.MC<-mean(scenario3Fit[["area_between_curves"]])
max_dist.MC<-mean(scenario3Fit[["max_dist"]])
area_between_curves.MC ## area
max_dist.MC ## max distance
correlationXC.MC<-mean(scenario3Fit[["correlationXC"]])
correlationXC.MC ## cor

plot(tt, surv.real.MC_scenario3,
     xlab = "time", ylab = "Survival␣Probability",
     type="l", xlim=c(0,50), ylim=c(0,1), col = "blue",
     cex = 1.2,
     cex.main = 1.2,
     cex.axis = 1.2,
     cex.lab=0.8
)
lines(tt, surv.cens.MC_scenario3, type = "l",lty=2, col = "red", lwd = 1.5)
legend(legend = c("Real",
                  "Standard␣method"),
       col = c("blue", "red"),
       lty = c(1,2),
       x = "topright",
       cex = 1.2,
       bty = "n"
)

df <- data.frame(
  id=seq(1,50,by=1),
  time=tt[scenario1Fit[["index_max"]]]
)

ggplot(df, aes(x=time)) +
  geom_histogram(aes(y=..density..),binwidth = 3,colour="black",fill="white"
  geom_density(alpha=.2, fill="#FF6666")+
  labs(title = "Density␣plot␣of␣time␣points␣of␣maximum␣difference")



set.seed(123)
N<-1000
x1<-sample(c(0,1),size = N,replace=TRUE,prob = c(0.5,0.5)) # simulate sex
x2<-round(rnorm(N,mean = 50,sd=10))
data<-as.data.frame(cbind(x1,x2))
data$x2 <- (data$x2 - 50)/10
data.sim1<-simexp(1000,data=data,0.1,0.05,0.5,0.1,0.5,0.1)
data.sim1$one <- 1
survtrue <- survfit(Surv(Tx, one) ~ 1, data = data.sim1)
surv<-survfit(Surv(time,status)~1,data=data.sim1)
ssf <- summary( survtrue, times=tt,extend = TRUE)
ssf1<- summary(surv, times=tt,extend = TRUE)
surv.real<-ssf$surv
surv.cens<-ssf1$surv
cens.perc <- 1 - sum(data.sim1$status)/1000
correlationXC <- cor(data.sim1$Tx, data.sim1$Tc)
max_dist<-max(abs(surv.real-surv.cens))
area_between_curves<-(geiger:::.area.between.curves(tt,surv.real,surv.cens,
xrange = c(0,50)))/50)
```

```
plot(survtrue, conf.int=F, xlab = "Time", xlim=c(0,50), ylim=c(0,1),
ylab = "Survival␣probability",col = "blue")
lines(surv, conf.int=T,lty = 2, xlab = "Time", ylab = "Survival␣probability"
col = "red")
legend(legend = c("Real",
                  "Standard␣method"),
       col = c("blue", "red"),
       lty = c(1,2),
       x = "topright",
       cex = 1.2,
       bty = "n"
)



## second method for weibull

# Functions

## simweib function to generate event and censoring times

simweib<-function (N,data,lamdax,lamdac,beta1,beta2,phi1,phi2,nx,nc)
{

  u1<-runif(N)
  u2<-runif(N)
  Tx<-(-log(u1)/(lamdax*exp(data$x1*beta1+data$x2*beta2)))^(1/nx)
  Tc<-(-log(u2)/(lamdac*exp(data$x1*phi1+data$x2*phi2)))^(1/nc)
  time<-pmin(Tx,Tc)
  status<-as.numeric(Tx<=Tc)
  data.frame(id=1:N,Tx=Tx,Tc=Tc,time=time,status=status,x1=data$x1,
  x2=data$x2)
}



## Simulation iterations
simIterations <- function(# Parameters simulation:
                         M, #number of iterations
                         tt, # time grid
                         N, # sample size
                         # Parameters survival model
                         lamdax, # ... .
                         beta1, # ...
                         beta2,
                         nx,
                         # Parameters censoring model
                         lamdac,
                         phi1,
                         phi2,
                          nc,
                          data

  )
{

surv.real <- matrix(nrow = M, ncol = length(tt))
surv.cens <- matrix(nrow = M, ncol = length(tt))
# Save censoring percentage in:
cens.perc <- vector(mode = "numeric", length = M)
# Save correlation between X and C in:
correlationXC <- vector(mode = "numeric", length = M)
#save the distance between the curves
area_between_curves<-vector(mode = "numeric", length = M)
#save the max distance between the curves
max_dist<-vector(mode = "numeric", length = M)
# index of max
index_max<-vector(mode = "numeric", length = M)

for(j in 1:M)
{

  data.sim1<-simweib(N,data,lamdax,lamdac,beta1,beta2,phi1,phi2,nx,nc)
  data.sim2<-data.sim1[data.sim1$status==1,]
  data.sim1$one <- 1
  survtrue <- survfit(Surv(Tx, one) ~ 1, data = data.sim1)
  surv<-survfit(Surv(time,status)~1,data=data.sim1)
  ssf <- summary( survtrue, times=tt,extend = TRUE)
  ssf1<- summary(surv, times=tt,extend = TRUE)
  surv.real[j,]<-ssf$surv
  surv.cens[j,]<-ssf1$surv
  cens.perc[j] <- 1 - sum(data.sim1$status)/N
  correlationXC[j] <- cor(data.sim1$Tx, data.sim1$Tc)
  max_dist[j]<-max(abs(surv.real[j,]-surv.cens[j,]))
  index_max[j]<-which.max(abs(surv.real[j,]-surv.cens[j,]))
area_between_curves[j]<-(geiger:::.area.between.curves(tt,surv.real[j,],
surv.cens[j,], xrange = c(0,max(data.sim2$time))))/max(data.sim2$time)

}
return(list(surv.real=surv.real,
              surv.cens=surv.cens,
              cens.perc=cens.perc,
              max_dist=max_dist,
              area_between_curves=area_between_curves,
              correlationXC=correlationXC,
            index_max=index_max))


}
```

```
# Fit models for different scenarios

## Define parameters that should be the same in each scenario
set.seed(123)
M <- 50 # number of Monte Carlo repetitions
N<-500
x1<-sample(c(0,1),size = N,replace=TRUE,prob = c(0.5,0.5)) # simulate treatm
x2<-round(rnorm(N,mean = 50,sd=10)) # simulate age
data<-as.data.frame(cbind(x1,x2))
data$x2 <- (data$x2 - 50)/10

# times at which survival prob are calculated
tt <- c(seq(0,10,by=0.1))

## Scenario 1:
set.seed(123)
scenario1Fit  <- simIterations(M=M, #number of iterations
                              tt=tt, # time grid
                              N=N, # sample size
                              # Parameters survival model
                              lamdax=0.1, # ... .
                              beta1=0.5, # ...
                              beta2=0.1,
                              nx=2,
                              # Parameters censoring model
                              lamdac=0.027,
                              phi1=1.5,
                              phi2=0.5,
                              nc=2,
                              data=data

  )

surv.real.MC_scenario <- colMeans(scenario1Fit[["surv.real"]])
surv.cens.MC_scenario<- colMeans(scenario1Fit[["surv.cens"]])
cens.perc.MC<-mean(scenario1Fit[["cens.perc"]])
cens.perc.MC # percentage of censoring
area_between_curves.MC<-mean(scenario1Fit[["area_between_curves"]])
max_dist.MC<-mean(scenario1Fit[["max_dist"]])
area_between_curves.MC ## area
max_dist.MC ## max distance
correlationXC.MC<-mean(scenario1Fit[["correlationXC"]])
correlationXC.MC ## cor

plot(tt, surv.real.MC_scenario,
     xlab = "time", ylab = "Survival␣Probability",
     type="l", xlim=c(0,10), ylim=c(0,1), col = "blue",
     cex = 1.2,
     cex.main = 1.2,
     cex.axis = 1.2,
     cex.lab=1.2
)
lines(tt, surv.cens.MC_scenario, type = "l",lty=2,col = "red", lwd = 1.5)
legend(legend = c("Real",
                  "Standard␣method"),
       col = c("blue", "red"),
       lty = c(1,2),
       x = "topright",
       cex = 1.2,
       bty = "n"
)

df <- data.frame(
  id=seq(1,50,by=1),
  time=tt[scenario1Fit[["index_max"]]]
)

ggplot(df, aes(x=time)) +
  geom_histogram(aes(y=..density..),binwidth = 0.5,colour="black",fill="whit
  geom_density(alpha=.2, fill="#FF6666") +
  labs(title = "Density␣plot␣of␣time␣points␣of␣maximum␣difference")




## Scenario 2:
set.seed(123)
scenario2Fit  <- simIterations(M=M, #number of iterations
                         tt=tt, # time grid
                         N=N, # sample size
                         # Parameters survival model
                         lamdax=0.1, # ... .
                         beta1=0.5, # ...
                         beta2=0.1,
                         nx=2,
                         # Parameters censoring model
                         lamdac=0.05,
                         phi1=0.5,
                         phi2=0.1,
                         nc=2,
                         data=data

  )

surv.real.MC_scenario <- colMeans(scenario2Fit[["surv.real"]])
surv.cens.MC_scenario<- colMeans(scenario2Fit[["surv.cens"]])
cens.perc.MC<-mean(scenario2Fit[["cens.perc"]])
cens.perc.MC # percentage of censoring
area_between_curves.MC<-mean(scenario2Fit[["area_between_curves"]])
max_dist.MC<-mean(scenario2Fit[["max_dist"]])
```

```
area_between_curves.MC ## area
max_dist.MC ## max distance
correlationXC.MC<-mean(scenario2Fit[["correlationXC"]])
correlationXC.MC ## cor

plot(tt, surv.real.MC_scenario,
     xlab = "time", ylab = "Survival␣Probability",
     type="l", xlim=c(0,10), ylim=c(0,1), col = "blue",
     cex = 1.2,
     cex.main = 1.2,
     cex.axis = 1.2,
     cex.lab=1.2
)
lines(tt, surv.cens.MC_scenario, type = "l", col = "red", lwd = 1.5,lty=2)
legend(legend = c("Real",
                  "Standard␣method"),
       col = c("blue", "red"),
       lty = c(1,2),
       x = "topright",
       cex = 1.2,
       bty = "n"
)


## Scenario 3:
set.seed(123)
scenario3Fit  <- simIterations(M=M, #number of iterations
                          tt=tt, # time grid
                          N=N, # sample size
                          # Parameters survival model
                          lamdax=0.1, # ... .
                          beta1=0.5, # ...
                          beta2=0.1,
                          nx=2,
                          # Parameters censoring model
                          lamdac=0.06,
                          phi1=0.1,
                          phi2=0,
                          nc=2,
                          data=data

  )

surv.real.MC_scenario <- colMeans(scenario3Fit[["surv.real"]])
surv.cens.MC_scenario<- colMeans(scenario3Fit[["surv.cens"]])
cens.perc.MC<-mean(scenario3Fit[["cens.perc"]])
cens.perc.MC # percentage of censoring
area_between_curves.MC<-mean(scenario3Fit[["area_between_curves"]])
max_dist.MC<-mean(scenario3Fit[["max_dist"]])
area_between_curves.MC ## area
max_dist.MC ## max distance
```

```
correlationXC.MC<-mean(scenario3Fit[["correlationXC"]])
correlationXC.MC ## cor

plot(tt, surv.real.MC_scenario,
     xlab = "time", ylab = "Survival␣Probability",
     type="l", xlim=c(0,10), ylim=c(0,1), col = "blue",
     cex = 1.2,
     cex.main = 1.2,
     cex.axis = 1.2,
     cex.lab=1.2
)
lines(tt, surv.cens.MC_scenario, type = "l", col = "red", lwd = 1.5,lty=2)
legend(legend = c("Real",
                  "Standard␣method"),
       col = c("blue", "red"),
       lty = c(1,2),
       x = "topright",
       cex = 1.2,
       bty = "n"
)


# Varying the censoring percentage

## Scenario 1:
set.seed(123)
scenario1Fit  <- simIterations(M=M, #number of iterations
                         tt=tt, # time grid
                         N=N, # sample size
                         # Parameters survival model
                         lamdax=0.1, # ... .
                         beta1=0.5, # ...
                         beta2=0.1,
                         nx=2,
                         # Parameters censoring model
                         lamdac=0.012,
                         phi1=0.5,
                         phi2=0.1,
                         nc=2,
                         data=data

  )

surv.real.MC_scenario <- colMeans(scenario1Fit[["surv.real"]])
surv.cens.MC_scenario<- colMeans(scenario1Fit[["surv.cens"]])
cens.perc.MC<-mean(scenario1Fit[["cens.perc"]])
cens.perc.MC # percentage of censoring
area_between_curves.MC<-mean(scenario1Fit[["area_between_curves"]])
max_dist.MC<-mean(scenario1Fit[["max_dist"]])
area_between_curves.MC ## area
max_dist.MC ## max distance
```

```
correlationXC.MC<-mean(scenario1Fit[["correlationXC"]])
correlationXC.MC ## cor

plot(tt, surv.real.MC_scenario,
     xlab = "time", ylab = "Survival␣Probability",
     type="l", xlim=c(0,10), ylim=c(0,1), col = "blue",
     cex = 1.2,
     cex.main = 1.2,
     cex.axis = 1.2,
     cex.lab=1.2
)
lines(tt, surv.cens.MC_scenario, type = "l",col = "red", lwd = 1.5,lty=2)
legend(legend = c("Real",
                  "Standard␣method"),
       col = c("blue", "red"),
       lty = c(1,2),
       x = "topright",
       cex = 1.2,
       bty = "n"
)

df <- data.frame(
  id=seq(1,50,by=1),
  time=tt[scenario1Fit[["index_max"]]]
)

ggplot(df, aes(x=time)) +
 geom_histogram(aes(y=..density..),binwidth = 0.5,colour="black",fill="white
 geom_density(alpha=.2, fill="#FF6666") +
 labs(title = "Density␣plot␣of␣time␣points␣of␣maximum␣difference")



set.seed(123)
data.sim1<-simweib(500,data=data,0.1,0.012,0.5,0.1,0.5,0.1,2,2)
data.sim1$one <- 1
survtrue <- survfit(Surv(Tx, one) ~ 1, data = data.sim1)
surv<-survfit(Surv(time,status)~1,data=data.sim1)
ssf <- summary( survtrue, times=tt,extend = TRUE)
ssf1<- summary(surv, times=tt,extend = TRUE)
surv.real<-ssf$surv
surv.cens<-ssf1$surv
cens.perc <- 1 - sum(data.sim1$status)/500
correlationXC <- cor(data.sim1$Tx, data.sim1$Tc)
max_dist<-max(abs(surv.real-surv.cens))
area_between_curves<-(geiger:::.area.between.curves(tt,surv.real,surv.cens,
xrange = c(0,max(tt))))/max(tt)

plot(survtrue, conf.int=F, xlab = "Time", xlim=c(0,8), ylim=c(0,1),
ylab = "Survival␣probability",col = "blue")
lines(surv, conf.int=T,lty = 2, xlab = "Time", ylab = "Survival␣probability"
col = "red")
legend(legend = c("Real",
                  "Standard␣method"),
       col = c("blue", "red"),
       lty = c(1,2),
       x = "topright",
       cex = 1.2,
       bty = "n"
)



## Scenario 2:
set.seed(123)
scenario2Fit  <- simIterations(M=M, #number of iterations
                          tt=tt, # time grid
                          N=N, # sample size
                          # Parameters survival model
                          lamdax=0.1, # ... .
                          beta1=0.5, # ...
                          beta2=0.1,
                          nx=2,
                          # Parameters censoring model
                          lamdac=0.045,
                          phi1=0.5,
                          phi2=0.1,
                          nc=2,
                          data=data

  )

surv.real.MC_scenario <- colMeans(scenario2Fit[["surv.real"]])
surv.cens.MC_scenario<- colMeans(scenario2Fit[["surv.cens"]])
cens.perc.MC<-mean(scenario2Fit[["cens.perc"]])
cens.perc.MC # percentage of censoring
area_between_curves.MC<-mean(scenario2Fit[["area_between_curves"]])
max_dist.MC<-mean(scenario2Fit[["max_dist"]])
area_between_curves.MC ## area
max_dist.MC ## max distance
correlationXC.MC<-mean(scenario2Fit[["correlationXC"]])
correlationXC.MC ## cor

plot(tt, surv.real.MC_scenario,
     xlab = "time", ylab = "Survival␣Probability",
     type="l", xlim=c(0,10), ylim=c(0,1), col = "blue",
     cex = 1.2,
     cex.main = 1.2,
     cex.axis = 1.2,
     cex.lab=1.2
)
lines(tt, surv.cens.MC_scenario, type = "l",col = "red", lwd = 1.5,lty=2)
```

```
legend(legend = c("Real",
                  "Standard␣method"),
       col = c("blue", "red"),
       lty = c(1,2),
       x = "topright",
       cex = 1.2,
       bty = "n"
)

df <- data.frame(
  id=seq(1,50,by=1),
  time=tt[scenario2Fit[["index_max"]]]
)

ggplot(df, aes(x=time)) +
 geom_histogram(aes(y=..density..),binwidth = 0.5,colour="black",fill="white
 geom_density(alpha=.2, fill="#FF6666") +
  abs(title = "Density␣plot␣of␣time␣points␣of␣maximum␣difference")




set.seed(123)
data.sim1<-simweib(500,data=data,0.1,0.045,0.5,0.1,0.5,0.1,2,2)
data.sim1$one <- 1
survtrue <- survfit(Surv(Tx, one) ~ 1, data = data.sim1)
surv<-survfit(Surv(time,status)~1,data=data.sim1)
ssf <- summary( survtrue, times=tt,extend = TRUE)
ssf1<- summary(surv, times=tt,extend = TRUE)
surv.real<-ssf$surv
surv.cens<-ssf1$surv
cens.perc <- 1 - sum(data.sim1$status)/500
correlationXC <- cor(data.sim1$Tx, data.sim1$Tc)
max_dist<-max(abs(surv.real-surv.cens))
area_between_curves<-(geiger:::.area.between.curves(tt,surv.real,surv.cens,
xrange = c(0,max(tt))))/max(tt))

plot(survtrue, conf.int=F, xlab = "Time", xlim=c(0,8), ylim=c(0,1),
ylab = "Survival␣probability",col = "blue")
lines(surv, conf.int=T,lty = 2, xlab = "Time", ylab = "Survival␣probability"
col = "red")
legend(legend = c("Real",
                  "Standard␣method"),
       col = c("blue", "red"),
       lty = c(1,2),
       x = "topright",
       cex = 1.2,
       bty = "n"
)
```

```
## Scenario 3:
set.seed(123)
scenario3Fit  <- simIterations(M=M, #number of iterations
                         tt=tt, # time grid
                         N=N, # sample size
                         # Parameters survival model
                         lamdax=0.1, # ... .
                         beta1=0.5, # ...
                         beta2=0.1,
                         nx=2,
                         # Parameters censoring model
                         lamdac=0.1,
                         phi1=0.5,
                         phi2=0.1,
                         nc=2,
                         data=data

  )

surv.real.MC_scenario <- colMeans(scenario3Fit[["surv.real"]])
surv.cens.MC_scenario<- colMeans(scenario3Fit[["surv.cens"]])
cens.perc.MC<-mean(scenario3Fit[["cens.perc"]])
cens.perc.MC # percentage of censoring
area_between_curves.MC<-mean(scenario3Fit[["area_between_curves"]])
max_dist.MC<-mean(scenario3Fit[["max_dist"]])
area_between_curves.MC ## area
max_dist.MC ## max distance
correlationXC.MC<-mean(scenario3Fit[["correlationXC"]])
correlationXC.MC ## cor

plot(tt, surv.real.MC_scenario,
     xlab = "time", ylab = "Survival␣Probability",
     type="l", xlim=c(0,10), ylim=c(0,1), col = "blue",
     cex = 1.2,
     cex.main = 1.2,
     cex.axis = 1.2,
     cex.lab=1.2
)
lines(tt, surv.cens.MC_scenario, type = "l",col = "red", lwd = 1.5,lty=2)
legend(legend = c("Real",
                  "Standard␣method"),
       col = c("blue", "black"),
       lty = c(1,2),
       x = "topright",
       cex = 1.2,
       bty = "n"
)

df <- data.frame(
  id=seq(1,50,by=1),
  time=tt[scenario3Fit[["index_max"]]]
```

```
)

ggplot(df, aes(x=time)) +
 geom_histogram(aes(y=..density..),binwidth = 0.5,colour="black",fill="white
 geom_density(alpha=.2, fill="#FF6666") +
 labs(title = "Density␣plot␣of␣time␣points␣of␣maximum␣difference")



set.seed(123)
data.sim1<-simweib(500,data=data,0.1,0.1,0.5,0.1,0.5,0.1,2,2)
data.sim1$one <- 1
survtrue <- survfit(Surv(Tx, one) ~ 1, data = data.sim1)
surv<-survfit(Surv(time,status)~1,data=data.sim1)
ssf <- summary( survtrue, times=tt,extend = TRUE)
ssf1<- summary(surv, times=tt,extend = TRUE)
surv.real<-ssf$surv
surv.cens<-ssf1$surv
cens.perc <- 1 - sum(data.sim1$status)/500
correlationXC <- cor(data.sim1$Tx, data.sim1$Tc)
max_dist<-max(abs(surv.real-surv.cens))
area_between_curves<-(geiger:::.area.between.curves(tt,surv.real,surv.cens,
xrange = c(0,max(tt))))/max(tt)

plot(survtrue, conf.int=F, xlab = "Time", xlim=c(0,8), ylim=c(0,1),
ylab = "Survival␣probability",col = "blue")
lines(surv, conf.int=T,lty = 2, xlab = "Time", ylab = "Survival␣probability"
col = "red")
legend(legend = c("Real",
                  "Standard␣method"),
       col = c("blue", "red"),
       lty = c(1,2),
       x = "topright",
       cex = 1.2,
       bty = "n"
)


# Varying the sample size

## Scenario 1:
set.seed(123)
N<-250
x1<-sample(c(0,1),size = N,replace=TRUE,prob = c(0.5,0.5)) #  treatment
x2<-round(rnorm(N,mean = 50,sd=10)) # age
data<-as.data.frame(cbind(x1,x2))
data$x2 <- (data$x2 - 50)/10
scenario1Fit  <- simIterations(M=M, #number of iterations
                         tt=tt, # time grid
                         N=N, # sample size
                         # Parameters survival model
```

```
                            lamdax=0.1, # ... .
                            beta1=0.5, # ...
                            beta2=0.1,
                            nx=2,
                            # Parameters censoring model
                            lamdac=0.05,
                            phi1=0.5,
                            phi2=0.1,
                            nc=2,
                            data=data

  )

surv.real.MC_scenario <- colMeans(scenario1Fit[["surv.real"]])
surv.cens.MC_scenario <- colMeans(scenario1Fit[["surv.cens"]])
cens.perc.MC<-mean(scenario1Fit[["cens.perc"]])
cens.perc.MC # percentage of censoring
area_between_curves.MC<-mean(scenario1Fit[["area_between_curves"]])
max_dist.MC<-mean(scenario1Fit[["max_dist"]])
area_between_curves.MC ## area
max_dist.MC ## max distance
correlationXC.MC<-mean(scenario1Fit[["correlationXC"]])
correlationXC.MC ## cor

plot(tt, surv.real.MC_scenario,
     xlab = "time", ylab = "Survival␣Probability",
     type="l", xlim=c(0,10), ylim=c(0,1), col = "blue",
     cex = 1.2,
     cex.main = 1.2,
     cex.axis = 1.2,
     cex.lab=1.2
)
lines(tt, surv.cens.MC_scenario, type = "l",col = "red", lwd = 1.5,lty=2)
legend(legend = c("Real",
                  "Standard␣method"),
       col = c("blue", "black"),
       lty = c(1,2),
       x = "topright",
       cex = 1.2,
       bty = "n"
)

df <- data.frame(
  id=seq(1,50,by=1),
  time=tt[scenario1Fit[["index_max"]]]
)

ggplot(df, aes(x=time)) +
 geom_histogram(aes(y=..density..),binwidth = 0.5,colour="black",fill="white
 geom_density(alpha=.2, fill="#FF6666") +
 labs(title = "Density␣plot␣of␣time␣points␣of␣maximum␣difference")
```

```
set.seed(123)
N<-250
x1<-sample(c(0,1),size = N,replace=TRUE,prob = c(0.5,0.5)) #  treatment
x2<-round(rnorm(N,mean = 50,sd=10)) # age
data<-as.data.frame(cbind(x1,x2))
data$x2 <- (data$x2 - 50)/10
data.sim1<-simweib(250,data=data,0.1,0.05,0.5,0.1,0.5,0.1,2,2)
data.sim1$one <- 1
survtrue <- survfit(Surv(Tx, one) ~ 1, data = data.sim1)
surv<-survfit(Surv(time,status)~1,data=data.sim1)
ssf <- summary( survtrue, times=tt,extend = TRUE)
ssf1<- summary(surv, times=tt,extend = TRUE)
surv.real<-ssf$surv
surv.cens<-ssf1$surv
cens.perc <- 1 - sum(data.sim1$status)/250
correlationXC <- cor(data.sim1$Tx, data.sim1$Tc)
max_dist<-max(abs(surv.real-surv.cens))
area_between_curves<-(geiger:::.area.between.curves(tt,surv.real,surv.cens,
xrange = c(0,max(tt))))/max(tt)

plot(survtrue, conf.int=F, xlab = "Time", xlim=c(0,8), ylim=c(0,1),
ylab = "Survival␣probability",col = "blue")
lines(surv, conf.int=T,lty = 2, xlab = "Time", ylab = "Survival␣probability"
col = "red")
legend(legend = c("Real",
                  "Standard␣method"),
       col = c("blue", "red"),
       lty = c(1,2),
       x = "topright",
       cex = 1.2,
       bty = "n"
)



## Scenario 2:
set.seed(123)
N<-500
x1<-sample(c(0,1),size = N,replace=TRUE,prob = c(0.5,0.5)) # treatment
x2<-round(rnorm(N,mean = 50,sd=10)) # age
data<-as.data.frame(cbind(x1,x2))
data$x2 <- (data$x2 - 50)/10
scenario2Fit  <- simIterations(M=M, #number of iterations
                         tt=tt, # time grid
                         N=N, # sample size
                         # Parameters survival model
                         lamdax=0.1, # ... .
```

```
                                beta1=0.5, # ...
                                beta2=0.1,
                                nx=2,
                                # Parameters censoring model
                                lamdac=0.05,
                                phi1=0.5,
                                phi2=0.1,
                                nc=2,
                                data=data

  )

surv.real.MC_scenario <- colMeans(scenario2Fit[["surv.real"]])
surv.cens.MC_scenario <- colMeans(scenario2Fit[["surv.cens"]])
cens.perc.MC<-mean(scenario2Fit[["cens.perc"]])
cens.perc.MC # percentage of censoring
area_between_curves.MC<-mean(scenario2Fit[["area_between_curves"]])
max_dist.MC<-mean(scenario2Fit[["max_dist"]])
area_between_curves.MC ## area
max_dist.MC ## max distance
correlationXC.MC<-mean(scenario2Fit[["correlationXC"]])
correlationXC.MC ## cor

plot(tt, surv.real.MC_scenario,
     xlab = "time", ylab = "Survival␣Probability",
     type="l", xlim=c(0,10), ylim=c(0,1), col = "blue",
     cex = 1.2,
     cex.main = 1.2,
     cex.axis = 1.2,
     cex.lab=1.2
)
lines(tt, surv.cens.MC_scenario, type = "l",col = "red", lwd = 1.5,lty=2)
legend(legend = c("Real",
                  "Standard␣method"),
       col = c("blue", "black"),
       lty = c(1,2),
       x = "topright",
       cex = 1.2,
       bty = "n"
)

df <- data.frame(
  id=seq(1,50,by=1),
  time=tt[scenario2Fit[["index_max"]]]
)

ggplot(df, aes(x=time)) +
  geom_histogram(aes(y=..density..),binwidth = 0.5,colour="black",fill="whit
  geom_density(alpha=.2, fill="#FF6666") +
  labs(title = "Density␣plot␣of␣time␣points␣of␣maximum␣difference")
```

```
set.seed(123)
N<-500
x1<-sample(c(0,1),size = N,replace=TRUE,prob = c(0.5,0.5)) # treatment
x2<-round(rnorm(N,mean = 50,sd=10)) # age
data<-as.data.frame(cbind(x1,x2))
data$x2 <- (data$x2 - 50)/10
data.sim1<-simweib(500,data=data,0.1,0.05,0.5,0.1,0.5,0.1,2,2)
data.sim1$one <- 1
survtrue <- survfit(Surv(Tx, one) ~ 1, data = data.sim1)
surv<-survfit(Surv(time,status)~1,data=data.sim1)
ssf <- summary( survtrue, times=tt,extend = TRUE)
ssf1<- summary(surv, times=tt,extend = TRUE)
surv.real<-ssf$surv
surv.cens<-ssf1$surv
cens.perc <- 1 - sum(data.sim1$status)/500
correlationXC <- cor(data.sim1$Tx, data.sim1$Tc)
max_dist<-max(abs(surv.real-surv.cens))
area_between_curves<-(geiger:::.area.between.curves(tt,surv.real,surv.cens,
xrange = c(0,max(tt))))/max(tt)

plot(survtrue, conf.int=F, xlab = "Time", xlim=c(0,8), ylim=c(0,1),
ylab = "Survival␣probability",col = "blue")
lines(surv, conf.int=T,lty = 2, xlab = "Time", ylab = "Survival␣probability"
col = "red")
legend(legend = c("Real",
                  "Standard␣method"),
       col = c("blue", "red"),
       lty = c(1,2),
       x = "topright",
       cex = 1.2,
       bty = "n"
)


## Scenario 3:
set.seed(123)
N<-1000
x1<-sample(c(0,1),size = N,replace=TRUE,prob = c(0.5,0.5)) # treatment
x2<-round(rnorm(N,mean = 50,sd=10)) # age
data<-as.data.frame(cbind(x1,x2))
data$x2 <- (data$x2 - 50)/10
scenario3Fit  <- simIterations(M=M, #number of iterations
                               tt=tt, # time grid
                               N=N, # sample size
                               # Parameters survival model
                               lamdax=0.1, # ... .
                               beta1=0.5, # ...
                               beta2=0.1,
                          nx=2,
                          # Parameters censoring model
                          lamdac=0.05,
                          phi1=0.5,
                          phi2=0.1,
                          nc=2,
                          data=data

  )

surv.real.MC_scenario <- colMeans(scenario3Fit[["surv.real"]])
surv.cens.MC_scenario <- colMeans(scenario3Fit[["surv.cens"]])
cens.perc.MC<-mean(scenario3Fit[["cens.perc"]])
cens.perc.MC # percentage of censoring
area_between_curves.MC<-mean(scenario3Fit[["area_between_curves"]])
max_dist.MC<-mean(scenario3Fit[["max_dist"]])
area_between_curves.MC ## area
max_dist.MC ## max distance
correlationXC.MC<-mean(scenario3Fit[["correlationXC"]])
correlationXC.MC ## cor

plot(tt, surv.real.MC_scenario,
     xlab = "time", ylab = "Survival␣Probability",
     type="l", xlim=c(0,10), ylim=c(0,1), col = "blue",
     cex = 1.2,
     cex.main = 1.2,
     cex.axis = 1.2,
     cex.lab=1.2
)
lines(tt, surv.cens.MC_scenario, type = "l",col = "red", lwd = 1.5,lty=2)
legend(legend = c("Real",
                  "Standard␣method"),
       col = c("blue", "black"),
       lty = c(1,2),
       x = "topright",
       cex = 1.2,
       bty = "n"
)

df <- data.frame(
  id=seq(1,50,by=1),
  time=tt[scenario3Fit[["index_max"]]]
)

ggplot(df, aes(x=time)) +
 geom_histogram(aes(y=..density..),binwidth = 0.5,colour="black",fill="white
 geom_density(alpha=.2, fill="#FF6666") +
  abs(title = "Density␣plot␣of␣time␣points␣of␣maximum␣difference")
```

```
set.seed(123)
N<-1000
x1<-sample(c(0,1),size = N,replace=TRUE,prob = c(0.5,0.5)) #  treatment
x2<-round(rnorm(N,mean = 50,sd=10)) # age
data<-as.data.frame(cbind(x1,x2))
data$x2 <- (data$x2 - 50)/10
data.sim1<-simweib(1000,data=data,0.1,0.05,0.5,0.1,0.5,0.1,2,2)
data.sim1$one <- 1
survtrue <- survfit(Surv(Tx, one) ~ 1, data = data.sim1)
surv<-survfit(Surv(time,status)~1,data=data.sim1)
ssf <- summary( survtrue, times=tt,extend = TRUE)
ssf1<- summary(surv, times=tt,extend = TRUE)
surv.real<-ssf$surv
surv.cens<-ssf1$surv
cens.perc <- 1 - sum(data.sim1$status)/1000
correlationXC <- cor(data.sim1$Tx, data.sim1$Tc)
max_dist<-max(abs(surv.real-surv.cens))
area_between_curves<-(geiger:::.area.between.curves(tt,surv.real,surv.cens,
xrange = c(0,max(tt))))/max(tt)

plot(survtrue, conf.int=F, xlab = "Time", xlim=c(0,8), ylim=c(0,1),
ylab = "Survival␣probability",col = "blue")
lines(surv, conf.int=T,lty = 2, xlab = "Time", ylab = "Survival␣probability"
col = "red")
legend(legend = c("Real",
                  "Standard␣method"),
       col = c("blue", "red"),
       lty = c(1,2),
       x = "topright",
       cex = 1.2,
       bty = "n"
)
```

B.2 R code chapter 4

```
    ## Load packages

library(MCMCpack)
library(stats)
library(Rlab)
library(MASS)
library(Matrix)
library(mvtnorm)
library(survival)
library(ggplot2)
library(coda)
library(lattice)
library(boa)
library(nlme)
```

```r
library(car)
library(JM)
library("lattice")
library(simsurv)
library(ranger)
library(dplyr)
library(survminer)
library(kableExtra)
library(knitr)


# Functions

## Lambert W function by Ngwa et al.(2019)

  ### L a m b e r t s  W Function ###
  LambertW=function(z, b=0,maxiter=10,eps=.Machine$double.eps,
                  min.imag = 1e-9) {
    if (any(round(Re(b)) != b))
      stop("branch␣number␣for␣W␣must␣be␣an␣integer")
    if (!is.complex(z) && any(z<0)) z=as.complex(z)
    ## series expansion about -1/e
    ##
    ## p = (1 - 2*abs(b)).*sqrt(2*e*z+2);
    ## w = (11/72)*p;
    ## w = (w - 1/3).*p;
    ## w = (w+1).*p - 1
    ##
    ## first-order version suffices:
    ##
    w = (1 - 2*abs(b))*sqrt(2*exp(1)*z+2) - 1
    ## asymptotic expansion at 0 and Inf
    ##
    v=log(z+as.numeric(z==0 & b==0)) + 2*pi*b*1i;
    v=v - log(v+as.numeric(v==0))
    ## choose strategy for initial guess
    ##
    c=abs(z+exp(-1));
    c = (c > 1.45 - 1.1*abs(b));
    c=c | (b*Im(z) > 0) | (!Im(z) & (b == 1))
    w = (1 - c)*w+c*v
    ## Halley iteration
    ##
    for (n in 1:maxiter) {
      p=exp(w)
      t=w*p - z
      f = (w != -1)
      t=f*t/(p*(w+f) - 0.5*(w+2.0)*t/(w+f))
      w=w - t
      if (abs(Re(t)) < (2.48*eps)*(1.0+abs(Re(w)))
          && abs(Im(t)) < (2.48*eps)*(1.0+abs(Im(w))))
```

```
        break
    }
    if (n==maxiter) warning(paste("iteration␣limit␣(",maxiter,")
reached,␣result␣of␣W␣may␣be␣inaccurate", sep=""))
    if (all(Im(w) < min.imag)) w=as.numeric(w)
    return(w)
  }

 ## simTIMEexp

 simTIMEexp<-function(Survweib,Censoring)
  {
    time<-pmin(Survweib, Censoring)
    status<-as.numeric(Survweib <= Censoring)
    data.frame(id=1:N,Tx=Survweib,Tc=Censoring,time=time,status=status)
  }

## Simulation iterations

simIterations <- function(# Parameters simulation:
                          K, #number of iterations
                          tt, # time grid
                          N, # sample size
                          M, # ...
                          Time,
                          # Parameters survival model
                          link, # ... .
                          biobeta, # ...
                          Treatbeta,
                          wshape,
                          scale,
                          # Parameters censoring model
                          link1,
                          biobeta1,
                          Treatbeta1,
                          wshape1,
                          scale1
  )
{

  ## Save results of each iteration in one vector:

  surv.real <- matrix(nrow = K, ncol = length(tt))
  surv.cens <- matrix(nrow = K, ncol = length(tt))
  # Save censoring percentage in:
  cens.perc <- vector(mode = "numeric", length = K)
  #save the distance between the curves
  area_between_curves<-vector(mode = "numeric", length = K)
  #save the max distance between the curves
  max_dist<-vector(mode = "numeric", length = K)
  # Save correlation between X and C in:
```

```
correlationXC <- vector(mode = "numeric", length = K)


for(k in 1:K){

  ### Creating Random Effects ###
  Ubeta=c(2, 1) ## fixed effects
  G=structure(.Data=c(0.5, -0.005, -0.005, 0.001), .Dim= c(2, 2))

  UY=rmvnorm(N, mean=Ubeta, sigma=G, method = "chol") # fixed + random
  # effects ~ N(Ubeta,G)
  Z=matrix(0,M,length(dim(G)))
  Z[,1] <- 1
  Z[,2] <- c(Time)
  muy <- matrix(0,N,M)
  Y <- matrix(0,N,M)
  R <- diag(rnorm(M,1.000,0.00001), M)
  V=Z%*%G%*%t(Z) + R #variance-covariance matrix of the repeated measures

  ### Covariate Specification ###
  SEX=rbern(N,0.540)
  TREATMENT=sample(c(0,1),size = N,replace=TRUE,prob = c(0.5,0.5))
  for (i in 1:N){if (SEX[i] == 0) SEX[i]=1 else SEX[i]=2 }
  x2<-round(rnorm(N,mean = 50,sd=10))
  Bio<-(x2 - 50)/10


  ### Generating the Trajectories Outcome ###
  for (j in 1:M){
    for (i in 1:N){
      muy[i, j] <- UY[i,1] + UY[i,2]*(Time[j])
    }
  }
  muy <- data.frame(muy)
  names(muy) <- c("X1", "X2", "X3", "X4", "X5", "X6")
  ### Simulating Y Values ###

  Y <- rmvnorm(n=N, mean=colMeans(muy), sigma=V, method ="chol")
  Y <- data.frame(Y)
  names(Y) <- c("Y1", "Y2", "Y3", "Y4", "Y5", "Y6")
  Error <- rnorm(N, 0, 0.001)
  for (i in 1:N){
    Y[i, ] <- Y[i, ] + Error[i]
  }

  ### Linear Mixed Model Fit ###

  Timeinterval <- rep(Time, N)
  YSimul <- c(t(as.matrix(Y)))
  ID <- rep(1:N, each=M)
  bioR <- rep(Bio, each=M)
  Long <- data.frame(ID, YSimul, Timeinterval, bioR)
  LME <- lme(YSimul~Timeinterval, random = ~ Timeinterval | ID,
           data=Long, control=lmeControl(msMaxIter = 100, msVerbose=TRUE,
                                       opt = "optim"))
  U <- coef(LME)
  #U1<-random.effects(LME)
  names(U) <- c("X1_1", "X2_1")
  LME_Coeff <- data.frame(coefficients(LME))
  colMeans(LME_Coeff)


  ### Survival Times Using Weibull Distribution ###
  # # Lambda (Scale) = (1/Lambda)*v


  Uni <- runif(N, min = 0, max = 1)
  Numweib <- -log(Uni)
  Denweib <- scale*exp(biobeta*Bio+Treatbeta*TREATMENT+link*(U[,1]))
  ratioweib <- link*(U[,2])*(1/wshape)*((Numweib/Denweib)^(1/wshape))
  Lweib <- LambertW(ratioweib)
  rLweib<-Re(Lweib)
  Survweib <- Lweib*1/(link*(U[,2])*(1/wshape))

  ### Censoring Times Using Weibull Distribution ###

  Uni <- runif(N, min = 0, max = 1)
  Numweib <- -log(Uni)
  Denweib <- scale1*exp(biobeta1*Bio+Treatbeta1*TREATMENT+link1*(U[,1]))
  ratioweib <- link1*(U[,2])*(1/wshape1)*((Numweib/Denweib)^(1/wshape1))
  Lweib <- LambertW(ratioweib)
  rLweib<-Re(Lweib)
  Censoring<- Lweib*1/(link1*(U[,2])*(1/wshape))

  data.sim1<-simTIMEexp(Survweib,Censoring)
  data.sim2<-data.sim1[data.sim1$status==1,]
  data.sim1$one<- 1
  survtrue <- survfit(Surv(Tx, one) ~ 1, data = data.sim1)
  ssf <- summary( survtrue, times=tt,extend = TRUE)


  surv<-survfit(Surv(time,status)~1,data=data.sim1)
  ssf1<- summary(surv, times=tt,extend = TRUE)
  surv.real[k,]<-ssf$surv
  surv.cens[k,]<-ssf1$surv
  cens.perc[k] <- 1 - sum(data.sim1$status)/500
  correlationXC[k] <- cor(data.sim1$Tx, data.sim1$Tc)
  max_dist[k]<-max(abs(surv.real[k,]-surv.cens[k,]))
area_between_curves[k]<-(geiger:::.area.between.curves(tt,surv.real[k,],
surv.cens[k,], xrange = c(0,max(data.sim2$time))))/max(data.sim2$time)
```

```
  }

  return(list(surv.real=surv.real,
              surv.cens=surv.cens,
              cens.perc=cens.perc,
              max_dist=max_dist,
              area_between_curves=area_between_curves,
              correlationXC=correlationXC,
              data=data.sim1))
}

# Fit models for different scenarios

## Define parameters that should be the same in each scenario

K <- 50 # number of iterations
tt <- c(seq(0,6,by=0.1)) # time grid
N <- 500 # sample size
M <- 6
Time <- c(0:5)

## Scenario 1

set.seed(123)
scenario1Fit <- simIterations(K = K,
                              tt = tt ,
                              N = N,
                              M = M,
                              Time = Time,
                              #Parameters survival model
                              link = 0.500,
                              biobeta = 0.1,
                              Treatbeta = 0.5,
                              wshape = 2,
                              scale = 0.01,
                              # Parameters censoring model
                              link1 = 0.5,
                              biobeta1 = 0.5,
                              Treatbeta1 = 1.5,
                              wshape1 = 2,
                              scale1 = 0.0028

)

# Summarize results:

surv.real.MC_scenario1 <- colMeans(scenario1Fit[["surv.real"]])
surv.cens.MC_scenario1 <- colMeans(scenario1Fit[["surv.cens"]])
cens.perc.MC<-mean(scenario1Fit[["cens.perc"]])
cens.perc.MC # percentage of censoring
area_between_curves.MC1<-mean(scenario1Fit[["area_between_curves"]])
```

```
max_dist.MC1<-mean(scenario1Fit[["max_dist"]])
area_between_curves.MC1
max_dist.MC1
correlationXC.MC<-mean(scenario1Fit[["correlationXC"]])

    plot(tt, surv.real.MC_scenario1,
     xlab = "time", ylab = "Survival␣Probability",
     type="l", xlim=c(0,6), ylim=c(0,1), col = "blue",
     cex = 1.2,
     cex.main = 1.2,
     cex.axis = 1.2,
     cex.lab=1.2
)
lines(tt, surv.cens.MC_scenario1, type = "l", col = "red", lwd = 1.5,lty=2)
legend(legend = c("Real",
                  "Standard␣method"),
       col = c("blue", "red"),
       lty = c(1,2),
       x = "topright",
       cex = 1.2,
       bty = "n"
)

data<-scenario1Fit[["data"]]
den<-density(data$time)
plot(den)


set.seed(123)
scenario2Fit <- simIterations(K = K,
                          tt = tt ,
                          N = N,
                          M = M,
                          Time = Time,
                          #Parameters survival model
                          link = 0.500,
                          biobeta = 0.1,
                          Treatbeta = 0.5,
                          wshape = 2,
                          scale = 0.01,
                          # Parameters censoring model
                          link1 = 0.5,
                          biobeta1 = 0.1,
                          Treatbeta1 = 0.5,
                          wshape1 = 2,
                          scale1 = 0.005

)

# Summarize results:
surv.real.MC_scenario2 <- colMeans(scenario2Fit[["surv.real"]])
```

```r
surv.cens.MC_scenario2 <- colMeans(scenario2Fit[["surv.cens"]])
cens.perc.MC<-mean(scenario2Fit[["cens.perc"]])
cens.perc.MC # percentage of censoring
area_between_curves.MC2<-mean(scenario2Fit[["area_between_curves"]])
max_dist.MC2<-mean(scenario2Fit[["max_dist"]])
area_between_curves.MC2
max_dist.MC2
correlationXC.MC<-mean(scenario2Fit[["correlationXC"]])

   plot(tt, surv.real.MC_scenario2,
     xlab = "time", ylab = "Survival␣Probability",
     type="l", xlim=c(0,6), ylim=c(0,1), col = "blue",
     cex = 1.2,
     cex.main = 1.2,
     cex.axis = 1.2,
     cex.lab=1.2
)
lines(tt, surv.cens.MC_scenario2, type = "l", col = "red", lwd = 1.5,lty=2)
legend(legend = c("Real",
                  "Standard␣method"),
       col = c("blue", "red"),
       lty = c(1,2),
       x = "topright",
       cex = 1.2,
       bty = "n"
)



set.seed(123)
scenario3Fit <- simIterations(K = K,
                              tt = tt ,
                              N = N,
                              M = M,
                              Time = Time,
                              #Parameters survival model
                              link = 0.500,
                              biobeta = 0.1,
                              Treatbeta = 0.5,
                              wshape = 2,
                              scale = 0.01,
                              # Parameters censoring model
                              link1 = 0.5,
                              biobeta1 = 0,
                              Treatbeta1 = 0.1,
                              wshape1 = 2,
                              scale1 = 0.0057

)

# Summarize results:
```

```
surv.real.MC_scenario3 <- colMeans(scenario3Fit[["surv.real"]])
surv.cens.MC_scenario3 <- colMeans(scenario3Fit[["surv.cens"]])
cens.perc.MC<-mean(scenario3Fit[["cens.perc"]])
cens.perc.MC # percentage of censoring
area_between_curves.MC3<-mean(scenario3Fit[["area_between_curves"]])
max_dist.MC3<-mean(scenario3Fit[["max_dist"]])
area_between_curves.MC3
max_dist.MC3
correlationXC.MC<-mean(scenario3Fit[["correlationXC"]])

    plot(tt, surv.real.MC_scenario3,
     xlab = "time", ylab = "Survival␣Probability",
     type="l", xlim=c(0,6), ylim=c(0,1), col = "blue",
     cex = 1.2,
     cex.main = 1.2,
     cex.axis = 1.2,
     cex.lab=1.2
)
lines(tt, surv.cens.MC_scenario3, type = "l", col = "red", lwd = 1.5,lty=2)
legend(legend = c("Real",
                  "Standard␣method"),
       col = c("blue", "red"),
       lty = c(1,2),
       x = "topright",
       cex = 1.2,
       bty = "n"
)

# Varying the link parameter gamma

## Scenario 1:

set.seed(123)
scenario1Fit <- simIterations(K = K,
                         tt = tt ,
                         N = N,
                         M = M,
                         Time = Time,
                         #Parameters survival model
                         link = 0.500,
                         biobeta = 0.1,
                         Treatbeta = 0.5,
                         wshape = 2,
                         scale = 0.01,
                         # Parameters censoring model
                         link1 = 0.1,
                         biobeta1 = 0.1,
                         Treatbeta1 = 0.5,
                         wshape1 = 2,
                         scale1 = 0.033
```

```
)

# Summarize results:

surv.real.MC_scenario1 <- colMeans(scenario1Fit[["surv.real"]])
surv.cens.MC_scenario1 <- colMeans(scenario1Fit[["surv.cens"]])
cens.perc.MC<-mean(scenario1Fit[["cens.perc"]])
cens.perc.MC # percentage of censoring
area_between_curves.MC1<-mean(scenario1Fit[["area_between_curves"]])
max_dist.MC1<-mean(scenario1Fit[["max_dist"]])
area_between_curves.MC1
max_dist.MC1
correlationXC.MC<-mean(scenario1Fit[["correlationXC"]])

     plot(tt, surv.real.MC_scenario1,
     xlab = "time", ylab = "Survival␣Probability",
     type="l", xlim=c(0,6), ylim=c(0,1), col = "blue",
     cex = 1.2,
     cex.main = 1.2,
     cex.axis = 1.2,
     cex.lab=1.2
)
lines(tt, surv.cens.MC_scenario1, type = "l", col = "red", lwd = 1.5,lty=2)
legend(legend = c("Real",
                  "Standard␣method"),
       col = c("blue", "red"),
       lty = c(1,2),
       x = "topright",
       cex = 1.2,
       bty = "n"
)




## Scenario 2:

set.seed(123)
scenario2Fit <- simIterations(K = K,
                              tt = tt ,
                              N = N,
                              M = M,
                              Time = Time,
                              #Parameters survival model
                              link = 0.500,
                              biobeta = 0.1,
                              Treatbeta = 0.5,
                              wshape = 2,
                              scale = 0.01,
                              # Parameters censoring model
                              link1 = 0.5,
                              biobeta1 = 0.1,
                              Treatbeta1 = 0.5,
                              wshape1 = 2,
                              scale1 = 0.005

   )

   # Summarize results:
   surv.real.MC_scenario2 <- colMeans(scenario2Fit[["surv.real"]])
   surv.cens.MC_scenario2 <- colMeans(scenario2Fit[["surv.cens"]])
   cens.perc.MC<-mean(scenario2Fit[["cens.perc"]])
   cens.perc.MC # percentage of censoring
   area_between_curves.MC2<-mean(scenario2Fit[["area_between_curves"]])
   max_dist.MC2<-mean(scenario2Fit[["max_dist"]])
   area_between_curves.MC2
   max_dist.MC2
   correlationXC.MC<-mean(scenario2Fit[["correlationXC"]]

     plot(tt, surv.real.MC_scenario2,
        xlab = "time", ylab = "Survival␣Probability",
        type="l", xlim=c(0,6), ylim=c(0,1), col = "blue",
        cex = 1.2,
        cex.main = 1.2,
        cex.axis = 1.2,
        cex.lab=1.2
   )
   lines(tt, surv.cens.MC_scenario2, type = "l", col = "red", lwd = 1.5,lty=2)
   legend(legend = c("Real",
                    "Standard␣method"),
          col = c("blue", "red"),
          lty = c(1,2),
          x = "topright",
          cex = 1.2,
          bty = "n"
   )




   ## Scenario 3:

   set.seed(123)
   scenario3Fit <- simIterations(K = K,
                              tt = tt ,
                              N = N,
                              M = M,
                              Time = Time,
                              #Parameters survival model
                              link = 0.500,
                              biobeta = 0.1,
                              Treatbeta = 0.5,
                              wshape = 2,
                              scale = 0.01,
                              # Parameters censoring model
                              link1 = 1,
                              biobeta1 = 0.1,
                              Treatbeta1 = 0.5,
                              wshape1 = 2,
                              scale1 = 0.00045

   )

   # Summarize results:
   surv.real.MC_scenario3 <- colMeans(scenario3Fit[["surv.real"]])
   surv.cens.MC_scenario3 <- colMeans(scenario3Fit[["surv.cens"]])
   cens.perc.MC<-mean(scenario3Fit[["cens.perc"]])
   cens.perc.MC # percentage of censoring
   area_between_curves.MC3<-mean(scenario3Fit[["area_between_curves"]])
   max_dist.MC3<-mean(scenario3Fit[["max_dist"]])
   area_between_curves.MC3
   max_dist.MC3
   correlationXC.MC<-mean(scenario3Fit[["correlationXC"]])


   plot(tt, surv.real.MC_scenario3,
        xlab = "time", ylab = "Survival␣Probability",
        type="l", xlim=c(0,6), ylim=c(0,1), col = "blue",
        cex = 1.2,
        cex.main = 1.2,
        cex.axis = 1.2,
        cex.lab=1.2
   )
   lines(tt, surv.cens.MC_scenario3, type = "l", col = "red", lwd = 1.5,lty=2)
   legend(legend = c("Real",
                    "Standard␣method"),
          col = c("blue", "red"),
          lty = c(1,2),
          x = "topright",
          cex = 1.2,
          bty = "n"
   )




   ## Scenario 4:

   set.seed(123)
   scenario4Fit <- simIterations(K = K,
                              tt = tt ,
                              N = N,
                              M = M,
                              Time = Time,
                              #Parameters survival model
                              link = 0.500,
                              biobeta = 0.1,
                              Treatbeta = 0.5,
                              wshape = 2,
                              scale = 0.01,
                              # Parameters censoring model
                              link1 = 1.5,
                              biobeta1 = 0.1,
                              Treatbeta1 = 0.5,
                              wshape1 = 2,
                              scale1 = 0.00004

)

# Summarize results:
surv.real.MC_scenario4 <- colMeans(scenario4Fit[["surv.real"]])
surv.cens.MC_scenario4 <- colMeans(scenario4Fit[["surv.cens"]])
cens.perc.MC<-mean(scenario4Fit[["cens.perc"]])
cens.perc.MC # percentage of censoring
area_between_curves.MC4<-mean(scenario4Fit[["area_between_curves"]])
max_dist.MC4<-mean(scenario4Fit[["max_dist"]])
area_between_curves.MC4
max_dist.MC4
correlationXC.MC<-mean(scenario4Fit[["correlationXC"]])

plot(tt, surv.real.MC_scenario4,
     xlab = "time", ylab = "Survival␣Probability",
     type="l", xlim=c(0,6), ylim=c(0,1), col = "blue",
     cex = 1.2,
     cex.main = 1.2,
     cex.axis = 1.2,
     cex.lab=1.2
)
lines(tt, surv.cens.MC_scenario4, type = "l", col = "red", lwd = 1.5,lty=2)
legend(legend = c("Real",
                  "Standard␣method"),
       col = c("blue", "red"),
       lty = c(1,2),
       x = "topright",
       cex = 1.2,
       bty = "n"
)

# Varying the censoring percentage

## Scenario 1:

set.seed(123)
scenario1Fit <- simIterations(K = K,
                                    tt = tt ,
                                    N = N,
                                    M = M,
                                    Time = Time,
                                    #Parameters survival model
                                    link = 0.500,
                                    biobeta = 0.1,
                                    Treatbeta = 0.5,
                                    wshape = 2,
                                    scale = 0.01,
                                    # Parameters censoring model
                                    link1 = 0.5,
                                    biobeta1 = 0.1,
                                    Treatbeta1 = 0.5,
                                    wshape1 = 2,
                                    scale1 = 0.0012

)

# Summarize results:

surv.real.MC_scenario1 <- colMeans(scenario1Fit[["surv.real"]])
surv.cens.MC_scenario1 <- colMeans(scenario1Fit[["surv.cens"]])
cens.perc.MC<-mean(scenario1Fit[["cens.perc"]])
cens.perc.MC # percentage of censoring
area_between_curves.MC1<-mean(scenario1Fit[["area_between_curves"]])
max_dist.MC1<-mean(scenario1Fit[["max_dist"]])
area_between_curves.MC1
max_dist.MC1
correlationXC.MC<-mean(scenario1Fit[["correlationXC"]])


plot(tt, surv.real.MC_scenario1,
     xlab = "time", ylab = "Survival␣Probability",
     type="l", xlim=c(0,6), ylim=c(0,1), col = "blue",
     cex = 1.2,
     cex.main = 1.2,
     cex.axis = 1.2,
     cex.lab=1.2
)
lines(tt, surv.cens.MC_scenario1, type = "l", col = "red", lwd = 1.5,lty=2)
legend(legend = c("Real",
                  "Standard␣method"),
       col = c("blue", "red"),
       lty = c(1,2),
       x = "topright",
       cex = 1.2,
       bty = "n"
)
```

```
## Scenario 2:

set.seed(123)
scenario2Fit <- simIterations(K = K,
                          tt = tt ,
                          N = N,
                          M = M,
                          Time = Time,
                          #Parameters survival model
                          link = 0.500,
                          biobeta = 0.1,
                          Treatbeta = 0.5,
                          wshape = 2,
                          scale = 0.01,
                          # Parameters censoring model
                          link1 = 0.5,
                          biobeta1 = 0.1,
                          Treatbeta1 = 0.5,
                          wshape1 = 2,
                          scale1 = 0.0044

)

# Summarize results:
surv.real.MC_scenario2 <- colMeans(scenario2Fit[["surv.real"]])
surv.cens.MC_scenario2 <- colMeans(scenario2Fit[["surv.cens"]])
cens.perc.MC<-mean(scenario2Fit[["cens.perc"]])
cens.perc.MC # percentage of censoring
area_between_curves.MC2<-mean(scenario2Fit[["area_between_curves"]])
max_dist.MC2<-mean(scenario2Fit[["max_dist"]])
area_between_curves.MC2
max_dist.MC2
correlationXC.MC<-mean(scenario2Fit[["correlationXC"]])

    plot(tt, surv.real.MC_scenario2,
     xlab = "time", ylab = "Survival␣Probability",
     type="l", xlim=c(0,6), ylim=c(0,1), col = "blue",
     cex = 1.2,
     cex.main = 1.2,
     cex.axis = 1.2,
     cex.lab=1.2
)
lines(tt, surv.cens.MC_scenario2, type = "l", col = "red", lwd = 1.5,lty=2)
legend(legend = c("Real",
                 "Standard␣method"),
       col = c("blue", "red"),
       lty = c(1,2),
       x = "topright",
       cex = 1.2,
       bty = "n"
```

```
)




## Scenario 3:

set.seed(123)
scenario3Fit <- simIterations(K = K,
                              tt = tt ,
                              N = N,
                              M = M,
                              Time = Time,
                              #Parameters survival model
                              link = 0.500,
                              biobeta = 0.1,
                              Treatbeta = 0.5,
                              wshape = 2,
                              scale = 0.01,
                              # Parameters censoring model
                              link1 = 0.5,
                              biobeta1 = 0.1,
                              Treatbeta1 = 0.5,
                              wshape1 = 2,
                              scale1 = 0.01

)

# Summarize results:
surv.real.MC_scenario3 <- colMeans(scenario3Fit[["surv.real"]])
surv.cens.MC_scenario3 <- colMeans(scenario3Fit[["surv.cens"]])
cens.perc.MC<-mean(scenario3Fit[["cens.perc"]])
cens.perc.MC # percentage of censoring
area_between_curves.MC3<-mean(scenario3Fit[["area_between_curves"]])
max_dist.MC3<-mean(scenario3Fit[["max_dist"]])
area_between_curves.MC3
max_dist.MC3
correlationXC.MC<-mean(scenario3Fit[["correlationXC"]])

   plot(tt, surv.real.MC_scenario3,
     xlab = "time", ylab = "Survival␣Probability",
     type="l", xlim=c(0,6), ylim=c(0,1), col = "blue",
     cex = 1.2,
     cex.main = 1.2,
     cex.axis = 1.2,
     cex.lab=1.2
)
lines(tt, surv.cens.MC_scenario3, type = "l", col = "red", lwd = 1.5,lty=2)
legend(legend = c("Real",
                  "Standard␣method"),
       col = c("blue", "red"),
       lty = c(1,2),
       x = "topright",
       cex = 1.2,
       bty = "n"
)
```

B.3 R code chapter 5

```
## Load packages


library(survival)
library(ranger)
library(ggplot2)
library(dplyr)
library(survminer)
library(kableExtra)
library(tcltk)
library(ipw)

# Transform the data
transform.data <- function(data)
  {
  # Define Tstart and the indicator "censored":
data$Tstart <- 0
data$censored <- 1 - data$status
# Times at which to split the intervals:
cut.times <- unique(data$time)
# Split data with event = di (event):
data.long <- survSplit(data = data,
cut = cut.times,
end = "time",
start = "Tstart",
event = "status")
data.long <- data.long[order(data.long$id,
data.long$time),]
# Split data with event = censored (censoring):
data.long.cens <- survSplit(data,
cut=cut.times,
end="time",
start="Tstart",
event="censored")
data.long.cens <- data.long.cens[order(data.long.cens$id,
data.long.cens$time),]
# Add "censored" indicator to long data format:
data.long$censored <- data.long.cens$censored
data.long$id <- as.numeric(data.long$id)
# Return long data format:
return(data.long)
}

#Function to estimate the survival curve
#(Product-Limit Estimator) with weighted subjects.

calc.surv.IPCW <- function(Tstart, Tstop, status,
 tt, IPCW.weights,data.long)
{
# Fit the Product-Limit estimator with weighted subjects
surv.IPCW <- survfit(Surv(Tstart, Tstop, status) ~ 1, data = data.long,
weights = IPCW.weights)
# Estimate survival probabilities at time points tt
ssurv.IPCW <- summary(surv.IPCW, times=tt, extend = TRUE)
# Return resulting survival probabilities:
return(ssurv.IPCW$surv)
}



## simweib

simweib<-function (N,data,lamdax,lamdac,beta1,beta2,phi1,phi2,nx,nc)
{

  u1<-runif(N)
  u2<-runif(N)
  Tx<-(-log(u1)/(lamdax*exp(data$x1*beta1+data$x2*beta2)))^(1/nx)
  Tc<-(-log(u2)/(lamdac*exp(data$x1*phi1+data$x2*phi2)))^(1/nc)
  time<-pmin(Tx,Tc)
  status<-as.numeric(Tx<=Tc)
  data.frame(id=1:N,Tx=Tx,Tc=Tc,time=time,status=status,x1=data$x1,
  x2=data$x2)
}

## Simulation iterations

simIterations <- function(# Parameters simulation:
                          M, #number of iterations
                          tt, # time grid
                          N, # sample size
                          # Parameters survival model
                          lamdax, # ... .
                          beta1, # ...
                          beta2,
                          nx,
                          # Parameters censoring model
                          lamdac,
                          phi1,
                            phi2,
                            nc,
                            data

  )
{

surv.real <- matrix(nrow = M, ncol = length(tt))
surv.cens <- matrix(nrow = M, ncol = length(tt))
# Save censoring percentage in:
cens.perc <- vector(mode = "numeric", length = M)
# Save correlation between X and C in:
correlationXC <- vector(mode = "numeric", length = M)
#save the distance between the curves
area_between_curves<-vector(mode = "numeric", length = M)
area_between_curvesUnStab<-vector(mode = "numeric", length = M)
area_between_curvesStab<-vector(mode = "numeric", length = M)
#save the max distance between the curves
max_dist<-vector(mode = "numeric", length = M)
index_max<-vector(mode = "numeric", length = M)
surv.IPCW.UnStab <- matrix(nrow = M, ncol = length(tt))
surv.IPCW.Stab <- matrix(nrow = M, ncol = length(tt))
phi.IPCW <- matrix(nrow = M, ncol = 2)

for(j in 1:M)
{

  data.sim1<-simweib(N,data,lamdax,lamdac,beta1,beta2,phi1,phi2,nx,nc)
  data.sim2<-data.sim1[data.sim1$status==1,]
  data.sim1$one <- 1
  survtrue <- survfit(Surv(Tx, one) ~ 1, data = data.sim1)
  surv<-survfit(Surv(time,status)~1,data=data.sim1)
  ssf <- summary( survtrue, times=tt,extend = TRUE)
  ssf1<- summary(surv, times=tt,extend = TRUE)
  surv.real[j,]<-ssf$surv
  surv.cens[j,]<-ssf1$surv
  cens.perc[j] <- 1 - sum(data.sim1$status)/N
  correlationXC[j] <- cor(data.sim1$Tx, data.sim1$Tc)
  max_dist[j]<-max(abs(surv.real[j,]-surv.cens[j,]))
  index_max[j]<-which.max(abs(surv.real[j,]-surv.cens[j,]))
area_between_curves[j]<-(geiger:::.area.between.curves(tt,surv.real[j,],
surv.cens[j,], xrange = c(0,max(data.sim2$time))))/max(data.sim2$time)

 # Transform the data into a long format:
data.sim.long <- transform.data(data.sim1)

# Calculate the IPCW weights:
ipwStab <- ipwtm(
           exposure = censored,
           family = "survival",
           numerator = ~ 1,
           denominator = ~ x1+x2,
           id = id,
           tstart = Tstart,
           timevar = time,
           type = "first",
           data = data.sim.long )

ipwUnStab <- ipwtm(
           exposure = censored,
           family = "survival",
           #numerator = ~ 1,
           denominator = ~ x1+x2,
           id = id,
           tstart = Tstart,
           timevar = time,
           type = "first",
           data = data.sim.long )

data.sim.long $ipw_weights <- ipwUnStab$ipw.weights
data.sim.long $ipw_Stabweights <- ipwStab$ipw.weights


# Calculate survival probabilities for the testpersons:
surv.IPCW.Stab[j,] <- calc.surv.IPCW(Tstart = data.sim.long$Tstart,
Tstop = data.sim.long$time,
status = data.sim.long$status,
tt = tt,
IPCW.weights = data.sim.long $ipw_Stabweights,
data.long = data.sim.long)

## Unstabilized Weights ##

# Calculate survival probabilities:
surv.IPCW.UnStab[j,] <- calc.surv.IPCW(Tstart = data.sim.long$Tstart,
Tstop = data.sim.long$time,
status = data.sim.long$status,
tt = tt,
IPCW.weights = data.sim.long $ipw_weights,
data.long = data.sim.long)



CZ <- coxph(Surv(Tstart,
                 time,
                 censored) ~  x1+x2,
            data = data.sim.long)

phi.IPCW[j,] <- summary(CZ)$coef[,1]


area_between_curvesStab[j]<-(geiger:::.area.between.curves(tt,surv.real[j,],
surv.IPCW.Stab[j,], xrange = c(0,max(data.sim2$time))))/max(data.sim2$time)
```

```
area_between_curvesUnStab[j]<-(geiger:::.area.between.curves(tt,surv.real[j,
surv.IPCW.UnStab[j,],xrange = c(0,max(data.sim2$time))))/max(data.sim2$time)




}
return(list(surv.real=surv.real,
              surv.cens=surv.cens,
              cens.perc=cens.perc,
              max_dist=max_dist,
              area_between_curves=area_between_curves,
              area_between_curvesStab=area_between_curvesStab,
              area_between_curvesUnStab=area_between_curvesUnStab,
              correlationXC=correlationXC,
             index_max=index_max,surv.IPCW.UnStab=surv.IPCW.UnStab,
             surv.IPCW.Stab=surv.IPCW.Stab,
             phi.IPCW=phi.IPCW

             ))

}

# Fit models for different scenarios

## Define parameters that should be the same in each scenario
set.seed(123)
M <- 50 # number of Monte Carlo repetitions
N<-500
x1<-sample(c(0,1),size = N,replace=TRUE,prob = c(0.5,0.5)) # treatment
x2<-round(rnorm(N,mean = 50,sd=10))
data<-as.data.frame(cbind(x1,x2))
data$x2 <- (data$x2 - 50)/10
# times at which survival prob are calculated
tt <- c(seq(0,10,by=0.1))

## Scenario 1:
set.seed(123)
scenario1Fit  <- simIterations(M=M, #number of iterations
                          tt=tt, # time grid
                          N=N, # sample size
                          # Parameters survival model
                          lamdax=0.1, # ... .
                          beta1=0.5, # ...
                          beta2=0.1,
                          nx=2,
                          # Parameters censoring model
                          lamdac=0.027,
                          phi1=1.5,
                          phi2=0.5,
                          nc=2,
                          data=data

  )

surv.real.MC_scenario <- colMeans(scenario1Fit[["surv.real"]])
surv.cens.MC_scenario <- colMeans(scenario1Fit[["surv.cens"]])
surv.IPCW.UnStab.MC_scenario<-colMeans(scenario1Fit[["surv.IPCW.UnStab"]])
surv.IPCW.Stab.MC_scenario<-colMeans(scenario1Fit[["surv.IPCW.Stab"]])
cens.perc.MC<-mean(scenario1Fit[["cens.perc"]])
cens.perc.MC # percentage of censoring
area_between_curves.MC<-mean(scenario1Fit[["area_between_curves"]])
area_between_curves.MC.UnStab<-mean(scenario1Fit[["area_between_curvesUnStab
area_between_curves.MC.Stab<-mean(scenario1Fit[["area_between_curvesStab"]])
max_dist.MC<-mean(scenario1Fit[["max_dist"]])
area_between_curves.MC ## area
max_dist.MC ## max distance
correlationXC.MC<-mean(scenario1Fit[["correlationXC"]])
correlationXC.MC ## cor

plot(tt, surv.real.MC_scenario,
     xlab = "time", ylab = "Survival␣Probability",
     type="l", xlim=c(0,8), ylim=c(0,1), col = "blue",
     cex = 1.2,
     cex.main = 1.2,
     cex.axis = 1.2,
     cex.lab=1.2
)
lines(tt, surv.cens.MC_scenario, type = "l",lty=2,col = "black", lwd = 1.5)
lines(tt, surv.IPCW.UnStab.MC_scenario, type = "l",lty=3,
col = "red", lwd = 1.5)
lines(tt, surv.IPCW.Stab.MC_scenario, type = "l",lty=4,
col = "green", lwd = 1.5)
legend(legend = c("Real",
                  "Standard␣method","UnStabW","StabW"),
       col = c("blue", "black","red","green"),
       lty = c(1,2,3,4),
       x = "topright",
       cex = 1.2,
       bty = "n"
)

colMeans(scenario1Fit[["phi.IPCW"]])

table<-matrix(NA,nrow=3,ncol = 1)
rownames(table)<-c("Standard␣method","UnStabW","StabW")
colnames(table)<-c("Area␣between␣the␣curves")
table[,1]<-c(area_between_curves.MC,area_between_curves.MC.UnStab,
area_between_curves.MC.Stab)

## Scenario 2:
set.seed(123)
```

```
scenario2Fit  <- simIterations(M=M, #number of iterations
                          tt=tt, # time grid
                          N=N, # sample size
                          # Parameters survival model
                          lamdax=0.1, # ... .
                          beta1=0.5, # ...
                          beta2=0.1,
                          nx=2,
                          # Parameters censoring model
                          lamdac=0.003,
                          phi1=4.5,
                          phi2=1.5,
                          nc=2,
                          data=data

  )


surv.real.MC_scenario <- colMeans(scenario2Fit[["surv.real"]])
surv.cens.MC_scenario <- colMeans(scenario2Fit[["surv.cens"]])
surv.IPCW.UnStab.MC_scenario<-colMeans(scenario2Fit[["surv.IPCW.UnStab"]])
surv.IPCW.Stab.MC_scenario<-colMeans(scenario2Fit[["surv.IPCW.Stab"]])
cens.perc.MC<-mean(scenario2Fit[["cens.perc"]])
cens.perc.MC # percentage of censoring
area_between_curves.MC<-mean(scenario2Fit[["area_between_curves"]])
area_between_curves.MC.UnStab<-mean(scenario2Fit[["area_between_curvesUnStab
area_between_curves.MC.Stab<-mean(scenario2Fit[["area_between_curvesStab"]])
max_dist.MC<-mean(scenario2Fit[["max_dist"]])
area_between_curves.MC ## area
max_dist.MC ## max distance
correlationXC.MC<-mean(scenario2Fit[["correlationXC"]])
correlationXC.MC ## cor

plot(tt, surv.real.MC_scenario,
     xlab = "time", ylab = "Survival␣Probability",
     type="l", xlim=c(0,8), ylim=c(0,1), col = "blue",
     cex = 1.2,
     cex.main = 1.2,
     cex.axis = 1.2,
     cex.lab=1.2
)
lines(tt, surv.cens.MC_scenario, type = "l",lty=2,col = "black", lwd = 1.5)
lines(tt, surv.IPCW.UnStab.MC_scenario, type = "l",lty=3,
col = "red", lwd = 1.5)
lines(tt, surv.IPCW.Stab.MC_scenario, type = "l",lty=4,
col = "green", lwd = 1.5)
legend(legend = c("Real",
                  "Standard␣method","UnStabW","StabW"),
       col = c("blue", "black","red","green"),
       lty = c(1,2,3,4),
       x = "topright",
       cex = 1.2,
       bty = "n"
)

colMeans(scenario2Fit[["phi.IPCW"]])

table<-matrix(NA,nrow=3,ncol = 1)
rownames(table)<-c("Standard␣method","UnStabW","StabW")
colnames(table)<-c("Area␣between␣the␣curves")
table[,1]<-c(area_between_curves.MC,area_between_curves.MC.UnStab,
area_between_curves.MC.Stab)


## scenario 3

set.seed(123)
scenario3Fit  <- simIterations(M=M, #number of iterations
                         tt=tt, # time grid
                         N=N, # sample size
                         # Parameters survival model
                         lamdax=0.1, # ... .
                         beta1=0.5, # ...
                         beta2=0.1,
                         nx=2,
                         # Parameters censoring model
                         lamdac=0.012,
                         phi1=0.5,
                         phi2=0.1,
                         nc=2,
                         data=data

  )


surv.real.MC_scenario <- colMeans(scenario3Fit[["surv.real"]])
surv.cens.MC_scenario <- colMeans(scenario3Fit[["surv.cens"]])
surv.IPCW.UnStab.MC_scenario<-colMeans(scenario3Fit[["surv.IPCW.UnStab"]])
surv.IPCW.Stab.MC_scenario<-colMeans(scenario3Fit[["surv.IPCW.Stab"]])
cens.perc.MC<-mean(scenario3Fit[["cens.perc"]])
cens.perc.MC # percentage of censoring
area_between_curves.MC<-mean(scenario3Fit[["area_between_curves"]])
area_between_curves.MC.UnStab<-mean(scenario3Fit[["area_between_curvesUnStab
area_between_curves.MC.Stab<-mean(scenario3Fit[["area_between_curvesStab"]])
max_dist.MC<-mean(scenario3Fit[["max_dist"]])
area_between_curves.MC ## area
max_dist.MC ## max distance
correlationXC.MC<-mean(scenario3Fit[["correlationXC"]])
correlationXC.MC ## cor

plot(tt, surv.real.MC_scenario,
     xlab = "time", ylab = "Survival␣Probability",
     type="l", xlim=c(0,8), ylim=c(0,1), col = "blue",
     cex = 1.2,
     cex.main = 1.2,
     cex.axis = 1.2,
     cex.lab=1.2
)
lines(tt, surv.cens.MC_scenario, type = "l",lty=2,col = "black", lwd = 1.5)
lines(tt, surv.IPCW.UnStab.MC_scenario, type = "l",lty=3,
col = "red", lwd = 1.5)
lines(tt, surv.IPCW.Stab.MC_scenario, type = "l",lty=4,
col = "green", lwd = 1.5)
legend(legend = c("Real",
                  "Standard␣method","UnStabW","StabW"),
       col = c("blue", "black","red","green"),
       lty = c(1,2,3,4),
       x = "topright",
       cex = 1.2,
       bty = "n"
)

colMeans(scenario3Fit[["phi.IPCW"]])

table<-matrix(NA,nrow=3,ncol = 1)
rownames(table)<-c("Standard␣method","UnStabW","StabW")
colnames(table)<-c("Area␣between␣the␣curves")
table[,1]<-c(area_between_curves.MC,area_between_curves.MC.UnStab,
area_between_curves.MC.Stab)

## Calculate weights with only one covariate
## Simulation iterations

## Remove x1~treatment
simIterations <- function(# Parameters simulation:
                          M, #number of iterations
                          tt, # time grid
                          N, # sample size
                          # Parameters survival model
                          lamdax, # ... .
                          beta1, # ...
                          beta2,
                          nx,
                          # Parameters censoring model
                          lamdac,
                          phi1,
                          phi2,
                          nc,
                          data

  )
{
```

```
surv.real <- matrix(nrow = M, ncol = length(tt))
surv.cens <- matrix(nrow = M, ncol = length(tt))
cens.perc <- vector(mode = "numeric", length = M)
# Save correlation between X and C in:
correlationXC <- vector(mode = "numeric", length = M)
#save the distance between the curves
area_between_curves<-vector(mode = "numeric", length = M)
area_between_curvesUnStab<-vector(mode = "numeric", length = M)
area_between_curvesStab<-vector(mode = "numeric", length = M)
#save the max distance between the curves
max_dist<-vector(mode = "numeric", length = M)
index_max<-vector(mode = "numeric", length = M)
surv.IPCW.UnStab <- matrix(nrow = M, ncol = length(tt))
surv.IPCW.Stab <- matrix(nrow = M, ncol = length(tt))

for(j in 1:M)
{

  data.sim1<-simweib(N,data,lamdax,lamdac,beta1,beta2,phi1,phi2,nx,nc)
  data.sim2<-data.sim1[data.sim1$status==1,]
  data.sim1$one <- 1
  survtrue <- survfit(Surv(Tx, one) ~ 1, data = data.sim1)
  surv<-survfit(Surv(time,status)~1,data=data.sim1)
  ssf <- summary( survtrue, times=tt,extend = TRUE)
  ssf1<- summary(surv, times=tt,extend = TRUE)
  surv.real[j,]<-ssf$surv
  surv.cens[j,]<-ssf1$surv
  cens.perc[j] <- 1 - sum(data.sim1$status)/N
  correlationXC[j] <- cor(data.sim1$Tx, data.sim1$Tc)
  max_dist[j]<-max(abs(surv.real[j,]-surv.cens[j,]))
  index_max[j]<-which.max(abs(surv.real[j,]-surv.cens[j,]))
area_between_curves[j]<-(geiger:::.area.between.curves(tt,surv.real[j,],
surv.cens[j,], xrange = c(0,max(data.sim2$time))))/max(data.sim2$time)

 # Transform the data into a long format:
data.sim.long <- transform.data(data.sim1)


# Calculate the IPCW weights:
ipwStab <- ipwtm(
           exposure = censored,
           family = "survival",
           numerator = ~ 1,
           denominator = ~ x2,
           id = id,
           tstart = Tstart,
           timevar = time,
           type = "first",
           data = data.sim.long )

ipwUnStab <- ipwtm(
           exposure = censored,
           family = "survival",
           #numerator = ~ 1,
           denominator = ~ x2,
           id = id,
           tstart = Tstart,
           timevar = time,
           type = "first",
           data = data.sim.long )

data.sim.long $ipw_weights <- ipwUnStab$ipw.weights
data.sim.long $ipw_Stabweights <- ipwStab$ipw.weights


# Calculate survival probabilities for the testpersons:
surv.IPCW.Stab[j,] <- calc.surv.IPCW(Tstart = data.sim.long$Tstart,
Tstop = data.sim.long$time,
status = data.sim.long$status,
tt = tt,
IPCW.weights = data.sim.long $ipw_Stabweights,
data.long = data.sim.long)

## Unstabilized Weights ##

# Calculate survival probabilities:
surv.IPCW.UnStab[j,] <- calc.surv.IPCW(Tstart = data.sim.long$Tstart,
Tstop = data.sim.long$time,
status = data.sim.long$status,
tt = tt,
IPCW.weights = data.sim.long $ipw_weights,
data.long = data.sim.long)


area_between_curvesStab[j]<-(geiger:::.area.between.curves(tt,surv.real[j,],
surv.IPCW.Stab[j,], xrange = c(0,max(data.sim2$time))))/max(data.sim2$time)
area_between_curvesUnStab[j]<-(geiger:::.area.between.curves(tt,surv.real[j,
surv.IPCW.UnStab[j,], xrange = c(0,max(data.sim2$time))))/max(data.sim2$time




}
return(list(surv.real=surv.real,
              surv.cens=surv.cens,
              cens.perc=cens.perc,
              max_dist=max_dist,
              area_between_curves=area_between_curves,
              area_between_curvesStab=area_between_curvesStab,
              area_between_curvesUnStab=area_between_curvesUnStab,
              correlationXC=correlationXC,
            index_max=index_max,surv.IPCW.UnStab=surv.IPCW.UnStab,
            surv.IPCW.Stab=surv.IPCW.Stab
            ))



}

## Scenario 1:
set.seed(123)
scenario1Fit  <- simIterations(M=M, #number of iterations
                         tt=tt, # time grid
                         N=N, # sample size
                         # Parameters survival model
                         lamdax=0.1, # ... .
                         beta1=0.5, # ...
                         beta2=0.1,
                         nx=2,
                         # Parameters censoring model
                         lamdac=0.027,
                         phi1=1.5,
                         phi2=0.5,
                         nc=2,
                         data=data

  )

surv.real.MC_scenario <- colMeans(scenario1Fit[["surv.real"]])
surv.cens.MC_scenario <- colMeans(scenario1Fit[["surv.cens"]])
surv.IPCW.UnStab.MC_scenario<-colMeans(scenario1Fit[["surv.IPCW.UnStab"]])
surv.IPCW.Stab.MC_scenario<-colMeans(scenario1Fit[["surv.IPCW.Stab"]])
cens.perc.MC<-mean(scenario1Fit[["cens.perc"]])
cens.perc.MC # percentage of censoring
area_between_curves.MC<-mean(scenario1Fit[["area_between_curves"]])
area_between_curves.MC.UnStab<-mean(scenario1Fit[["area_between_curvesUnStab
area_between_curves.MC.Stab<-mean(scenario1Fit[["area_between_curvesStab"]])
max_dist.MC<-mean(scenario1Fit[["max_dist"]])
area_between_curves.MC ## area
max_dist.MC ## max distance
correlationXC.MC<-mean(scenario1Fit[["correlationXC"]])
correlationXC.MC ## cor

plot(tt, surv.real.MC_scenario,
     xlab = "time", ylab = "Survival␣Probability",
     type="l", xlim=c(0,8), ylim=c(0,1), col = "blue",
     cex = 1.2,
     cex.main = 1.2,
     cex.axis = 1.2,
     cex.lab=1.2
)
lines(tt, surv.cens.MC_scenario, type = "l",lty=2,col = "black", lwd = 1.5)
lines(tt, surv.IPCW.UnStab.MC_scenario, type = "l",lty=3,
col = "red", lwd = 1.5)
lines(tt, surv.IPCW.Stab.MC_scenario , type = "l",lty=4,
col = "green", lwd = 1.5)
legend(legend = c("Real",
                  "Standard␣method","UnStabW","StabW"),
       col = c("blue", "black","red","green"),
       lty = c(1,2,3,4),
       x = "topright",
       cex = 1.2,
       bty = "n"
)


table<-matrix(NA,nrow=3,ncol = 1)
rownames(table)<-c("Standard␣method","UnStabW","StabW")
colnames(table)<-c("Area␣between␣the␣curves")
table[,1]<-c(area_between_curves.MC,area_between_curves.MC.UnStab,
area_between_curves.MC.Stab)

## Remove x2~bio
simIterations <- function(# Parameters simulation:
                            M, #number of iterations
                            tt, # time grid
                            N, # sample size
                            # Parameters survival model
                            lamdax, # ... .
                            beta1, # ...
                            beta2,
                            nx,
                            # Parameters censoring model
                            lamdac,
                            phi1,
                            phi2,
                            nc,
                            data

  )
{

surv.real <- matrix(nrow = M, ncol = length(tt))
surv.cens <- matrix(nrow = M, ncol = length(tt))
# Save censoring percentage in:
cens.perc <- vector(mode = "numeric", length = M)
# Save correlation between X and C in:
correlationXC <- vector(mode = "numeric", length = M)
#save the distance between the curves
area_between_curves<-vector(mode = "numeric", length = M)
area_between_curvesUnStab<-vector(mode = "numeric", length = M)
area_between_curvesStab<-vector(mode = "numeric", length = M)
#save the max distance between the curves
max_dist<-vector(mode = "numeric", length = M)
```

```
# index of max
index_max<-vector(mode = "numeric", length = M)
surv.IPCW.UnStab <- matrix(nrow = M, ncol = length(tt))
surv.IPCW.Stab <- matrix(nrow = M, ncol = length(tt))

for(j in 1:M)
{

  data.sim1<-simweib(N,data,lamdax,lamdac,beta1,beta2,phi1,phi2,nx,nc)
  data.sim2<-data.sim1[data.sim1$status==1,]
  data.sim1$one <- 1
  survtrue <- survfit(Surv(Tx, one) ~ 1, data = data.sim1)
  surv<-survfit(Surv(time,status)~1,data=data.sim1)
  ssf <- summary( survtrue, times=tt,extend = TRUE)
  ssf1<- summary(surv, times=tt,extend = TRUE)
  surv.real[j,]<-ssf$surv
  surv.cens[j,]<-ssf1$surv
  cens.perc[j] <- 1 - sum(data.sim1$status)/N
  correlationXC[j] <- cor(data.sim1$Tx, data.sim1$Tc)
  max_dist[j]<-max(abs(surv.real[j,]-surv.cens[j,]))
  index_max[j]<-which.max(abs(surv.real[j,]-surv.cens[j,]))
area_between_curves[j]<-(geiger:::.area.between.curves(tt,surv.real[j,],
surv.cens[j,], xrange = c(0,max(data.sim2$time))))/max(data.sim2$time)

 # Transform the data into a long format:
data.sim.long <- transform.data(data.sim1)


# Calculate the IPCW weights:
ipwStab <- ipwtm(
           exposure = censored,
           family = "survival",
           numerator = ~ 1,
           denominator = ~ x1,
           id = id,
           tstart = Tstart,
           timevar = time,
           type = "first",
           data = data.sim.long )

ipwUnStab <- ipwtm(
           exposure = censored,
           family = "survival",
           #numerator = ~ 1,
           denominator = ~ x1,
           id = id,
           tstart = Tstart,
           timevar = time,
           type = "first",
           data = data.sim.long )
```

```
data.sim.long $ipw_weights <- ipwUnStab$ipw.weights
data.sim.long $ipw_Stabweights <- ipwStab$ipw.weights


# Calculate survival probabilities for the testpersons:
surv.IPCW.Stab[j,] <- calc.surv.IPCW(Tstart = data.sim.long$Tstart,
Tstop = data.sim.long$time,
status = data.sim.long$status,
tt = tt,
IPCW.weights = data.sim.long $ipw_Stabweights,
data.long = data.sim.long)

## Unstabilized Weights ##

# Calculate survival probabilities:
surv.IPCW.UnStab[j,] <- calc.surv.IPCW(Tstart = data.sim.long$Tstart,
Tstop = data.sim.long$time,
status = data.sim.long$status,
tt = tt,
IPCW.weights = data.sim.long $ipw_weights,
data.long = data.sim.long)


area_between_curvesStab[j]<-(geiger:::.area.between.curves(tt,surv.real[j,],
surv.IPCW.Stab[j,], xrange = c(0,max(data.sim2$time))))/max(data.sim2$time)
area_between_curvesUnStab[j]<-(geiger:::.area.between.curves(tt,surv.real[j,
surv.IPCW.UnStab[j,], xrange = c(0,max(data.sim2$time))))/max(data.sim2$time




}
return(list(surv.real=surv.real,
              surv.cens=surv.cens,
              cens.perc=cens.perc,
              max_dist=max_dist,
              area_between_curves=area_between_curves,
              area_between_curvesStab=area_between_curvesStab,
              area_between_curvesUnStab=area_between_curvesUnStab,
              correlationXC=correlationXC,
            index_max=index_max,surv.IPCW.UnStab=surv.IPCW.UnStab,
            surv.IPCW.Stab=surv.IPCW.Stab
            ))




}


## Scenario 2:
set.seed(123)
```

```
scenario2Fit  <- simIterations(M=M, #number of iterations
                              tt=tt, # time grid
                              N=N, # sample size
                              # Parameters survival model
                              lamdax=0.1, # ... .
                              beta1=0.5, # ...
                              beta2=0.1,
                              nx=2,
                              # Parameters censoring model
                              lamdac=0.027,
                              phi1=1.5,
                              phi2=0.5,
                              nc=2,
                              data=data

  )


surv.real.MC_scenario <- colMeans(scenario2Fit[["surv.real"]])
surv.cens.MC_scenario <- colMeans(scenario2Fit[["surv.cens"]])
surv.IPCW.UnStab.MC_scenario<-colMeans(scenario2Fit[["surv.IPCW.UnStab"]])
surv.IPCW.Stab.MC_scenario<-colMeans(scenario2Fit[["surv.IPCW.Stab"]])
cens.perc.MC<-mean(scenario2Fit[["cens.perc"]])
cens.perc.MC # percentage of censoring
area_between_curves.MC<-mean(scenario2Fit[["area_between_curves"]])
area_between_curves.MC.UnStab<-mean(scenario2Fit[["area_between_curvesUnStab
area_between_curves.MC.Stab<-mean(scenario2Fit[["area_between_curvesStab"]])
max_dist.MC<-mean(scenario2Fit[["max_dist"]])
area_between_curves.MC ## area
max_dist.MC ## max distance
correlationXC.MC<-mean(scenario2Fit[["correlationXC"]])
correlationXC.MC ## cor

plot(tt, surv.real.MC_scenario,
     xlab = "time", ylab = "Survival␣Probability",
     type="l", xlim=c(0,8), ylim=c(0,1), col = "blue",
     cex = 1.2,
     cex.main = 1.2,
     cex.axis = 1.2,
     cex.lab=1.2
)
lines(tt, surv.cens.MC_scenario, type = "l",lty=2,col = "black", lwd = 1.5)
lines(tt, surv.IPCW.UnStab.MC_scenario, type = "l",lty=3,
col = "red", lwd = 1.5)
lines(tt, surv.IPCW.Stab.MC_scenario, type = "l",lty=4,
col = "green", lwd = 1.5)
legend(legend = c("Real",
                 "Standard␣method","UnStabW","StabW"),
       col = c("blue", "black","red","green"),
       lty = c(1,2,3,4),
       x = "topright",
       cex = 1.2,
       bty = "n"
)


table<-matrix(NA,nrow=3,ncol = 1)
rownames(table)<-c("Standard␣method","UnStabW","StabW")
colnames(table)<-c("Area␣between␣the␣curves")
table[,1]<-c(area_between_curves.MC,area_between_curves.MC.UnStab,
area_between_curves.MC.Stab)
```

B.4 R code chapter 6

```
 ## Load packages

library(MCMCpack)
library(stats)
library(Rlab)
library(MASS)
library(Matrix)
library(mvtnorm)
library(survival)
library(ggplot2)
library(coda)
library(lattice)
library(boa)
library(nlme)
library(car)
library(JM)
library("lattice")
library(simsurv)
library(ranger)
library(dplyr)
library(survminer)
library(kableExtra)
library(knitr)
library(stats)
library(tcltk)
library(ipw)

# Functions

### Lamberts W Function ###  Ngwa et al.(2019)

  LambertW=function(z, b=0,maxiter=10,eps=.Machine$double.eps,
                  min.imag = 1e-9) {
    if (any(round(Re(b)) != b))
      stop("branch␣number␣for␣W␣must␣be␣an␣integer")
    if (!is.complex(z) && any(z<0)) z=as.complex(z)
    ## series expansion about -1/e
    ##
    ## p = (1 - 2*abs(b)).*sqrt(2*e*z+2);
    ## w = (11/72)*p;
    ## w = (w - 1/3).*p;
    ## w = (w+1).*p - 1
    ##
    ## first-order version suffices:
    ##
    w = (1 - 2*abs(b))*sqrt(2*exp(1)*z+2) - 1
    ## asymptotic expansion at 0 and Inf
    ##
    v=log(z+as.numeric(z==0 & b==0)) + 2*pi*b*1i;
    v=v - log(v+as.numeric(v==0))
    ## choose strategy for initial guess
    ##
    c=abs(z+exp(-1));
    c = (c > 1.45 - 1.1*abs(b));
    c=c | (b*Im(z) > 0) | (!Im(z) & (b == 1))
    w = (1 - c)*w+c*v
    ## Halley iteration
    ##
    for (n in 1:maxiter) {
      p=exp(w)
      t=w*p - z
      f = (w != -1)
      t=f*t/(p*(w+f) - 0.5*(w+2.0)*t/(w+f))
      w=w - t
      if (abs(Re(t)) < (2.48*eps)*(1.0+abs(Re(w)))
          && abs(Im(t)) < (2.48*eps)*(1.0+abs(Im(w))))
        break
    }
    if (n==maxiter) warning(paste("iteration␣limit␣(",maxiter,")
reached,␣result␣of␣W␣may␣be␣inaccurate", sep=""))
    if (all(Im(w) < min.imag)) w=as.numeric(w)
    return(w)
  }

 ## simTIMEexp

simTIMEexp<-function(Survweib,Censoring)
  {
    time<-pmin(Survweib, Censoring)
    status<-as.numeric(Survweib <= Censoring)
    data.frame(id=1:N,Tx=Survweib,Tc=Censoring,time=time,status=status)
  }


## transform the data

transform.data <- function(LongSurv,Y)
```

```
{



## transform Y(matrix with the longtidutinal measurements) into long format
Y.long<-reshape(Y, sep = "", times = c(0,1,2,3,4,5), direction = "long",
                varying = c("Y1", "Y2", "Y3", "Y4", "Y5", "Y6"))
## adjust time from 1:6 to 0:5
Y.long$time<-Y.long$time-1

# add censored status
LongSurv$censored <- 1 - LongSurv$status

## add Tstart in the longsurv dataframe
df1<-tmerge(LongSurv,LongSurv,id=id,
            event=event(time,status))



## merge Y.long and df1
df1 <- tmerge(df1,Y.long,id=id,
              biomarker=tdc(time,Y))


# use the tmerge function again with status=censored to fix the censored
# status in the df1 dataset

# merge data with censored
df1_censored<-tmerge(LongSurv,LongSurv,id=id,
            censored=event(time,censored))
# merge Y.long and df1
df1_censored <- tmerge(df1_censored,Y.long,id=id,
              biomarker=tdc(time,Y))

if(all( df1$id == df1_censored$id &
        df1$tstart == df1_censored$tstart &
        df1$tstop == df1_censored$tstop))
{
  df1$censored <- df1_censored$censored
}


cuttimes <- unique(df1$tstop)

# LONG FORMAT WITH CORRECT EVENT STATUS :

df1_long <- survSplit(df1,
                      cut = cuttimes,
                      start = "tstart",
                      end = "tstop",
                      event = "event",
                        id="ID")
df1_long <- df1_long[order(df1_long$id,
                           df1_long$tstart),]

# LONG FORMAT WITH CORRECT CENSORING STATUS

df1_long_censored <- survSplit(df1,
                               cut = cuttimes,
                               start = "tstart",
                               end = "tstop",
                               event = "censored",
                               id="ID")

df1_long_censored <- df1_long_censored[order(df1_long_censored$id,
                                             df1_long_censored$tstart),]

if(all( df1_long$id == df1_long_censored$id &
        df1_long$tstart == df1_long_censored$tstart &
        df1_long$tstop == df1_long_censored$tstop))
{
  df1_long$censored <- df1_long_censored$censored
}




return(df1_long)

}

# Function to estimate the survival curve (Product-Limit Estimator)
# with weighted subjects.

calc.surv.IPCW <- function(Tstart, Tstop, status, tt,
IPCW.weights,data.long)
{
# Fit the Product-Limit estimator with weighted subjects
surv.IPCW <- survfit(Surv(Tstart, Tstop, event) ~ 1, data = data.long,
weights = IPCW.weights)
# Estimate survival probabilities at time points tt
ssurv.IPCW <- summary(surv.IPCW, times=tt, extend = TRUE)
# Return resulting survival probabilities:
return(ssurv.IPCW$surv)
}


## Simulation iterations
simIterations <- function(# Parameters simulation:
                          K, #number of iterations
                          tt, # time grid
                          N, # sample size
```

```
                             M, # ...
                             Time,
                             # Parameters survival model
                             link, # ... .
                             biobeta, # ...
                             Treatbeta,
                             wshape,
                             scale,
                             # Parameters censoring model
                             link1,
                             biobeta1,
                             Treatbeta1,
                             wshape1,
                             scale1
  )
{

  ## Save results of each iteration in one vector:

  surv.real <- matrix(nrow = K, ncol = length(tt))
  surv.cens <- matrix(nrow = K, ncol = length(tt))
  # Save censoring percentage in:
  cens.perc <- vector(mode = "numeric", length = K)
  #save the distance between the curves
  area_between_curves<-vector(mode = "numeric", length = K)
  area_between_curvesUnStab<-vector(mode = "numeric", length = M)
  area_between_curvesStab<-vector(mode = "numeric", length = M)
  #save the max distance between the curves
  max_dist<-vector(mode = "numeric", length = K)
  # Save correlation between X and C in:
  correlationXC <- vector(mode = "numeric", length = K)
  # index of max
  index_max<-vector(mode = "numeric", length = K)
  surv.IPCW.UnStab <- matrix(nrow = K, ncol = length(tt))
  surv.IPCW.Stab <- matrix(nrow = K, ncol = length(tt))
  phi.IPCW <- matrix(nrow = K, ncol = 3)

  for(k in 1:K){

    ### Creating Random Effects ###
    Ubeta=c(2, 1) ## fixed effects
    G=structure(.Data=c(0.5, -0.005, -0.005, 0.001), .Dim= c(2, 2)) ##

    UY=rmvnorm(N, mean=Ubeta, sigma=G, method = "chol") # fixed + random
    # effects ~ N(Ubeta,G)
    Z=matrix(0,M,length(dim(G)))
    Z[,1] <- 1
    Z[,2] <- c(Time)
    muy <- matrix(0,N,M)
    Y <- matrix(0,N,M)
    R <- diag(rnorm(M,1.000,0.00001), M)
```

```
V=Z%*%G%*%t(Z) + R #variance-covariance matrix of the repeated measures

### Covariate Specification ###

SEX=rbern(N,0.540)
TREATMENT=sample(c(0,1),size = N,replace=TRUE,prob = c(0.5,0.5))
for (i in 1:N){if (SEX[i] == 0) SEX[i]=1 else SEX[i]=2 }
x2<-round(rnorm(N,mean = 50,sd=10))
Bio<-(x2 - 50)/10 #

### Generating the Trajectories Outcome ###

for (j in 1:M){
  for (i in 1:N){
    muy[i, j] <- UY[i,1] + UY[i,2]*(Time[j])
  }
}
muy <- data.frame(muy)
names(muy) <- c("X1", "X2", "X3", "X4", "X5", "X6")

### Simulating Y Values ###

Y <- rmvnorm(n=N, mean=colMeans(muy), sigma=V, method ="chol")
Y <- data.frame(Y)
names(Y) <- c("Y1", "Y2", "Y3", "Y4", "Y5", "Y6")
Error <- rnorm(N, 0, 0.001)
for (i in 1:N){
  Y[i, ] <- Y[i, ] + Error[i]
}


### Linear Mixed Model Fit ###

Timeinterval <- rep(Time, N)
YSimul <- c(t(as.matrix(Y)))
ID <- rep(1:N, each=M)
bioR <- rep(Bio, each=M)
Long <- data.frame(ID, YSimul, Timeinterval, bioR)
LME <- lme(YSimul~Timeinterval, random = ~ Timeinterval | ID,
         data=Long, control=lmeControl(msMaxIter = 100, msVerbose=TRUE,
                                         opt = "optim"))
U <- coef(LME)
#U1<-random.effects(LME)
names(U) <- c("X1_1", "X2_1")
LME_Coeff <- data.frame(coefficients(LME))
colMeans(LME_Coeff)


### Survival Times Using Weibull Distribution ###

# # Lambda (Scale) = (1/Lambda)*v
```

```
    Uni <- runif(N, min = 0, max = 1)
    Numweib <- -log(Uni)
    Denweib <- scale*exp(biobeta*Bio+Treatbeta*TREATMENT+link*(U[,1]))
    ratioweib <- link*(U[,2])*(1/wshape)*((Numweib/Denweib)^(1/wshape))
    Lweib <- LambertW(ratioweib)
    rLweib<-Re(Lweib)
    Survweib <- Lweib*1/(link*(U[,2])*(1/wshape))

    ### Censoring Times Using Weibull Distribution ###

    # Lambda (Scale) = (1/Lambda)*v


    Uni <- runif(N, min = 0, max = 1)
    Numweib <- -log(Uni)
    Denweib <- scale1*exp(biobeta1*Bio+Treatbeta1*TREATMENT+link1*(U[,1]))
    ratioweib <- link1*(U[,2])*(1/wshape1)*((Numweib/Denweib)^(1/wshape1))
    Lweib <- LambertW(ratioweib)
    rLweib<-Re(Lweib)
    Censoring<- Lweib*1/(link1*(U[,2])*(1/wshape))
    #hist(Censoring)

    data.sim1<-simTIMEexp(Survweib,Censoring)
    data.sim1$one<- 1
    data.sim2<-data.sim1[data.sim1$status==1,]
    survtrue <- survfit(Surv(Tx, one) ~ 1, data = data.sim1)
    ssf <- summary( survtrue, times=tt,extend = TRUE)


    surv<-survfit(Surv(time,status)~1,data=data.sim1)
    ssf1<- summary(surv, times=tt,extend = TRUE)
    surv.real[k,]<-ssf$surv
    surv.cens[k,]<-ssf1$surv
    cens.perc[k] <- 1 - sum(data.sim1$status)/500
    correlationXC[k] <- cor(data.sim1$Tx, data.sim1$Tc)
    max_dist[k]<-max(abs(surv.real[k,]-surv.cens[k,]))
     index_max[k]<-which.max(abs(surv.real[k,]-surv.cens[k,]))
   area_between_curves[k]<-(geiger:::.area.between.curves(tt,surv.real[k,],
   surv.cens[k,],xrange = c(0,max(data.sim2$time))))/max(data.sim2$time)
   LongSurv <- data.frame( data.sim1,TREATMENT,Bio)
   LongSurv<-LongSurv[-c(2,3,6)]

   # Transform the data into a long format
   data.sim.long <- transform.data( LongSurv,Y)
   # Calculate the IPW weights:

  temp.unstab <- ipwtm(
  exposure = censored,
  family = "survival",
  denominator = ~TREATMENT + Bio + biomarker,
  id = id,
  tstart = tstart,
  timevar = tstop,
  type = "first",
  data = data.sim.long)

data.sim.long$ipwunstab<-temp.unstab$ipw.weights

temp.stab <- ipwtm(
  exposure = censored,
  family = "survival",
  numerator = ~ 1 ,
  denominator = ~ TREATMENT + Bio + biomarker,
  id = id,
  tstart = tstart,
  timevar = tstop,
  type = "first",
  data = data.sim.long)

data.sim.long$ipwstab<-temp.stab$ipw.weights

# Calculate survival probabilities:
surv.IPCW.UnStab[k,] <- calc.surv.IPCW(Tstart = data.sim.long$tstart,
Tstop = data.sim.long$tstop,
status = data.sim.long$event,
tt = tt,
IPCW.weights = data.sim.long$ipwunstab,
data.long = data.sim.long)

surv.IPCW.Stab[k,] <- calc.surv.IPCW(Tstart = data.sim.long$tstart,
Tstop = data.sim.long$tstop,
status = data.sim.long$event,
tt = tt,
IPCW.weights = data.sim.long$ipwstab,
data.long = data.sim.long)


CZ <- coxph(Surv(tstart,
                 tstop,
                 censored) ~ TREATMENT+Bio+biomarker,
            data = data.sim.long)

phi.IPCW[k,] <- summary(CZ)$coef[,1]

area_between_curvesStab[k]<-(geiger:::.area.between.curves(tt,surv.real[k,],
surv.IPCW.Stab[k,],xrange = c(0,max(data.sim2$time))))/max(data.sim2$time)
area_between_curvesUnStab[k]<-(geiger:::.area.between.curves(tt,surv.real[k,
surv.IPCW.UnStab[k,],xrange = c(0,max(data.sim2$time))))/max(data.sim2$time)
```

```
  }

 return(list(surv.real=surv.real,
             surv.cens=surv.cens,
             cens.perc=cens.perc,
             max_dist=max_dist,
             area_between_curves=area_between_curves,
            area_between_curvesStab=area_between_curvesStab,
             area_between_curvesUnStab=area_between_curvesUnStab,
             correlationXC=correlationXC,
           index_max=index_max,surv.IPCW.UnStab=surv.IPCW.UnStab,
           surv.IPCW.Stab=surv.IPCW.Stab,
           phi.IPCW =phi.IPCW
           ))
}

# Fit models for different scenarios

## Define parameters that should be the same in each scenario


K <-50 #number of iterations
tt <- c(seq(0,6,by=0.1)) # time grid
N <- 500 # sample size
M <- 6 #
Time <- c(0:5)

## Scenario 1:   Baseline phi1,phi2=(0.1,0.5) gammac=0.5

set.seed(123) # save seed
scenario1Fit <- simIterations(K = K,
                              tt = tt ,
                              N = N,
                              M = M,
                              Time = Time,
                              #Parameters survival model
                              link = 0.500,
                              biobeta = 0.1,
                              Treatbeta = 0.5,
                              wshape = 2,
                              scale = 0.01,
                              # Parameters censoring model
                              link1 = 0.5,
                              biobeta1 = 0.1,
                              Treatbeta1 = 0.5,
                              wshape1 = 2,
                              scale1 = 0.005

)

# Summarize results:
```

```
surv.real.MC_scenario1 <- colMeans(scenario1Fit[["surv.real"]])
surv.cens.MC_scenario1 <- colMeans(scenario1Fit[["surv.cens"]])
surv.IPCW.UnStab.MC_scenario<-colMeans(scenario1Fit[["surv.IPCW.UnStab"]])
surv.IPCW.Stab.MC_scenario<-colMeans(scenario1Fit[["surv.IPCW.Stab"]])
cens.perc.MC<-mean(scenario1Fit[["cens.perc"]])
cens.perc.MC # percentage of censoring
area_between_curves.MC<-mean(scenario1Fit[["area_between_curves"]])
area_between_curves.MC.UnStab<-mean(scenario1Fit[["area_between_curvesUnStab
area_between_curves.MC.Stab<-mean(scenario1Fit[["area_between_curvesStab"]])
max_dist.MC1<-mean(scenario1Fit[["max_dist"]])
area_between_curves.MC ## area
max_dist.MC1 ## max distance
correlationXC.MC1<-mean(scenario1Fit[["correlationXC"]])
correlationXC.MC1 ## cor

plot(tt, surv.real.MC_scenario1,
     xlab = "time", ylab = "Survival␣Probability",
     type="l", xlim=c(0,6), ylim=c(0,1), col = "blue",
     cex = 1.2,
     cex.main = 1.2,
     cex.axis = 1.2,
     cex.lab=1.2
)
lines(tt, surv.cens.MC_scenario1, type = "l",lty=2,col = "black", lwd = 1.5)
lines(tt, surv.IPCW.UnStab.MC_scenario, type = "l",lty=3,
col = "red", lwd = 1.5)
lines(tt, surv.IPCW.Stab.MC_scenario, type = "l",lty=4,
col = "green", lwd = 1.5)
legend(legend = c("Real",
                  "Standard␣method","UnStabW","StabW"),
       col = c("blue", "black","red","green"),
       lty = c(1,2,3,4),
       x = "topright",
       cex = 1.2,
       bty = "n"
)

colMeans(scenario1Fit[["phi.IPCW"]])

table<-matrix(NA,nrow=3,ncol = 1)
rownames(table)<-c("Standard␣method","UnStabW","StabW")
colnames(table)<-c("Area␣between␣the␣curves")
table[,1]<-c(area_between_curves.MC,area_between_curves.MC.UnStab,
area_between_curves.MC.Stab)



## scenario 3 10% censoring

set.seed(123) # save seed
scenario3Fit <- simIterations(K = K,
                              tt = tt ,
                              N = N,
                              M = M,
                              Time = Time,
                              #Parameters survival model
                              link = 0.500,
                              biobeta = 0.1,
                              Treatbeta = 0.5,
                              wshape = 2,
                              scale = 0.01,
                              # Parameters censoring model
                              link1 = 0.5,
                              biobeta1 = 0.1,
                              Treatbeta1 = 0.5,
                              wshape1 = 2,
                              scale1 = 0.0012

)

# Summarize results:
surv.real.MC_scenario3 <- colMeans(scenario3Fit[["surv.real"]])
surv.cens.MC_scenario3 <- colMeans(scenario3Fit[["surv.cens"]])
surv.IPCW.UnStab.MC_scenario<-colMeans(scenario3Fit[["surv.IPCW.UnStab"]])
surv.IPCW.Stab.MC_scenario<-colMeans(scenario3Fit[["surv.IPCW.Stab"]])
cens.perc.MC<-mean(scenario3Fit[["cens.perc"]])
cens.perc.MC # percentage of censoring
area_between_curves.MC<-mean(scenario3Fit[["area_between_curves"]])
area_between_curves.MC.UnStab<-mean(scenario3Fit[["area_between_curvesUnStab
area_between_curves.MC.Stab<-mean(scenario3Fit[["area_between_curvesStab"]])
max_dist.MC3<-mean(scenario3Fit[["max_dist"]])
area_between_curves.MC ## area
max_dist.MC3 ## max distance
correlationXC.MC3<-mean(scenario3Fit[["correlationXC"]])
correlationXC.MC3 ## cor

plot(tt, surv.real.MC_scenario3,
     xlab = "time", ylab = "Survival␣Probability",
     type="l", xlim=c(0,6), ylim=c(0,1), col = "blue",
     cex = 1.2,
     cex.main = 1.2,
     cex.axis = 1.2,
     cex.lab=1.2
)
lines(tt, surv.cens.MC_scenario3, type = "l",lty=2,col = "black", lwd = 1.5)
lines(tt, surv.IPCW.UnStab.MC_scenario, type = "l",lty=3,
col = "red", lwd = 1.5)
lines(tt, surv.IPCW.Stab.MC_scenario, type = "l",lty=4,
col = "green", lwd = 1.5)
legend(legend = c("Real",
                  "Standard␣method","UnStabW","StabW"),
       col = c("blue", "black","red","green"),
       lty = c(1,2,3,4),
       x = "topright",
       cex = 1.2,
       bty = "n"
)

colMeans(scenario3Fit[["phi.IPCW"]])
table<-matrix(NA,nrow=3,ncol = 1)
rownames(table)<-c("Standard␣method","UnStabW","StabW")
colnames(table)<-c("Area␣between␣the␣curves")
table[,1]<-c(area_between_curves.MC,area_between_curves.MC.UnStab,
area_between_curves.MC.Stab)

## scenario 4  (0.1,0.5) ,gamma=3

set.seed(123) # save seed
scenario4Fit <- simIterations(K = K,
                         tt = tt ,
                         N = N,
                         M = M,
                         Time = Time,
                         #Parameters survival model
                         link = 0.500,
                         biobeta = 0.1,
                         Treatbeta = 0.5,
                         wshape = 2,
                         scale = 0.01,
                         # Parameters censoring model
                         link1 = 3,
                         biobeta1 = 0.1,
                         Treatbeta1 = 0.5,
                         wshape1 = 2,
                         scale1 = 0.00000003

)

# Summarize results:
surv.real.MC_scenario4 <- colMeans(scenario4Fit[["surv.real"]])
surv.cens.MC_scenario4 <- colMeans(scenario4Fit[["surv.cens"]])
surv.IPCW.UnStab.MC_scenario<-colMeans(scenario4Fit[["surv.IPCW.UnStab"]])
surv.IPCW.Stab.MC_scenario<-colMeans(scenario4Fit[["surv.IPCW.Stab"]])
cens.perc.MC<-mean(scenario4Fit[["cens.perc"]])
cens.perc.MC # percentage of censoring
area_between_curves.MC<-mean(scenario4Fit[["area_between_curves"]])
area_between_curves.MC.UnStab<-mean(scenario4Fit[["area_between_curvesUnStab
area_between_curves.MC.Stab<-mean(scenario4Fit[["area_between_curvesStab"]])
max_dist.MC4<-mean(scenario4Fit[["max_dist"]])
area_between_curves.MC ## area
max_dist.MC4 ## max distance
correlationXC.MC4<-mean(scenario4Fit[["correlationXC"]])
correlationXC.MC4 ## cor
```

```
plot(tt, surv.real.MC_scenario4,
     xlab = "time", ylab = "Survival␣Probability",
     type="l", xlim=c(0,6), ylim=c(0,1), col = "blue",
     cex = 1.2,
     cex.main = 1.2,
     cex.axis = 1.2,
     cex.lab=1.2
)
lines(tt, surv.cens.MC_scenario4, type = "l",lty=2,col = "black", lwd = 1.5)
lines(tt, surv.IPCW.UnStab.MC_scenario, type = "l",lty=3,
col = "red", lwd = 1.5)
lines(tt, surv.IPCW.Stab.MC_scenario, type = "l",lty=4,
col = "green", lwd = 1.5)
legend(legend = c("Real",
                  "Standard␣method","UnStabW","StabW"),
       col = c("blue", "black","red","green"),
       lty = c(1,2,3,4),
       x = "topright",
       cex = 1.2,
       bty = "n"
)

colMeans(scenario4Fit[["phi.IPCW"]])

table<-matrix(NA,nrow=3,ncol = 1)
rownames(table)<-c("Standard␣method","UnStabW","StabW")
colnames(table)<-c("Area␣between␣the␣curves")
table[,1]<-c(area_between_curves.MC,area_between_curves.MC.UnStab,
area_between_curves.MC.Stab)

## scenario 5  (0.1,0.5) ,gamma=1.5

set.seed(123)
scenario5Fit <- simIterations(K = K,
                          tt = tt ,
                          N = N,
                          M = M,
                          Time = Time,
                          #Parameters survival model
                          link = 0.500,
                          biobeta = 0.1,
                          Treatbeta = 0.5,
                          wshape = 2,
                          scale = 0.01,
                          # Parameters censoring model
                          link1 = 1.5,
                          biobeta1 = 0.1,
                          Treatbeta1 = 0.5,
                          wshape1 = 2,
                          scale1 = 0.00004
```

```
)

# Summarize results:
surv.real.MC_scenario5 <- colMeans(scenario5Fit[["surv.real"]])
surv.cens.MC_scenario5 <- colMeans(scenario5Fit[["surv.cens"]])
surv.IPCW.UnStab.MC_scenario<-colMeans(scenario5Fit[["surv.IPCW.UnStab"]])
surv.IPCW.Stab.MC_scenario<-colMeans(scenario5Fit[["surv.IPCW.Stab"]])
cens.perc.MC<-mean(scenario5Fit[["cens.perc"]])
cens.perc.MC # percentage of censoring
area_between_curves.MC<-mean(scenario5Fit[["area_between_curves"]])
area_between_curves.MC.UnStab<-mean(scenario5Fit[["area_between_curvesUnStab
area_between_curves.MC.Stab<-mean(scenario5Fit[["area_between_curvesStab"]])
max_dist.MC5<-mean(scenario5Fit[["max_dist"]])
area_between_curves.MC ## area
max_dist.MC5 ## max distance
correlationXC.MC5<-mean(scenario5Fit[["correlationXC"]])
correlationXC.MC5 ## cor

plot(tt, surv.real.MC_scenario5,
     xlab = "time", ylab = "Survival␣Probability",
     type="l", xlim=c(0,6), ylim=c(0,1), col = "blue",
     cex = 1.2,
     cex.main = 1.2,
     cex.axis = 1.2,
     cex.lab=1.2
)
lines(tt, surv.cens.MC_scenario5, type = "l",lty=2,col = "black", lwd = 1.5)
lines(tt, surv.IPCW.UnStab.MC_scenario, type = "l",lty=3,
col = "red", lwd = 1.5)
lines(tt, surv.IPCW.Stab.MC_scenario, type = "l",lty=4,
col = "green", lwd = 1.5)
legend(legend = c("Real",
                 "Standard␣method","UnStabW","StabW"),
       col = c("blue", "black","red","green"),
       lty = c(1,2,3,4),
       x = "topright",
       cex = 1.2,
       bty = "n"
)

colMeans(scenario5Fit[["phi.IPCW"]])
table<-matrix(NA,nrow=3,ncol = 1)
rownames(table)<-c("Standard␣method","UnStabW","StabW")
colnames(table)<-c("Area␣between␣the␣curves")
table[,1]<-c(area_between_curves.MC,area_between_curves.MC.UnStab,
area_between_curves.MC.Stab)


 ## Try IPW method with only one time-fixed covariate
## Simulation iterations with only BIO

simIterations <- function(# Parameters simulation:
                          K, #number of iterations
                          tt, # time grid
                          N, # sample size
                          M, # ...
                          Time,
                          # Parameters survival model
                          link, # ... .
                          biobeta, # ...
                          Treatbeta,
                          wshape,
                          scale,
                          # Parameters censoring model
                          link1,
                          biobeta1,
                          Treatbeta1,
                          wshape1,
                          scale1
  )
{

  ## Save results of each iteration in one vector:

  surv.real <- matrix(nrow = K, ncol = length(tt))
  surv.cens <- matrix(nrow = K, ncol = length(tt))
  # Save censoring percentage in:
  cens.perc <- vector(mode = "numeric", length = K)
  #save the distance between the curves
  area_between_curves<-vector(mode = "numeric", length = K)
  area_between_curvesUnStab<-vector(mode = "numeric", length = M)
  area_between_curvesStab<-vector(mode = "numeric", length = M)
  #save the max distance between the curves
  max_dist<-vector(mode = "numeric", length = K)
  # Save correlation between X and C in:
  correlationXC <- vector(mode = "numeric", length = K)
  index_max<-vector(mode = "numeric", length = K)
  surv.IPCW.UnStab <- matrix(nrow = K, ncol = length(tt))
  surv.IPCW.Stab <- matrix(nrow = K, ncol = length(tt))
  phi.IPCW <- matrix(nrow = K, ncol = 3)

  for(k in 1:K){


    ### Creating Random Effects ###

    Ubeta=c(2, 1) ## fixed effects
    G=structure(.Data=c(0.5, -0.005, -0.005, 0.001), .Dim= c(2, 2)) ##

    UY=rmvnorm(N, mean=Ubeta, sigma=G, method = "chol") # fixed + random
```

```
# effects ~ N(Ubeta,G)
Z=matrix(0,M,length(dim(G)))
Z[,1] <- 1
Z[,2] <- c(Time)
muy <- matrix(0,N,M)
Y <- matrix(0,N,M)
R <- diag(rnorm(M,1.000,0.00001), M)
V=Z%*%G%*%t(Z) + R #variance-covariance matrix of the repeated measures


### Covariate Specification ###

SEX=rbern(N,0.540)
TREATMENT=sample(c(0,1),size = N,replace=TRUE,prob = c(0.5,0.5))
for (i in 1:N){if (SEX[i] == 0) SEX[i]=1 else SEX[i]=2 }
x2<-round(rnorm(N,mean = 50,sd=10))
Bio<-(x2 - 50)/10 #

### Generating the Trajectories Outcome ###

for (j in 1:M){
  for (i in 1:N){
    muy[i, j] <- UY[i,1] + UY[i,2]*(Time[j])
  }
}
muy <- data.frame(muy)
names(muy) <- c("X1", "X2", "X3", "X4", "X5", "X6")

### Simulating Y Values ###

Y <- rmvnorm(n=N, mean=colMeans(muy), sigma=V, method ="chol")
Y <- data.frame(Y)
names(Y) <- c("Y1", "Y2", "Y3", "Y4", "Y5", "Y6")
Error <- rnorm(N, 0, 0.001)
for (i in 1:N){
  Y[i, ] <- Y[i, ] + Error[i]
}


### Linear Mixed Model Fit ###

Timeinterval <- rep(Time, N)
YSimul <- c(t(as.matrix(Y)))
ID <- rep(1:N, each=M)
bioR <- rep(Bio, each=M)
Long <- data.frame(ID, YSimul, Timeinterval, bioR)
LME <- lme(YSimul~Timeinterval, random = ~ Timeinterval | ID,
     data=Long, control=lmeControl(msMaxIter = 100,msVerbose=TRUE,
                                        opt = "optim"))
U <- coef(LME)
#U1<-random.effects(LME)
```

```
 names(U) <- c("X1_1", "X2_1")
 LME_Coeff <- data.frame(coefficients(LME))
 colMeans(LME_Coeff)



 ### Survival Times Using Weibull Distribution ###


 Uni <- runif(N, min = 0, max = 1)
 Numweib <- -log(Uni)
 Denweib <- scale*exp(biobeta*Bio+Treatbeta*TREATMENT+link*(U[,1]))
 ratioweib <- link*(U[,2])*(1/wshape)*((Numweib/Denweib)^(1/wshape))
 Lweib <- LambertW(ratioweib)
 rLweib<-Re(Lweib)
 Survweib <- Lweib*1/(link*(U[,2])*(1/wshape))



 ### Censoring Times Using Weibull Distribution ###

 Uni <- runif(N, min = 0, max = 1)
 Numweib <- -log(Uni)
 Denweib <- scale1*exp(biobeta1*Bio+Treatbeta1*TREATMENT+link1*(U[,1]))
 ratioweib <- link1*(U[,2])*(1/wshape1)*((Numweib/Denweib)^(1/wshape1))
 Lweib <- LambertW(ratioweib)
 rLweib<-Re(Lweib)
 Censoring<- Lweib*1/(link1*(U[,2])*(1/wshape))

 data.sim1<-simTIMEexp(Survweib,Censoring)
 data.sim1$one<- 1
 data.sim2<-data.sim1[data.sim1$status==1,]
 survtrue <- survfit(Surv(Tx, one) ~ 1, data = data.sim1)
 ssf <- summary( survtrue, times=tt,extend = TRUE)


 surv<-survfit(Surv(time,status)~1,data=data.sim1)
 ssf1<- summary(surv, times=tt,extend = TRUE)
 surv.real[k,]<-ssf$surv
 surv.cens[k,]<-ssf1$surv
 cens.perc[k] <- 1 - sum(data.sim1$status)/500
 correlationXC[k] <- cor(data.sim1$Tx, data.sim1$Tc)
 max_dist[k]<-max(abs(surv.real[k,]-surv.cens[k,]))
 index_max[k]<-which.max(abs(surv.real[k,]-surv.cens[k,]))
area_between_curves[k]<-(geiger:::.area.between.curves(tt,surv.real[k,],
surv.cens[k,],xrange = c(0,max(data.sim2$time))))/max(data.sim2$time)
LongSurv <- data.frame( data.sim1,TREATMENT,Bio)
LongSurv<-LongSurv[-c(2,3,6)]

# Transform the data into a long format
data.sim.long <- transform.data( LongSurv,Y)
```

```
   # Calculate the IPW weights:

  temp.unstab <- ipwtm(
  exposure = censored,
  family = "survival",
  denominator = ~  Bio + biomarker,
  id = id,
  tstart = tstart,
  timevar = tstop,
  type = "first",
  data = data.sim.long)

data.sim.long$ipwunstab<-temp.unstab$ipw.weights

temp.stab <- ipwtm(
  exposure = censored,
  family = "survival",
  numerator = ~ 1 ,
  denominator = ~  Bio + biomarker,
  id = id,
  tstart = tstart,
  timevar = tstop,
  type = "first",
  data = data.sim.long)

data.sim.long$ipwstab<-temp.stab$ipw.weights

# Calculate survival probabilities:
surv.IPCW.UnStab[k,] <- calc.surv.IPCW(Tstart = data.sim.long$tstart,
Tstop = data.sim.long$tstop,
status = data.sim.long$event,
tt = tt,
IPCW.weights = data.sim.long$ipwunstab,
data.long = data.sim.long)

surv.IPCW.Stab[k,] <- calc.surv.IPCW(Tstart = data.sim.long$tstart,
Tstop = data.sim.long$tstop,
status = data.sim.long$event,
tt = tt,
IPCW.weights = data.sim.long$ipwstab,
data.long = data.sim.long)


CZ <- coxph(Surv(tstart,
                 tstop,
                 censored) ~ TREATMENT+Bio+biomarker,
            data = data.sim.long)

phi.IPCW[k,] <- summary(CZ)$coef[,1]

area_between_curvesStab[k]<-(geiger:::.area.between.curves(tt,surv.real[k,],
surv.IPCW.Stab[k,], xrange = c(0,max(data.sim2$time))))/max(data.sim2$time)
area_between_curvesUnStab[k]<-(geiger:::.area.between.curves(tt,surv.real[k,
surv.IPCW.UnStab[k,], xrange = c(0,max(data.sim2$time))))/max(data.sim2$time


  }

 return(list(surv.real=surv.real,
              surv.cens=surv.cens,
              cens.perc=cens.perc,
              max_dist=max_dist,
              area_between_curves=area_between_curves,
             area_between_curvesStab=area_between_curvesStab,
              area_between_curvesUnStab=area_between_curvesUnStab,
              correlationXC=correlationXC,
            index_max=index_max,surv.IPCW.UnStab=surv.IPCW.UnStab,
            surv.IPCW.Stab=surv.IPCW.Stab,
            phi.IPCW =phi.IPCW
            ))
}

## Scenario 1:   Baseline phi1,phi2=(0.1,0.5) gammac=0.5

set.seed(123)
scenario1Fit <- simIterations(K = K,
                          tt = tt ,
                          N = N,
                          M = M,
                          Time = Time,
                          #Parameters survival model
                          link = 0.500,
                          biobeta = 0.1,
                          Treatbeta = 0.5,
                          wshape = 2,
                          scale = 0.01,
                          # Parameters censoring model
                          link1 = 1.5,
                          biobeta1 = 0.1,
                          Treatbeta1 = 0.5,
                          wshape1 = 2,
                          scale1 = 0.00004

)

# Summarize results:
 surv.real.MC_scenario1 <- colMeans(scenario1Fit[["surv.real"]])
surv.cens.MC_scenario1 <- colMeans(scenario1Fit[["surv.cens"]])
surv.IPCW.UnStab.MC_scenario<-colMeans(scenario1Fit[["surv.IPCW.UnStab"]])
surv.IPCW.Stab.MC_scenario<-colMeans(scenario1Fit[["surv.IPCW.Stab"]])
cens.perc.MC<-mean(scenario1Fit[["cens.perc"]])
cens.perc.MC # percentage of censoring
```

```
area_between_curves.MC<-mean(scenario1Fit[["area_between_curves"]])
area_between_curves.MC.UnStab<-mean(scenario1Fit[["area_between_curvesUnStab
area_between_curves.MC.Stab<-mean(scenario1Fit[["area_between_curvesStab"]])
max_dist.MC1<-mean(scenario1Fit[["max_dist"]])
area_between_curves.MC## area
max_dist.MC1 ## max distance
correlationXC.MC1<-mean(scenario1Fit[["correlationXC"]])
correlationXC.MC1 ## cor

plot(tt, surv.real.MC_scenario1,
     xlab = "time", ylab = "Survival␣Probability",
     type="l", xlim=c(0,6), ylim=c(0,1), col = "blue",
     cex = 1.2,
     cex.main = 1.2,
     cex.axis = 1.2,
     cex.lab=0.8
)
lines(tt, surv.cens.MC_scenario1, type = "l",lty=2,col = "black", lwd = 1.5)
lines(tt, surv.IPCW.UnStab.MC_scenario, type = "l",lty=3,
col = "red", lwd = 1.5)
lines(tt, surv.IPCW.Stab.MC_scenario, type = "l",lty=4,
col = "green", lwd = 1.5)
legend(legend = c("Real",
                  "Standard␣method","UnStabW","StabW"),
       col = c("blue", "black","red","green"),
       lty = c(1,2,3,4),
       x = "topright",
       cex = 1.2,
       bty = "n"
)

colMeans(scenario1Fit[["phi.IPCW"]])

table<-matrix(NA,nrow=3,ncol = 1)
rownames(table)<-c("Standard␣method","UnStabW","StabW")
colnames(table)<-c("Area␣between␣the␣curves")
table[,1]<-c(area_between_curves.MC,area_between_curves.MC.UnStab,
area_between_curves.MC.Stab)


## Scenario 2
## Baseline phi1,phi2=(0.1,0.5) gammac=0.5 with only treatment

## Simulation iterations

simIterations <- function(# Parameters simulation:
                          K, #number of iterations
                          tt, # time grid
                          N, # sample size
                          M, # ...
                          Time,
                          # Parameters survival model
                          link, # ... .
                          biobeta, # ...
                          Treatbeta,
                          wshape,
                          scale,
                          # Parameters censoring model
                          link1,
                          biobeta1,
                          Treatbeta1,
                          wshape1,
                          scale1
 )
{

 ## Save results of each iteration in one vector:

 surv.real <- matrix(nrow = K, ncol = length(tt))
 surv.cens <- matrix(nrow = K, ncol = length(tt))
 # Save censoring percentage in:
 cens.perc <- vector(mode = "numeric", length = K)
 #save the distance between the curves
 area_between_curves<-vector(mode = "numeric", length = K)
 area_between_curvesUnStab<-vector(mode = "numeric", length = M)
 area_between_curvesStab<-vector(mode = "numeric", length = M)
 #save the max distance between the curves
 max_dist<-vector(mode = "numeric", length = K)
 # Save correlation between X and C in:
 correlationXC <- vector(mode = "numeric", length = K)
 # index of max
 index_max<-vector(mode = "numeric", length = K)
 surv.IPCW.UnStab <- matrix(nrow = K, ncol = length(tt))
 surv.IPCW.Stab <- matrix(nrow = K, ncol = length(tt))
 phi.IPCW <- matrix(nrow = K, ncol = 3)

 for(k in 1:K){


   ### Creating Random Effects ###

   Ubeta=c(2, 1) ## fixed effects
   G=structure(.Data=c(0.5, -0.005, -0.005, 0.001), .Dim= c(2, 2)) ##

   UY=rmvnorm(N, mean=Ubeta, sigma=G, method = "chol") # fixed + random
   # effects ~ N(Ubeta,G)
   Z=matrix(0,M,length(dim(G)))
   Z[,1] <- 1
   Z[,2] <- c(Time)
   muy <- matrix(0,N,M)
   Y <- matrix(0,N,M)
   R <- diag(rnorm(M,1.000,0.00001), M)
```

```
V=Z%*%G%*%t(Z) + R #variance-covariance matrix of the repeated measures


### Covariate Specification ###

SEX=rbern(N,0.540)
TREATMENT=sample(c(0,1),size = N,replace=TRUE,prob = c(0.5,0.5))
for (i in 1:N){if (SEX[i] == 0) SEX[i]=1 else SEX[i]=2 }
x2<-round(rnorm(N,mean = 50,sd=10))
Bio<-(x2 - 50)/10 #


### Generating the Trajectories Outcome ###

for (j in 1:M){
  for (i in 1:N){
    muy[i, j] <- UY[i,1] + UY[i,2]*(Time[j])
  }
}
muy <- data.frame(muy)
names(muy) <- c("X1", "X2", "X3", "X4", "X5", "X6")

### Simulating Y Values ###

Y <- rmvnorm(n=N, mean=colMeans(muy), sigma=V, method ="chol")
Y <- data.frame(Y)
names(Y) <- c("Y1", "Y2", "Y3", "Y4", "Y5", "Y6")
Error <- rnorm(N, 0, 0.001)
for (i in 1:N){
  Y[i, ] <- Y[i, ] + Error[i]
}

### Linear Mixed Model Fit ###

Timeinterval <- rep(Time, N)
YSimul <- c(t(as.matrix(Y)))
ID <- rep(1:N, each=M)
bioR <- rep(Bio, each=M)
Long <- data.frame(ID, YSimul, Timeinterval, bioR)
LME <- lme(YSimul~Timeinterval, random = ~ Timeinterval | ID,
      data=Long, control=lmeControl(msMaxIter = 100,msVerbose=TRUE,
                                    opt = "optim"))
U <- coef(LME)
#U1<-random.effects(LME)
names(U) <- c("X1_1", "X2_1")
LME_Coeff <- data.frame(coefficients(LME))
colMeans(LME_Coeff)


#### Specification of Parameter Estimates ###
```

```
  ### Survival Times Using Weibull Distribution ###


  Uni <- runif(N, min = 0, max = 1)
  Numweib <- -log(Uni)
  Denweib <- scale*exp(biobeta*Bio+Treatbeta*TREATMENT+link*(U[,1]))
  ratioweib <- link*(U[,2])*(1/wshape)*((Numweib/Denweib)^(1/wshape))
  Lweib <- LambertW(ratioweib)
  rLweib<-Re(Lweib)
  Survweib <- Lweib*1/(link*(U[,2])*(1/wshape))


  ### Censoring Times Using Weibull Distribution ###


  Uni <- runif(N, min = 0, max = 1)
  Numweib <- -log(Uni)
  Denweib <- scale1*exp(biobeta1*Bio+Treatbeta1*TREATMENT+link1*(U[,1]))
  ratioweib <- link1*(U[,2])*(1/wshape1)*((Numweib/Denweib)^(1/wshape1))
  Lweib <- LambertW(ratioweib)
  rLweib<-Re(Lweib)
  Censoring<- Lweib*1/(link1*(U[,2])*(1/wshape))
  #hist(Censoring)

  data.sim1<-simTIMEexp(Survweib,Censoring)
  data.sim1$one<- 1
  data.sim2<-data.sim1[data.sim1$status==1,]
  survtrue <- survfit(Surv(Tx, one) ~ 1, data = data.sim1)
  ssf <- summary( survtrue, times=tt,extend = TRUE)


  surv<-survfit(Surv(time,status)~1,data=data.sim1)
  ssf1<- summary(surv, times=tt,extend = TRUE)
  surv.real[k,]<-ssf$surv
  surv.cens[k,]<-ssf1$surv
  cens.perc[k] <- 1 - sum(data.sim1$status)/500
  correlationXC[k] <- cor(data.sim1$Tx, data.sim1$Tc)
  max_dist[k]<-max(abs(surv.real[k,]-surv.cens[k,]))
  index_max[k]<-which.max(abs(surv.real[k,]-surv.cens[k,]))
 area_between_curves[k]<-(geiger:::.area.between.curves(tt,surv.real[k,],
 surv.cens[k,],xrange = c(0,max(data.sim2$time))))/max(data.sim2$time)
 LongSurv <- data.frame( data.sim1,TREATMENT,Bio)
 LongSurv<-LongSurv[-c(2,3,6)]

 # Transform the data into a long format
 data.sim.long <- transform.data( LongSurv,Y)
 # Calculate the IPW weights:

temp.unstab <- ipwtm(
exposure = censored,
  family = "survival",
  denominator = ~  TREATMENT + biomarker,
  id = id,
  tstart = tstart,
  timevar = tstop,
  type = "first",
  data = data.sim.long)

data.sim.long$ipwunstab<-temp.unstab$ipw.weights

temp.stab <- ipwtm(
  exposure = censored,
  family = "survival",
  numerator = ~ 1 ,
  denominator = ~  TREATMENT + biomarker,
  id = id,
  tstart = tstart,
  timevar = tstop,
  type = "first",
  data = data.sim.long)

data.sim.long$ipwstab<-temp.stab$ipw.weights

# Calculate survival probabilities:
surv.IPCW.UnStab[k,] <- calc.surv.IPCW(Tstart = data.sim.long$tstart,
Tstop = data.sim.long$tstop,
status = data.sim.long$event,
tt = tt,
IPCW.weights = data.sim.long$ipwunstab,
data.long = data.sim.long)

surv.IPCW.Stab[k,] <- calc.surv.IPCW(Tstart = data.sim.long$tstart,
Tstop = data.sim.long$tstop,
status = data.sim.long$event,
tt = tt,
IPCW.weights = data.sim.long$ipwstab,
data.long = data.sim.long)


CZ <- coxph(Surv(tstart,
                 tstop,
                 censored) ~ TREATMENT+Bio+biomarker,
            data = data.sim.long)

phi.IPCW[k,] <- summary(CZ)$coef[,1]

area_between_curvesStab[k]<-(geiger:::.area.between.curves(tt,surv.real[k,],
surv.IPCW.Stab[k,], xrange = c(0,max(data.sim2$time))))/max(data.sim2$time)
area_between_curvesUnStab[k]<-(geiger:::.area.between.curves(tt,surv.real[k,
surv.IPCW.UnStab[k,], xrange = c(0,max(data.sim2$time))))/max(data.sim2$time
```

```
  }

 return(list(surv.real=surv.real,
              surv.cens=surv.cens,
              cens.perc=cens.perc,
              max_dist=max_dist,
              area_between_curves=area_between_curves,
             area_between_curvesStab=area_between_curvesStab,
              area_between_curvesUnStab=area_between_curvesUnStab,
              correlationXC=correlationXC,
            index_max=index_max,surv.IPCW.UnStab=surv.IPCW.UnStab,
            surv.IPCW.Stab=surv.IPCW.Stab,
            phi.IPCW =phi.IPCW
            ))
}

set.seed(123)
scenario2Fit <- simIterations(K = K,
                          tt = tt ,
                          N = N,
                          M = M,
                          Time = Time,
                          #Parameters survival model
                          link = 0.500,
                          biobeta = 0.1,
                          Treatbeta = 0.5,
                          wshape = 2,
                          scale = 0.01,
                          # Parameters censoring model
                          link1 = 1.5,
                          biobeta1 = 0.1,
                          Treatbeta1 = 0.5,
                          wshape1 = 2,
                          scale1 = 0.00004

)

# Summarize results:
surv.real.MC_scenario2 <- colMeans(scenario2Fit[["surv.real"]])
surv.cens.MC_scenario2 <- colMeans(scenario2Fit[["surv.cens"]])
surv.IPCW.UnStab.MC_scenario<-colMeans(scenario2Fit[["surv.IPCW.UnStab"]])
surv.IPCW.Stab.MC_scenario<-colMeans(scenario2Fit[["surv.IPCW.Stab"]])
cens.perc.MC<-mean(scenario2Fit[["cens.perc"]])
cens.perc.MC # percentage of censoring
area_between_curves.MC<-mean(scenario2Fit[["area_between_curves"]])
area_between_curves.MC.UnStab<-mean(scenario2Fit[["area_between_curvesUnStab
area_between_curves.MC.Stab<-mean(scenario2Fit[["area_between_curvesStab"]])
max_dist.MC2<-mean(scenario2Fit[["max_dist"]])
area_between_curves.MC ## area
max_dist.MC2 ## max distance
```

```
correlationXC.MC2<-mean(scenario2Fit[["correlationXC"]])
correlationXC.MC2 ## cor

plot(tt, surv.real.MC_scenario2,
     xlab = "time", ylab = "Survival␣Probability",
     type="l", xlim=c(0,6), ylim=c(0,1), col = "blue",
     cex = 1.2,
     cex.main = 1.2,
     cex.axis = 1.2,
     cex.lab=0.8
)
lines(tt, surv.cens.MC_scenario2, type = "l",lty=2,col = "black", lwd = 1.5)
lines(tt, surv.IPCW.UnStab.MC_scenario, type = "l",lty=3,
col = "red", lwd = 1.5)
lines(tt, surv.IPCW.Stab.MC_scenario, type = "l",lty=4,
col = "green", lwd = 1.5)
legend(legend = c("Real",
                  "Standard␣method","UnStabW","StabW"),
       col = c("blue", "black","red","green"),
       lty = c(1,2,3,4),
       x = "topright",
       cex = 1.2,
       bty = "n"
)

colMeans(scenario2Fit[["phi.IPCW"]])

table<-matrix(NA,nrow=3,ncol = 1)
rownames(table)<-c("Standard␣method","UnStabW","StabW")
colnames(table)<-c("Area␣between␣the␣curves")
table[,1]<-c(area_between_curves.MC,area_between_curves.MC.UnStab,
area_between_curves.MC.Stab)
```