NTNU – Trondheim
Norwegian University of
Science and Technology

Modelling of Dependent Competing Risks and Semi-Competing Risks by Means of First Passage Times of Gamma Processes

Beate Sildnes

Master of Science in Physics and Mathematics
Submission date: June 2015
Supervisor: Bo Henry Lindqvist, MATH

Norwegian University of Science and Technology
Department of Mathematical Sciences

Problem description

- give an introduction to modelling and inference for competing and semi-competing risks
- study models for dependent competing risks and semi-competing risks based on first passage times of gamma processes
- apply models to both simulated and real data

Assignment given: January 16, 2015
Supervisor: Bo H. Lindqvist

Preface

This thesis concludes my Master of Science degree in Applied Physics and Mathematics with specialization in Industrial Mathematics. The work on the thesis has been carried out during my tenth and final semester at the Norwegian University of Science and Technology (NTNU), from January 2015 to June 2015. The thesis can be viewed as an extension of my Master's project from the fall semester of 2014.

I would like to express my sincere gratitude to my supervisor, Professor Bo H. Lindqvist at the Department of Mathematical Sciences at NTNU for all of his guidance and advice throughout the semester. His enthusiasm, ideas and comments have been invaluable.

Finally, I also wish to thank Simen Mikkelsen for always supporting and encouraging me.

Beate Sildnes

Trondheim, June 2015

Abstract

In this thesis we model both dependent competing risks and semi-competing risks by means of first passage times in a gamma process. In both cases, we consider a terminal event, such as death of a person or component failure, and a non-terminal event like for instance disease recurrence or preventive maintenance of a component. We let the time to the terminal event equal the first passage time to a fixed level c in a gamma process. The time to the non-terminal event is represented by the first passage time to a stochastic level S. We have assumed that S is independent of the gamma process so that we have random signs censoring.

In the competing risks case, a similar model based on Wiener processes has been considered before. For semi-competing risks this is a new modelling approach, as semi-competing risks data have mostly been analysed through copula models in the past. We conduct simulation studies that show that the parameters in the gamma process model can be estimated satisfactorily for both competing and semi-competing risks data. The model is also applied to real datasets and seems to be able to fit the data well, at least for certain chosen distributions of the random level S. It is particularly interesting to note that our results for semi-competing risks are consistent with earlier published results.

Sammendrag

I denne masteroppgaven modellerer vi både avhengige konkurrerende risikoer og semi-konkurrerende risikoer ved hjelp av tidspunkter for første krysning av visse nivåer i en gammaprosess. I begge tilfeller betrakter vi en terminerende hendelse, som for eksempel at en person dør eller at en komponent svikter, samt en ikke-terminerende hendelse, slik som tilbakefall av en sykdom eller preventivt vedlikehold av en komponent. Vi lar tiden til den terminerende hendelsen være lik tidspunktet for første krysning av et bestemt nivå c i en gammaprosess. Tiden til den ikke-terminerende hendelsen representeres av tidspunktet for første krysning av et annet, stokastisk nivå S. Vi har antatt at S er uavhengig av gammaprosessen slik at vi har random signs censoring.

For konkurrerende risikoer har en lignende modell basert på Wienerprosesser blitt studert tidligere. For semi-konkurrerende risikoer er dette en ny tilnærming, ettersom data innen semi-konkurrerende risikoer stort sett har blitt modellert gjennom copula-modeller frem til nå. Vi utfører simuleringsstudier som viser at parameterne i gammaprosess-modellen kan estimeres på tilfredsstillende vis, for både for data innen konkurrerende og semi-konkurrerende risikoer. Modellen blir også anvendt på reelle datasett og ser ut til å kunne tilpasses bra til dataene, i hvert fall for visse utvalgte fordelinger av det tilfeldige nivået S. Det er spesielt interessant å merke seg at våre resultater for semi-konkurrerende risikoer er konsistente med tidligere publiserte resultater.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In survival analysis the goal is typically to model the time to failure, where failure can be defined as any suitable event. An important aspect of survival analysis is when there may be more than one event. Often, the occurrence of the first event precludes the occurrence of any other event. This is called a competing risks situation. For instance, one may wish to model the time to failure when this can happen from more than one cause. To our knowledge, one of the first people to apply competing risks theory was Daniel Bernoulli. In 1760 he tried to separate the risk of dying due to smallpox from the risk of dying due to other causes [10]. Today, in addition to being of use in many medical studies, the competing risks approach has many applications in for instance reliability and maintenance analysis, actuarial science and demography studies [23].

Many types of failure occur gradually through a degradation process [1]. To model the time to such failures, one may use first passage times in stochastic processes. These types of degradation models are widely used in reliability and maintenance studies, and also in medicine and biomedical research. The most prevalent stochastic process is perhaps the Wiener process. It has the pleasant property that the first passage time to a specific level follows an inverse Gaussian distribution. However, in many situations a gamma process may be more suitable [38]. This is because the increments of the gamma process are always non-negative.

In this thesis, we will model dependent competing risks through first passage times in a gamma process. More specifically, we will consider the setting where there are two competing events: preventive maintenance (PM) and failure. It is natural to assume that these two events are not independent. In that case, the marginal distributions of the time to PM and the time to failure are not identifiable without making any further assumptions [37]. The marginal distribution of the times to failure could for instance be of interest as a basis for maintenance optimization. To deal with this problem, it is assumed that the probability of experiencing a failure or a PM is independent of the process, i.e. the age of the item. This means that

we have random signs censoring, and that the marginal distribution of the time to failure is identifiable [7].

Everything that is described up to and including the previous paragraph was also studied in my project thesis [35]. There, we let the time to failure be defined as the time until the degradation process had reached a certain level, c. The time to a preventive maintenance action was similarly defined as the time until another level, $s < c$, was reached. The idea is that a signal indicating that something is wrong with the item is emitted once it reaches level s, and there is only a certain probability that this signal is detected. If it is, the process stops at s and a failure is prevented. If not, the degradation process continues up to the critical threshold c where the item fails. We will in the following refer to this as the basic model.

In this thesis, we want to extend the basic model by letting the level of potential preventive maintenance (s) be a random variable (S). In this respect, the item will fail if $S > c$ or have a PM if $S < c$. This should make the model more flexible. Skogsrud and Lindqvist studied an equivalent model, only with a Wiener process instead of a gamma process [25][36]. They considered both a model with constant s and models with random S. When applied to real data, a model with normally distributed S seemed to provide the best fit. In my project thesis we used the same datasets that they had used, and found out that the gamma process model with fixed s resulted in a much better fit to the data than any of the Wiener process models had done. We will now find out what the effect of randomizing S in the gamma process model will be.

A gamma process model like the one we have described should also be of interest with regards to semi-competing risks situations. In semi-competing risks one is often concerned with two types of events, terminal and non-terminal. The difference between ordinary competing risks and semi-competing risks is that in semi competing risks one always gets to observe the time to the terminal event. Semi-competing risks data often arise in medical research, where the non-terminal event may for instance be disease recurrence, and the terminal event is typically death. The term semi-competing risks was first introduced by Fine, Jiang and Chappell in 2001[12]. In most previous studies, semi-competing risks have been modelled by copula models. It should however not be a problem to model the dependency between the time to the terminal and the time to the non-terminal event through a model such as the gamma process model, if Cooke's random signs censoring property [7] holds.

In a way, this thesis therefore consists of two parts, one on competing risks and one on semi-competing risks, but we will apply the same gamma process model to both. An outline of the rest of the thesis is as follows: in chapter 2 some theory on basic probability and reliability is introduced. This is followed by some theory on competing risks in chapter 3 and on semi-competing risks in chapter 4. In chapter 5 the gamma process is presented. The chapter on competing risks and the chapter on the gamma process, as well as the basic theory in chapter 2, is taken from my project thesis [35] and repeated here for the sake of completeness. Once the theoretical foundation has been laid, we move on to chapter 6 where we describe the gamma process models that we will use in this thesis. In chapter 7 we explain how to simulate data from these models and present some simulation studies, both for competing risks data and semi-competing risks data. There, we also evaluate how well the parameters of the models can be estimated. The models will then be applied to some real datasets. This is done for competing risks in chapter 8 and for semi-competing risks in chapter 9. Finally, in chapter 10 we discuss our results and present some suggestions for further work. All data analysis is done with R (a programming language and software environment for statistical computing [32]).

Chapter 2

Basic concepts

Before we describe our gamma process model for competing and semi-competing risks, we will in this chapter present some basic theory from survival analysis and probability theory. These are concepts that will be important throughout the thesis. Most of this chapter is taken from the project thesis [35]. Supplementary theory that is relevant to the thesis, but generally assumed to be known to the reader can be found in appendix A.

2.1 Time to failure

In general, the data we deal with in survival analysis represent the time before we observe a specific event or endpoint. For example we can study the number of hours a machine is functioning before it breaks down or the number of days a patient survives a terminal disease. This event time is often called the *survival time* or the *time to failure*. The time to failure is usually denoted by the random variable T. T will in most cases represent calendar time, but it may also denote other measures such as the number of kilometres driven by a car, the number of times a machine is started or the length of a crack.

2.2 Probability functions

Even though T may be a discrete variable, we will throughout this thesis assume that T is continuously distributed with *cumulative distribution function* $F(t) = P(T \leq t)$. This is the probability that the event has occurred within the time interval $(0, t]$. The *probability density function* of T is further given by $f(t) = \frac{d}{dt}F(t)$. The *survival function* (or *reliability function*) is defined by $S(t) = P(T > t)$. This is the probability that the event does *not* happen within the time interval $(0, t]$. The relations between the survival function, the distribution function and the probability density function are as follows:

$$S(t) = 1 - F(t), \qquad f(t) = -\frac{d}{dt}S(t)$$

Another useful function is the *hazard function* or the *failure rate function* $\lambda(t)$. The hazard function is defined by

$$\lambda(t) = \lim_{\Delta t \to 0} \frac{P(t < T < t + \Delta t | T > t)}{\Delta t} = \frac{f(t)}{S(t)}$$

$\lambda(t)$ describes the probability of failing in a small interval, given that the item has survived up to the starting point of the interval. An intuitive interpretation of the failure rate is that it is the amount of risk an item is subject to at time t. Furthermore, the *cumulative hazard function* is given by

$$\Lambda(t) = \int_0^t \lambda(u) du$$

A relation that is quite useful and that we will use later is the following

$$S(t) = e^{-\int_0^t \lambda(u) du} = e^{-\Lambda(t)} \tag{2.1}$$

2.3 Censoring

By the term censoring we mean a condition where the value of a measurement or observation is only partially known [33]. In survival analysis it is often the case that one or more of the failure times will not be registered, and we then have what we call a *censored dataset* (in contrast to a *complete dataset*). Censoring can either be planned or out of our control. There are several types of censoring, but for the purpose of this thesis we will only consider what is called *right censoring*, and more specifically *type I censoring*. These terms are defined as follows:

- **Right censoring**: when the experiment is terminated before the item has failed
- **Type I censoring**: a type of right censoring that occurs when all items are activated at the same time, $t = 0$, and the experiment is terminated at time t_0. Here, only the lifetimes of the items that failed before t_0 will be known

In censored datasets one often includes an indicator variable δ_i in addition to the registered failure times. δ_i is defined by

$$\delta_i = \begin{cases} 1 & \text{if } t_i \text{ is a failure time} \\ 0 & \text{if } t_i \text{ is a censoring time} \end{cases}$$

2.4 Non-parametric estimators of $S(t)$ and $\Lambda(t)$

Non-parametric estimators do not rely on any assumptions regarding the distribution of T, other than that it is continuous. The estimators presented here can be used for both censored and uncensored data. We assume that two or more items may not fail at the same time, nor can an item be censored at the same time as another item fails.

2.4.1 The Kaplan-Meier estimator

To estimate the survival function, it is common in survival analysis to use the Kaplan-Meier estimator. As done in [33], denote the observed times, either to failure or to censoring, by t_i, $i = 1, ..., n$. Let $t_{(1)} < t_{(2)} < \cdots < t_{(n)}$ be the same times sorted in ascending order. Define n_i to be the number of items at risk (i.e. functioning and still under observation) immediately before time $t_{(i)}$ and d_i to be the number of failures at $t_{(i)}$. The Kaplan-Meier estimator is then defined by

$$\hat{S}(t) = \prod_{i;t_{(i)} \leq t} \frac{n_i - d_i}{n_i} \tag{2.2}$$

2.4.2 The Nelson-Aalen estimator

To estimate the cumulative hazard function $\Lambda(t)$, one can use the Nelson-Aalen estimator. This estimator can be deduced by the Kaplan-Meier estimator in (2.2) and the relation in (2.1). The Nelson-Aalen estimator is defined by

$$\hat{\Lambda}(t) = \sum_{i;t_{(i)} \leq t} \frac{1}{n_i} \tag{2.3}$$

As before, n_i is the number of items at risk just before time t_i.

2.5 Estimation of variance and confidence intervals

We will in this thesis use maximum likelihood estimation to find parameter estimates in our models. The maximum likelihood procedure is described in appendix A.2. In the following we will use θ to denote a vector of parameters, and $\hat{\theta}$ to denote a vector of parameter estimates. No matter how efficient our parameter estimators are, they are not exact. It is therefore of great interest to estimate their variances and/or their corresponding confidence intervals. In this section we will describe two methods of finding an estimate of the standard deviation of a parameter estimate, $\text{SD}(\hat{\theta})$. The first is by the Hessian matrix, and the second is by bootstrapping. We

will further present the standard confidence interval and the standard confidence interval for positive parameters and how to construct them. In addition, we will see how to find bootstrap confidence intervals. The theory in this section is in large part from [5].

2.5.1 Estimating SD($\hat{\theta}$) by the Hessian matrix

We begin by defining the Hessian matrix, sometimes just called the Hessian. The Hessian is a square matrix of second-order partial derivatives of a function. When the function in question is a log likelihood function, $l(\theta)$, the Hessian matrix will be:

$$H(l) = \begin{bmatrix} \frac{\partial^2 l(\theta)}{\partial \theta_1^2} & \frac{\partial^2 l(\theta)}{\partial \theta_1 \partial \theta_2} & \cdots & \frac{\partial^2 l(\theta)}{\partial \theta_1 \partial \theta_k} \\ \frac{\partial^2 l(\theta)}{\partial \theta_2 \partial \theta_1} & \frac{\partial^2 l(\theta)}{\partial \theta_2^2} & \cdots & \frac{\partial^2 l(\theta)}{\partial \theta_2 \partial \theta_k} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 l(\theta)}{\partial \theta_k \partial \theta_1} & \frac{\partial^2 l(\theta)}{\partial \theta_k \partial \theta_2} & \cdots & \frac{\partial^2 l(\theta)}{\partial \theta_k^2} \end{bmatrix}$$

One can note that this is the negative of the observed Fisher information matrix. By taking the inverse of this Hessian matrix, the variance of each parameter $\theta_i, i = 1, ..., k$ can be estimated by the diagonal elements. Then, the square root of these estimates will be estimators for SD($\hat{\theta}_i$), i.e.

$$\widehat{\mathrm{SD}(\hat{\theta}_i)} = \sqrt{|\mathrm{H}(l)|_{ii}^{-1}} \tag{2.4}$$

In the simulation study in chapter 7 and the data analysis in chapters 8 and 9, we will use this method of estimating SD($\hat{\theta}_i$). This is easily done in R, as the `optim`() function calculates the Hessian matrix.

2.5.2 Estimating SD($\hat{\theta}$) by bootstrapping

Another way to evaluate the accuracy of an estimator is by bootstrapping. The following theory is selected from chapter 9 in [14]. Assume that you have independent observations $\mathbf{x} = x_1, ..., x_n$ from a population with cumulative distribution function F. Assume further that you have made a parameter estimate $\hat{\theta} = t(\mathbf{x})$ (for example an estimate of the mean of the distribution). Now, the idea of bootstrapping is to make B bootstrap samples $\mathbf{x}^{*b}, b = 1, ..., B$, all of size n. This can be done in several ways. Here, we will focus on the non-parametric bootstrap. The advantage of the non-parametric technique, is that we do not have to make any assumptions of the distribution of $\mathbf{x}$ (in the parametric bootstrap we assume that the data comes from

the distribution F, or at least that this provides a good representation of reality. However, if this turns out to not be the case, the results of parametric bootstrapping may be misleading).

Non-parametric bootstrapping
Let $\hat{F}$ denote the empirical distribution of F. $\hat{F}(t)$ of the observed data is defined as $\frac{1}{n}\sum_{i=1}^{n} I_{\{x_i \leq t\}}$. The way we generate bootstrap samples non-parametrically is to draw from this empirical distribution, i.e. draw with replacement from the original data $\mathbf{x} = x_1, ..., x_n$. From each bootstrap sample $\mathbf{x}^{*b}$ one can then find an estimate for the parameter θ, $\hat{\theta}^*(b)$.

Once the bootstrap estimates $\hat{\theta}^*(b), b = 1, ...B$ have been found, one can calculate the bias and the standard deviation of the estimator $\hat{\theta}$. The bias can be estimated as the mean value of $\hat{\theta}^* - \hat{\theta}$:

$$\widehat{\text{bias}} = \sum_{b=1}^{B} \frac{(\hat{\theta}^*(b) - \hat{\theta})}{B} = \bar{\theta}^* - \hat{\theta}$$

The bootstrap standard deviation is estimated by

$$\widehat{\text{SD}}_{\text{B}} = \sqrt{\frac{1}{B-1}\sum_{b=1}^{B}(\hat{\theta}^*(b) - \bar{\theta}^*)^2}$$

where $\bar{\theta}^* = \frac{1}{B}\sum_{b=1}^{B}\hat{\theta}^*(b)$

2.5.3 Confidence intervals from the Hessian matrix

General theory (for example [5] p. 472) states that if a maximum likelihood estimator $\hat{\theta}$ is calculated from a large sample, then it is approximately normally distributed with mean θ, i.e.

$$\frac{\hat{\theta} - \theta}{\widehat{\text{SD}(\hat{\theta})}} \to N(0, 1)$$

Here, $N(0, 1)$ is the usual way to denote the normal distribution with mean 0 and variance 1. It then follows that for a significance level α

$$P\left(-z_{\alpha/2} < \frac{\hat{\theta} - \theta}{\widehat{\text{SD}(\hat{\theta})}} < z_{\alpha/2}\right) \approx 1 - \alpha$$

where $\widehat{\text{SD}(\hat{\theta})}$ is the estimate we found in (2.4) and $z_{\alpha/2}$ is the $\alpha/2$ quantile in the standard normal distribution. This gives what is called the $(1 - \alpha)100\%$ *standard*

confidence interval. We usually write

$$\hat{\theta} \pm z_{\alpha/2}\widehat{\mathrm{SD}(\hat{\theta})}$$

If we want to get a 95%= (1 - 0.05)100% standard confidence interval we use the quantile $z_{0.025} \approx 1.96$.

In the gamma process models that we shall explore later in this thesis, all of the parameters should be positive. It is then common to use what we will call the *standard confidence interval for positive parameters*. This interval is found by the re-parameterization $g(\theta) = \ln \theta$, which will result in only positive values in the confidence intervals if θ itself also is positive. By the delta method (which is further described in appendix A.4) it follows that

$$\frac{g(\hat{\theta}) - g(\theta)}{[g'(\theta)]^2 \operatorname{Var}(\hat{\theta})} \to N(0,1) \tag{2.5}$$

As mentioned above, we have $g(\hat{\theta}) = \ln \hat{\theta}$. We then get:

$$[g'(\theta)]^2 \operatorname{Var}(\hat{\theta}) = \frac{1}{\theta^2}\operatorname{Var}(\hat{\theta})$$

The standard deviation is then the square root of this, that is

$$\mathrm{SD}(\ln \hat{\theta}) = \frac{1}{\theta}\mathrm{SD}(\hat{\theta})$$

The relation in (2.5) still holds if we insert an estimate of $\mathrm{SD}(\ln \hat{\theta})$, namely $\frac{1}{\hat{\theta}}\widehat{\mathrm{SD}(\hat{\theta})}$, which in this case is the estimate we found in (2.4). Thus, for a large enough sample size the confidence interval for $g(\theta)$ (and thereby also for θ), can be found from:

$$\frac{\ln \hat{\theta} - \ln \theta}{\frac{1}{\hat{\theta}}\widehat{\mathrm{SD}(\hat{\theta})}} \to N(0,1)$$

If we for example let $\alpha = 0.05$, the corresponding 95% confidence interval for $\ln \theta$ is given by

$$\ln \hat{\theta} \pm 1.96\frac{1}{\hat{\theta}}\widehat{\mathrm{SD}(\hat{\theta})}$$

By exponentiating this, we get the following 95% standard confidence interval for the positive parameter θ

$$\hat{\theta} \exp\left\{\pm 1.96 \frac{1}{\hat{\theta}} \widehat{\mathrm{SD}(\hat{\theta})}\right\} \tag{2.6}$$

2.5.4 Bootstrap confidence intervals

Confidence intervals can also be estimated by bootstrapping techniques. The perhaps simplest two-sided (1-α)100% bootstrap confidence interval is the *percentile interval* $[q^*_{\alpha/2}, q^*_{1-\alpha/2}]$ where q^*_α is the α-quantile in the distribution of $\hat{\theta}^*$. This estimate is often quite good, but is prone to bias and may provide too low coverage [14]. There exist a wide range of other bootstrap confidence intervals that correct for bias and/or skewness in the bootstrap distribution. Here, we will only compute one of these, using the *accelerated bias-corrected percentile method*, BC_a. The BC_a approach is often a lot better than the simple percentile method since the percentile levels in the confidence intervals are corrected for bias and skewness. Instead of using the $\frac{\alpha}{2}$ and the $1 - \frac{\alpha}{2}$ quantiles in the distribution of $\hat{\theta}^*$, we now use the β_1 and β_2 quantiles, where the βs are functions of a bias correction b, and an acceleration a in the following manner [14]:

$$\beta_1 = \Phi\left(b + \frac{b + z_{\alpha/2}}{1 - a(b + z_{\alpha/2})}\right), \quad \beta_2 = \Phi\left(b + \frac{b + z_{1-\alpha/2}}{1 - a(b + z_{1-\alpha/2})}\right)$$

Here, Φ is the standard normal cumulative distribution function. According to [14], the simplest, non-parametric choices for b and a are

$$b = \Phi^{-1}\left(\hat{F}^*(\hat{\theta})\right)$$

$$a = \frac{\sum_{i=1}^n (\hat{\theta}_{(\cdot)} - \hat{\theta}_{(-i)})^3}{6\left[\sum_{i=1}^n (\hat{\theta}_{(\cdot)} - \hat{\theta}_{(-i)})^2\right]^{3/2}}$$

with $\hat{\theta}_{(-i)}$ being the parameter estimate computed without observation i and $\hat{\theta}_{(\cdot)} = \frac{1}{n}\sum_{i=1}^n \hat{\theta}_{(-i)}$. Later in this thesis we will compute the BC_a intervals by the script in appendix D.2.5.

Chapter 3

Competing risks

In standard survival analysis one considers the time until failure of some item. Thus, a failure is the only possible event. In many cases however, we would like to include the possibility of different types of events. For example in medical studies one may wish to model the time until recurrence of a disease or until death, whichever comes first. One way to model this is by competing risks. We will in the following use the same notation as in [23]. The chapter is taken from the project thesis [35].

3.1 Definition and model specification

Consider an item which can fail due to one out of k possible causes. For each item we observe both the time to failure, T, and the failure cause, $C \in \{1, 2, ..., k\}$. A useful approach in reliability applications of competing risks is the latent failure time approach. One can imagine that each of the k failure types has an associated potential failure time $T_j, j = 1, ..., k$. These k times are hypothetical failure times that would have been realized if the other risks were not present, and they are therefore called latent failure times. When all of the risks (or possible failure causes) are present, the observed time to failure for the system is the smallest of the latent failure times. That is, for each item we observe the pair (T, C), where $T = \min_j T_j$ and $C = \arg\min_j T_j$.

3.1.1 Special case with two competing risks

In this thesis we will only consider a particular situation with two competing risks, that is $k = 2$. Here, we let T_1 denote the time until critical failure of a component and T_2 denote the time until preventive maintenance (PM), see figure 3.1. By preventive maintenance we mean that the component is removed and maintained prior to failure. In the following we will denote T_1 by X and T_2 by Z, as it was done in [7]. The observed data will in this case be $T = \min(X, Z) = Y$ and C, where

$$C = \begin{cases} 1 & \text{if there is a failure, i.e. } X < Z \\ 0 & \text{if there is a PM, i.e. } X > Z \end{cases}$$

In this model, the probability of observing Z is defined as $q = P(Z < X)$, which means that $C = 1$ with a probability $1 - q$ and $C = 0$ with a probability q.

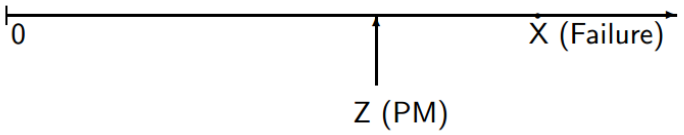

FIGURE 3.1: The lifetime (X) and the potential time to preventive maintenance (Z). Figure copied from [24]

3.2 Probability functions

In the competing risks setting one needs to define functions for the probability distributions that take the cause of failure into account.

3.2.1 Sub-functions

From observations on the form (T, C) one can in general only find the joint distribution of T and C. These joint distributions are described by what is called sub-functions, as their density functions do not necessarily integrate to one. For instance, the cumulative distribution function of (T, C) is called the *sub-distribution function*, and for the jth cause this is expressed by:

$$F_j^*(t) = P(T \leq t, C = j)$$

The *sub-density function* (when it exists) is found by differentiating the sub-distribution function:

$$f_j^*(t) = F_j^{*\prime}(t)$$

In a similar manner, the *sub-survival function* for the jth cause is given by

$$S_j^*(t) = P(T > t, C = j)$$

One should note that the probability of failing due to cause j is given by $P(C = j) = S_j^*(0) = F_j^*(\infty)$.

In the setting with only two competing risks, the sub-survival function for X is written as $S_X^*(t) = P(Y > t, C = 1) = P(\min(X, Z) > t, X < Z)$. Likewise, for Z we get: $S_Z^*(t) = P(Y > t, C = 0) = P(\min(X, Z) > t, Z < X)$.

The joint distribution of (T, C) can also be described by the *sub-hazard function* which is defined as follows:

$$\lambda_j^*(t) = \lim_{\Delta t \to 0} \frac{P(T \leq t + \Delta t, C = j | T > t)}{\Delta t} = \frac{f_j^*(t)}{S(t)}$$

This is sometimes called the cause specific hazard rate. The *cumulative sub-hazard function* is defined as

$$\Lambda_j^*(t) = \int_0^t \lambda_j^*(u) du$$

3.2.2 Conditional sub-functions

The *conditional sub-functions* are sub-functions conditioned on the event that the failure cause in question lead to the failure. More precisely, the *conditional sub-survival function* is defined as

$$\tilde{S}_j(t) = P(T > t | C = j) = \frac{P(T > t, C = j)}{P(C = j)} = \frac{S_j^*(t)}{S_j^*(0)}$$

The *conditional sub-distribution function* is in a similar way given by

$$\tilde{F}_j(t) = P(T \leq t | C = j) = \frac{P(T \leq t, C = j)}{P(C = j)} = \frac{F_j^*(t)}{F_j^*(\infty)} \tag{3.1}$$

When there are only two competing risks, the conditional sub-survival functions are

$$\tilde{S}_X(t) = \frac{S_X^*(t)}{S_X^*(0)} \quad \text{and} \quad \tilde{S}_Z(t) = \frac{S_Z^*(t)}{S_Z^*(0)}$$

The concept of conditional sub-survival functions will be important when discussing the property random signs censoring in section 3.5.

3.2.3 Sub-survival pair

We here define the term *sub-survival pair* for the special case with two competing risks, as done in [7]. This term will be used later in relation to random signs censoring.

Definition 3.1. Real functions $S_X^*(t)$ and $S_Z^*(t)$ form a sub-survival pair if

1. $S_X^*(t)$ and $S_Z^*(t)$ are non-negative, non-increasing real functions with $S_X^*(0) \leq 1$ and $S_Z^*(0) \leq 1$
2. $\lim_{t\to\infty} S_X^*(t) = \lim_{t\to\infty} S_Z^*(t) = 0$
3. $S_X^*(0) + S_Z^*(0) = 1$

3.3 Non-parametric estimation

For a given dataset, we want to estimate the sub-distribution functions and the conditional sub-distribution functions. If we have these estimates, we can easily find estimates for the sub-survival function and the conditional sub-survival function as $S_j^*(t) = 1 - F_j^*(t)$ and $\tilde{S}_j(t) = 1 - \tilde{F}_j(t)$ respectively. The theory in this section is in large part from [22].

3.3.1 Estimating the sub-distribution function

An estimate of the sub-distribution function is provided by

$$\hat{F}_j^*(t) = \int_0^t \hat{S}(u) d\hat{\Lambda}_j^*(u), \quad j = 1, ..., k. \tag{3.2}$$

Here, $\hat{S}(t)$ is an estimate of the marginal survival function $S(t)$ and $\hat{\Lambda}_j^*(t)$ is an estimate of the cumulative sub-hazard function. The marginal survival function $S(t)$ is estimated by the Kaplan-Meier estimator from section 2.4.1. By simply ignoring the observed failure causes we get the following estimate:

$$\hat{S}(t) = \prod_{i; t_{(i)} \leq t} \frac{n_i' - d_i'}{n_i'}$$

where $t_{(1)} < ... < t_{(N)}$ are the sorted failure or censoring times, d_i' is the number of failures at time $t_{(i)}$ and n_i' is the number of individuals at risk at time $t_{(i)}$. A non-parametric estimate of $\Lambda_j^*(t)$ is found by using the Nelson-Aalen estimator from section 2.4.2:

$$\hat{\Lambda}_j^*(t) = \sum_{i; t_i \leq t} \frac{\delta_{ij}}{n_i}, \quad j = 1, ..., k$$

where n_i is the number of individuals alive and not censored just prior to time t_i, and δ_{ij} is the indicator function $I(C_i = j)$. Inserted into (3.2), this gives the following estimate for the sub-distribution function

$$\hat{F}_j^*(t) = \sum_{i; t_i \leq t} \hat{S}(t_i) \frac{\delta_{ij}}{n_i}, \quad j = 1, ..., k$$

This is sometimes called the Aalen-Johansen estimator.

3.3.2 Estimating the conditional sub-distribution functions

From equation (3.1) one might think that the conditional sub-distribution function should be estimated by $\hat{\tilde{F}}_j(t) = \frac{\hat{F}_j^*(t)}{\hat{F}_j^*(\infty)}$. However, the estimates of the sub-distribution functions may not fulfil the property $\sum_{j=1}^k \hat{F}_j^*(\infty) = 1$, so the following re-normalization of the estimates is done:

$$\hat{F}_j^*(\infty)' = \frac{\hat{F}_j^*(\infty)}{\sum_{j=1}^k \hat{F}_j^*(\infty)}$$

This ensures that $\sum_{j=1}^k \hat{F}_j^*(\infty)' = 1$. Estimates of the conditional sub-distribution functions are then given by:

$$\hat{\tilde{F}}_j(t) = \frac{\hat{F}_j^*(t)}{\hat{F}_j^*(\infty)'}$$

In the special case of only two competing risks, the re-normalization of the sub-distribution function can be used as a non-parametric estimate of the probability $q = P(Z < X)$

$$\hat{q} = \frac{\hat{F}_Z^*(\infty)}{\hat{F}_X^*(\infty) + \hat{F}_Z^*(\infty)} \tag{3.3}$$

The estimates of the conditional sub-distribution functions $\tilde{F}_X(t)$ and $\tilde{F}_Z(t)$ then become

$$\hat{\tilde{F}}_X(t) = \frac{\hat{F}_X^*(t)}{1-\hat{q}} \quad \text{and} \quad \hat{\tilde{F}}_Z(t) = \frac{\hat{F}_Z^*(t)}{\hat{q}} \tag{3.4}$$

In R we will compute these estimates by the function `condSurv()`, which is included in appendix D.2.1.

3.4 The identifiability problem

As briefly mentioned earlier, a problem with the competing risks model is that the distribution of the observable pair (T, C) does not in general determine the distribution of the latent failure times $T_1, ..., T_k$. We usually say that the joint and marginal distributions of $T_1, ..., T_k$ are non-identifiable from observations of (T, C). This issue was first noted by Cox in 1959 [9] and has later been studied in great detail, especially by Tsiatis. To deal with the identifiability problem one needs to impose some additional restrictions on the model. The simplest solution would be to assume that the latent failure times are independent of each other, i.e. that the competing risks act independently.

In many cases, to assume independence is not reasonable. Often, the competing events share some common factors such as the surrounding environment, the manufacturer or what kind of maintenance is performed. Then, the rates of occurrence for the different events are likely to affect each other. This is also the case in this thesis, with the competing risks being preventive maintenance and critical failure. If these events were independent, the rate of occurrence of critical failures would be unaffected if we stopped doing any preventive maintenance, which does not make sense. Another argument is that one would assume that the PM-team has some knowledge about the state of the item during operation, and that this will affect what moment they choose to do the PM (as they want to avoid failure). In these situations one can use random signs censoring to deal with the identifiability problem. Unfortunately, the additional assumptions we make are non-testable when we only have observations (T, C). More on this subject can be read in [37].

3.5 Random signs censoring

Random signs censoring (RSC) was first introduced by Cooke in 1993 as age-dependent censoring [7]. The concept of RSC is that whether a component is censored or not is independent of the age of the component. However, given that the component is censored, the censoring time may depend on its age. In our setting with only two competing risks, RSC can be defined as follows [26]:

Definition 3.2. Let (X, Z) be a pair of life variables. Then Z is called a *random signs censoring* of X if the event $\{Z < X\}$ is stochastically independent of X.

This means that PM is either done or not done on an item, independent of the time X where the item will/would have failed. We can imagine that before the item fails, it will emit a signal indicating that a failure is emerging. This is in many cases a reasonable assumption to make. For instance, if the item in question is a machine, typical signals may be excessive noise and/or vibration. If we were to consider a human being in a medical study instead, symptoms of disease could serve as the signal. We must further assume that the emitted signal will be discovered with a probability that does not depend on the age of the item we are considering.

Random signs censoring implies that the marginal distribution of X is identifiable [23]. The distribution of Z however, is in general not identifiable under these assumptions, only the conditional distribution of Z given that $Z < X$. As noted in [36], the definition of random signs censoring leads to the following conditional distribution for X

$$\tilde{S}_X(t) = P(X > t | X < Z) = \frac{P(X > t, X < Z)}{P(X < Z)} = P(X > t) = S_X(t) \tag{3.5}$$

Thus, the marginal distribution of X actually is the same as the distribution of the observed failure times. To later check whether our data is suited to fit to a random signs distribution for (X, Z), we will use a theorem described in [7]:

Theorem 3.3. *Let K_1, K_2 be a sub-distribution pair. Then the following are equivalent:*

1. *There exists a pair (X, Z) of life variables such that Z is a random signs censoring of X, and such that:*

$$F_X^*(t) = K_1(t), F_Z^*(t) = K_2(t) \quad \textit{for all} \quad t \geq 0$$

2. $$\frac{K_1(t)}{K_1(\infty)} < \frac{K_2(t)}{K_2(\infty)} \quad \textit{for all} \quad t > 0$$

The theorem says that a random signs distribution for (X, Z) exists if and only if the conditional distribution function of X, $\tilde{F}_X(t)$, is below that of Z, $\tilde{F}_Z(t)$, for all t [25]. Equivalently, a random signs distribution for (X, Z) exists if and only if $\tilde{S}_Z(t) < \tilde{S}_X(t)$ for all t.

Chapter 4

Semi-competing risks

A version of the competing risks problem that often arises in medical research and clinical trials is semi-competing risks. In this chapter we will introduce the concept of semi-competing risks and some modelling approaches.

4.1 Introduction

Semi-competing risks is a variation of ordinary competing risks. The term semi-competing risks was first introduced by Fine, Jiang and Chappell in 2001 [12]. In semi-competing risks, one often considers two types of events: non-terminal and terminal. The difference from ordinary competing risks is that the focus is not restricted to the first event. A non-terminal event may be censored by a terminal event but *not* vice versa. That is, the non-terminal event does not prevent the observation of the terminal event (as it would have done in ordinary competing risks problems). Thereby, more information regarding event times is obtained with semi-competing risks than with ordinary competing risks.

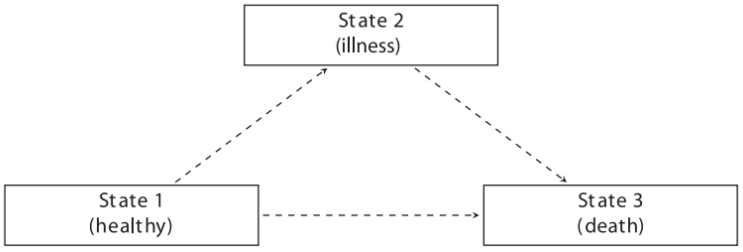

FIGURE 4.1: The illness-death model

Semi-competing risks has especially many applications in medicine. In medical research and clinical trials one can often observe several distinct events related to disease progression or a patient's condition. This results in a set of numerous event times. In this setting, a non-terminal event may for instance be disease recurrence, while the terminal event typically is death or drop-out from the study. The semi-competing risks problem is equivalent to the classical illness-death model, which was introduced by Fix and Neyman in 1951 [13], see figure 4.1. The illness-death model is further considered to be a special case of the multi-state models [39].

Some examples of semi-competing risks models applied to medicine can be found in cancer research. Typically, relapse is defined to be the non-terminal event while death is the terminal event. This is for instance the case in the article by Fine, Jiang and Chappell from 2001 [12] where a set of bone marrow transplant data is analysed. Semi-competing risks problems also naturally arise in studies of ageing (gerontology). The event death will often censor some other event that is under study, like dementia (Alzheimer's disease) or disability [39].

4.2 Notation

We will in this thesis only deal with two events in the semi-competing risks case, one terminal and one non-terminal. Let T_1 and T_2 be the times of the terminal event and the non-terminal event respectively. As in the competing risks case, we choose to denote T_1 by X and T_2 by Z, even though we are no longer considering preventive maintenance and critical failure as events. We may also wish to include a time of censoring, τ, for cases where there is loss to follow up.

Let $Y_1 = \min\{Z, X, \tau\}, Y_2 = \min\{X, \tau\}, \delta_1 = I\{Z \leq Y_2\}$ and $\delta_2 = I\{X \leq \tau\}$, where I is the indicator function. Now, as explained in the previous section, X is always observed if we let the study go on long enough and there are no drop-outs. Z however, may be prevented from happening by the occurrence of the terminal event. Ideally, to make inferences on Z we would like to observe Z_i for each individual $i = 1, ..., n$. In reality however, we only get to observe n i.i.d. copies of $\{Y_{1i}, Y_{2i}, \delta_{1i}, \delta_{2i}\}$. There are six possible orders that Z, X and τ can occur in. Table 4.1 lists these potential permutations and what kind of semi-competing risks data they will result in.

TABLE 4.1: Possibilities of semi-competing risks data depending on the order of the terminal and non-terminal events and when the observations are censored

Order	Resulting data	Case
Z, X, τ	(Z, X, 1, 1)	1
X, Z , τ	(X, X, 0, 1)	2
X, τ, Z	(X, X, 0, 1)	
Z, τ, X	(Z, τ, 1, 0)	3
τ, Z, X	(τ, τ, 0, 0)	4
τ, X, Z	(τ, τ, 0, 0)	

The permutation listed in the first row results in observations of both Z and X. We will denote this as case 1. Furthermore, we can see that the order in the second and

third rows result in the same observations of only the time to the terminal event. We refer to this as case 2. Similarly, the order in row number four will represent case 3, and results in observations of the time to the non-terminal event and a censoring time. The permutations of the bottom two rows will give the same set of observations of only the censoring time τ. In the following, we will refer to this as case 4. Thus, we have in total four possible scenarios.

In figure 4.2(b), an illustration of semi-competing risks data is given and compared to simple right-censored data in figure 4.2(a) and ordinary competing risks data in figure 4.2(c). The figures are taken from [18]. A dot represents the observation of both Z and X. A vertical arrow indicates a censoring of X, while a horizontal arrow indicates a censoring of Z. For example, in figure 4.2(b), subject 1 experienced the non-terminal event and later the terminal event, so we observe both Z and X. Subject 2 experienced the non-terminal event, but did not experience the terminal event before the end of the study, so we observe Z and a censoring time τ. Subject 3 experienced the terminal event before it got to experience the non-terminal event, so we only observe X and so on. Note that for the simple right-censored data in 4.2(a), the data points may be located anywhere in the first quadrant. For the semi-competing risks data in 4.2(b), every realization must be above the diagonal line. For the ordinary competing risks data in 4.2(c) we never observe Z and X together, and so every point must be on the diagonal line.

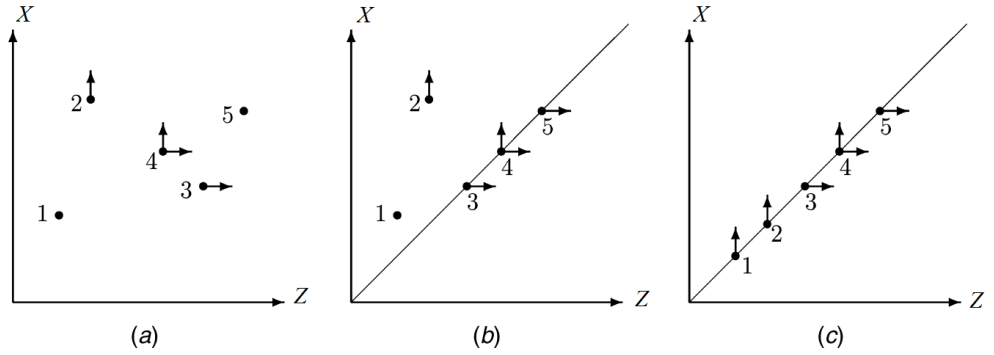

FIGURE 4.2: Illustration of semi-competing risks data (b) compared to simple right censored data (a) and ordinary competing risks data (c). Figure copied from [18]

4.3 Approaches to semi-competing risks

While competing risks theory has been used in a wide range of applications and studied in great detail, semi-competing risks models have not yet become that popular. The reason for this is probably that for a long time there has not been any appropriate way to make inferences with semi-competing risks data. The terminal

and the non-terminal event can seldom be considered to be independent of each other. Thus, we face the same kind of identifiability problem as in competing risks, described in section 3.4. The extra information regarding the times to the terminal events makes the semi-competing risks problem more complex to analyse.

During the past 15 years however, several semi-parametric models have been developed and successfully applied to semi-competing risks data for instance in [12], [30] and [17]. These models are still relatively new, and some believe that they are too sophisticated to catch on amongst a broad range of researchers [39]. To take the dependent censoring into account, it is common to assume that the bivariate distribution of non-terminal and terminal event times is a known copula. This may for example be the gamma frailty copula [6]. The joint distribution (X, Z) is identifiable in the upper wedge where $Z < X$, but the marginal distribution of the non-terminal event Z is not identifiable without having to make additional assumptions.

In general one may say that there are two main approaches to semi-competing risks: analysis by net quantities and analysis by crude quantities [31]. The two approaches have also been denoted as models based on potential (latent) failure times and models based on only observable quantities, respectively [39]. The point of the following section is not to weigh in on the debate on which approach is better to use, only to present the two approaches objectively. We will use both later in the thesis.

4.3.1 Analysis by net quantities

The goal in many medical studies is to estimate patient survival. One then needs to study the distribution of the terminal event, which is non-parametrically identifiable in semi-competing risks. However, there are also many cases where it is of interest to make inference on the marginal distribution of the time to the non-terminal event [12]. This approach is often referred to as analysis based on *net quantities*. The problem with estimating these net quantities is that the terminal event may occur first, and thereby prevent the non-terminal event from happening. The Kaplan-Meier estimate for $P(Z > t)$ is often too optimistic, and does not take into account the dependency between Z and X. One therefore needs to make further assumptions in order to estimate the marginal distribution of the time to the non-terminating event. The estimation of this quantity is somewhat controversial, as it depicts a situation where the terminal event never will prevent the non-terminal event from happening. Some [3] have argued that this has no root in reality. On the other hand, Fine, Jiang and Chappell for instance argue that it is a useful quantity to estimate in many respects. In [12] they state that it "addresses the issue of [...] the behaviour

of morbidity as a process distinct from mortality due to other causes." An example they use is cancer patients who have undergone bone marrow transplantation. In that case, they state that it would be useful for the researchers to estimate the marginal distribution to evaluate the efficacy of the treatment.

4.3.2 Analysis by crude quantities

Another approach has been to analyse semi-competing risks data as if they were competing risks data, thereby ignoring the information after the first event. Typically, one then estimates functions such as the cause-specific hazard (which we called the sub-hazard function in chapter 3) and the cumulative incidence function (which we called the sub-distribution function). This approach is often termed analysis by *crude quantities*. The advantage of these quantities is that they are non-parametrically identifiable. In addition, these functions take the presence of the terminal event into account. The downside is that the marginal distributions of Z or X cannot be targeted unless Z and X are independent. In that respect one may say that the crude quantities reflect the observational process more than the underlying distribution of the time to the event. One also needs to be a little careful since these quantities may be susceptible to survival bias. This is because they are conditional quantities (you condition on survival up to time t).

4.4 Non-parametric estimation of crude quantities

We will in the following use the theory from [31]. As in chapter 3, we let $F_Z^*(t)$ describe the distribution function of the non-terminating event in the presence of the terminating event.

$$F_Z^*(t) = P(Z \leq t, X > Z) \tag{4.1}$$

Let $\lambda_Z^*(t)$ describe the instantaneous rate of the non-terminating event in the presence of the terminating event.

$$\lambda_Z^*(t) = \lim_{h \to 0} \frac{P(t < Z < t + h, X > Z | Z \geq t, X \geq t)}{h} \tag{4.2}$$

Now, we want to estimate these functions non-parametrically. This is essentially done in the same manner as in section 3.3. Here, we will however use the notation introduced for semi-competing risks data earlier in this chapter.

Let (as before) $Y_1 = \min\{Z, X, \tau\}, Y_2 = \min\{X, \tau\}, \delta_1 = I(Z \leq Y_2), \delta_2 = I(X \leq \tau)$. Moreover, let $V_i(t) = I(Y_{1i} \geq t), N_{1,i}(t) = I(Y_{1i} \leq t, \delta_{1i} = 1), N_{2,i}(t) = I(Y_{1i} \leq$

$t, \delta_{1i} = 0, \delta_{2i} = 1$). Let further $\bar{V}(t) = \sum_{i=1}^{n} V_i(t), \bar{N}_1(t) = \sum_{i=1}^{n} N_{1,i}(t)$ and $\bar{N}_2(t) = \sum_{i=1}^{n} N_{2,i}(t)$

The non-parametric estimate for $\Lambda_Z^*(t) = \int_0^t \lambda_Z^*(s)ds$ is given by an expression similar to the Nelson-Aalen estimator

$$\hat{\Lambda}_Z^*(t) = \sum_{i=1}^{n} \frac{I(Y_{1i} \leq t, \delta_{1i} = 1)}{\sum_{l=1}^{n} I(Y_{1i} \leq Y_{1l})} \tag{4.3}$$

while for $F_Z^*(t)$ it is

$$\hat{F}_Z^*(t) = \int_0^t \hat{S}_T(u^-) d\hat{\Lambda}_Z^*(u)du \tag{4.4}$$

where $\hat{S}_T(t)$ is the Kaplan-Meier estimator for $S_T(t)$ based on $\{Y_{1i}, I(Y_{1i} < \tau_i), i = 1, ..., n\}$

$$\hat{S}_T(t) = \prod_{Y_{1i} \leq t} \left[1 - \frac{d\bar{N}_1(t) + d\bar{N}_2(t)}{\bar{V}(Y_{1i})}\right]$$

In R we will use the script `semiQuant` to plot these estimates. `semiQuant` is included in appendix D.2.3.

Chapter 5

The gamma process

In this chapter, we will introduce the gamma process and its properties. For a definition of a stochastic process in general, see appendix A.3. After defining the gamma process, we will move on to explore the distribution of the first passage time to a specific threshold in this process. This chapter is taken from the project thesis [35].

5.1 The gamma process

The gamma process was first suggested as a model for deterioration occurring random in time by Abdel-Hameed in 1975 [38]. In contrast to the Wiener process, the level of the gamma process is always non-decreasing. This makes the gamma process suitable for modelling gradual damage accumulation over time, such as wear, corrosion or fatigue. A thorough review of the applications of the gamma process (and how these can be used to optimize maintenance) is given in [38]. There, the following definition of a gamma process can be found:

Definition 5.1. A continuous time stochastic process $\{X(t), t \geq 0\}$ is a gamma process with shape function $v(t) > 0$ and scale parameter $u > 0$ if

1. $X(0) = 0$ with probability 1
2. $\{X(t), t \geq 0\}$ has independent increments,
3. $X(t) - X(s)$ is gamma distributed with shape parameter $v(t) - v(s)$ and scale parameter u for every $0 < s < t$.

In accordance with definition 5.1, the probability density function of the gamma process $X(t)$ is simply given by

$$f_{X(t)}(x) = Ga(x; v(t), u) \tag{5.1}$$

where $Ga(x; v(t), u)$ is the notation for the gamma distribution described in equation (A.4). The mean and variance of $X(t)$ is thus

$$\mathrm{E}[X(t)] = \frac{v(t)}{u}, \quad \mathrm{Var}[X(t)] = \frac{v(t)}{u^2}$$

Empirical studies have shown that the expected deterioration at time t often follows a power law model [38]. Therefore, $v(t)$ is in many cases chosen to be a function on the form $\alpha \cdot t^{\beta}$, for some constants $\alpha > 0$ and $\beta > 0$. The gamma process is stationary if its expected value is linear, i.e. $\beta = 1$ and non-stationary if $\beta \neq 1$.

5.2 First passage time distribution of the gamma process

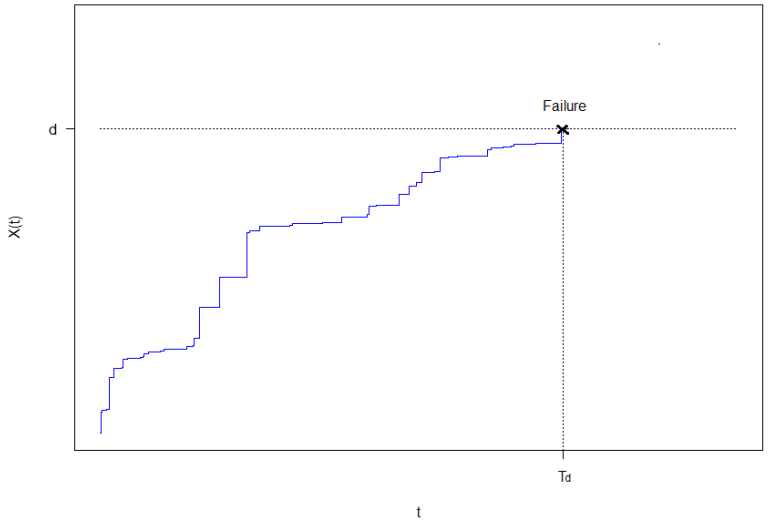

FIGURE 5.1: Illustration of a gamma process $X(t)$ showing the connection between the critical threshold d and the first passage time T_d

In the following we let $X(t)$ be a gamma process with shape function $v(t)$ and scale parameter u describing the level of deterioration of an item. Let d be a constant, deterministic threshold of the level of deterioration for which the item will fail. The first passage time of the process $X(t)$ is defined as the the time until it reaches the threshold d. We denote this time by T_d. In figure 5.1 the relation between the level d and the first passage time T_d is illustrated. The cumulative distribution function for T_d is found to be

$$\begin{aligned} F_{T_d}(t; v(t), u, d) &= P(T_d \leq t) \\ &= P(X(t) > d) \\ &= \int_{x=d}^{\infty} f_{X(t)}(x)dx \\ &= \frac{\Gamma(v(t), d \cdot u)}{\Gamma(v(t))} \end{aligned} \tag{5.2}$$

where $\Gamma(a, x)$ is the upper incomplete gamma function defined by $\Gamma(a, x) = \int_{z=x}^{\infty} z^{a-1}e^{-z}dz$. The corresponding survival function for T_d is consequently

$$S_{T_d}(t; v(t), u, d) = P(T_d > t) = 1 - F_{T_d}(t; v(t), u, d) = 1 - \frac{\Gamma(v(t), d \cdot u)}{\Gamma(v(t))} \tag{5.3}$$

A few examples of how this CDF may look is shown to the left of figure 5.2.

It can be shown (see for example [29] and [28]) that if $v(t)$ is differentiable, the probability density function of T_d is given by

$$\begin{aligned} f_{T_d}(t; v(t), u, d) &= v'(t) \left[\Psi(v(t)) - \log(d \cdot u)\right] \left(1 - \frac{\Gamma(v(t), d \cdot u)}{\Gamma(v(t))}\right) \\ &+ \frac{v'(t)}{v(t)^2\Gamma(v(t))}(d \cdot u)^{v(t)}{}_2F_2(v(t), v(t); v(t)+1, v(t)+1; -d \cdot u) \end{aligned} \tag{5.4}$$

Here, $\Psi(a) = \frac{d}{da}\ln\Gamma(a) = \frac{\Gamma'(a)}{\Gamma(a)}$ is the digamma function, and ${}_2F_2()$ is the generalized hypergeometric function of order (2,2). The generalized hypergeometric function of order (p, q) is defined as

$${}_pF_q(a_1, ..., a_p; b_1, ..., b_q; z) = \sum_{k=0}^{\infty} \frac{(a_1)_k \cdots (a_p)_k}{(b_1)_k \cdots (b_q)_k} \frac{z^k}{k!}$$

where $(x)_n = \frac{\Gamma(x+n)}{\Gamma(x)}$ is the Pochammer symbol. A few plots of the PDF is displayed to the right of figure 5.2.

From the PDF of T_d in equation (5.4) it can be seen that the scale parameter u only appears together with the critical level d as the product $d \cdot u$. As noted in [29], this means that we cannot estimate values for u and d separately. However, we can without loss of generality set $u = 1$, since it only is a scaling factor and the level of deterioration is a latent quantity. This leads to the density function

$$\begin{aligned} f_{T_d}(t; v(t), d) &= v'(t) \left[\Psi(v(t)) - \log(d)\right] \left(1 - \frac{\Gamma(v(t), d)}{\Gamma(v(t))}\right) \\ &+ \frac{v'(t)}{v(t)^2\Gamma(v(t))}(d)^{v(t)}{}_2F_2(v(t), v(t); v(t)+1, v(t)+1; -d) \end{aligned} \tag{5.5}$$

which is the one we will use later in this thesis.

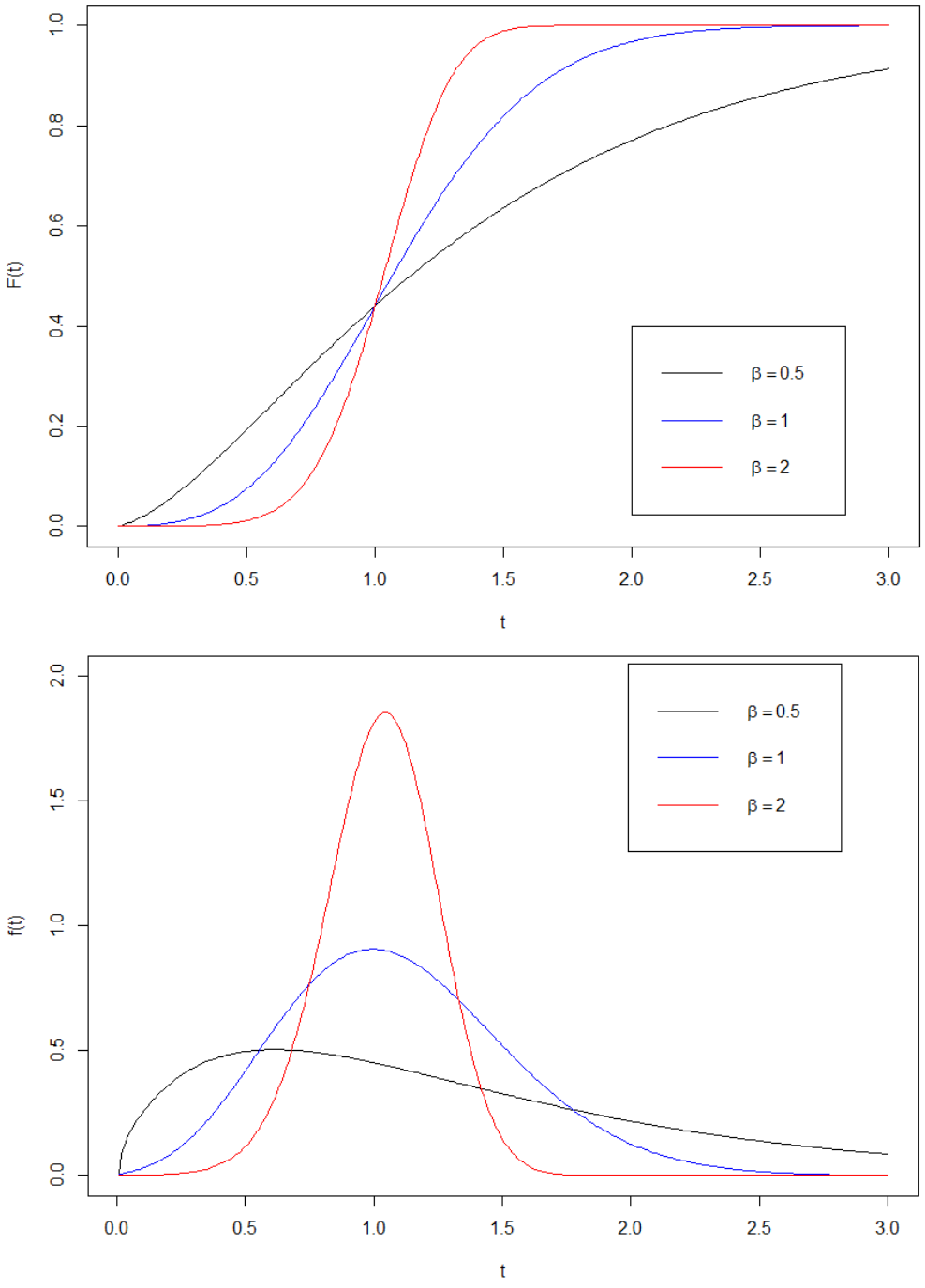

FIGURE 5.2: Examples of the CDF and PDF of the first passage time T_d for $u = 1$, $\alpha = 5$, $d = 5$ and different values of β

Chapter 6

Gamma process models

We have now introduced the most important concepts of both competing and semi-competing risks as well as the gamma process. In this chapter, we will define the gamma process degradation models that we will be studying for the rest of the thesis. These models represent an extended version of the model in my project thesis [35]. The general principles of the models are the same both if we are dealing with competing risks and semi-competing risks.

6.1 The basic model

The basic gamma process model from my master's project is not the focus of this thesis. Still, we wish to compare the results we get with the extended model to the results obtained by the basic model. Therefore, the general principles of the basic model will be repeated from the project thesis. Note that the basic model will only be defined for ordinary competing risks data, not semi-competing risks data. In particular, we only consider the situation described in section 3.1, where a component either fails at time X or is given preventive maintenance at time Z. In the basic model, we assume that the degradation of an item follows a gamma process $X(t)$ with shape function $v(t) = \alpha t^{\beta}$. As discussed in section 5.2, we may set the scale parameter $u = 1$ without loss of generality. The time to failure is equal to the first passage time to a specific level c in the gamma process. The time to preventive maintenance is equal to the first passage time to another, lower level s (see figure 6.1).

In section 3.5 it was explained why it is plausible to assume that when the gamma process reaches the level s, a signal indicating emerging failure is emitted. Whether or not we observe this signal is determined by a draw that is independent of the deterioration process. If the signal is detected, preventive maintenance will be issued, and we observe T_s, the first passage time to the level s, and set $Z = T_s$. If the signal is not detected, then the gamma process will continue until it reaches the level c, where the item will fail. We then observe T_c, the first passage time to the level c, and set $X = T_c$. In this case, one can imagine that $Z = T_v$, where $v > c$ so

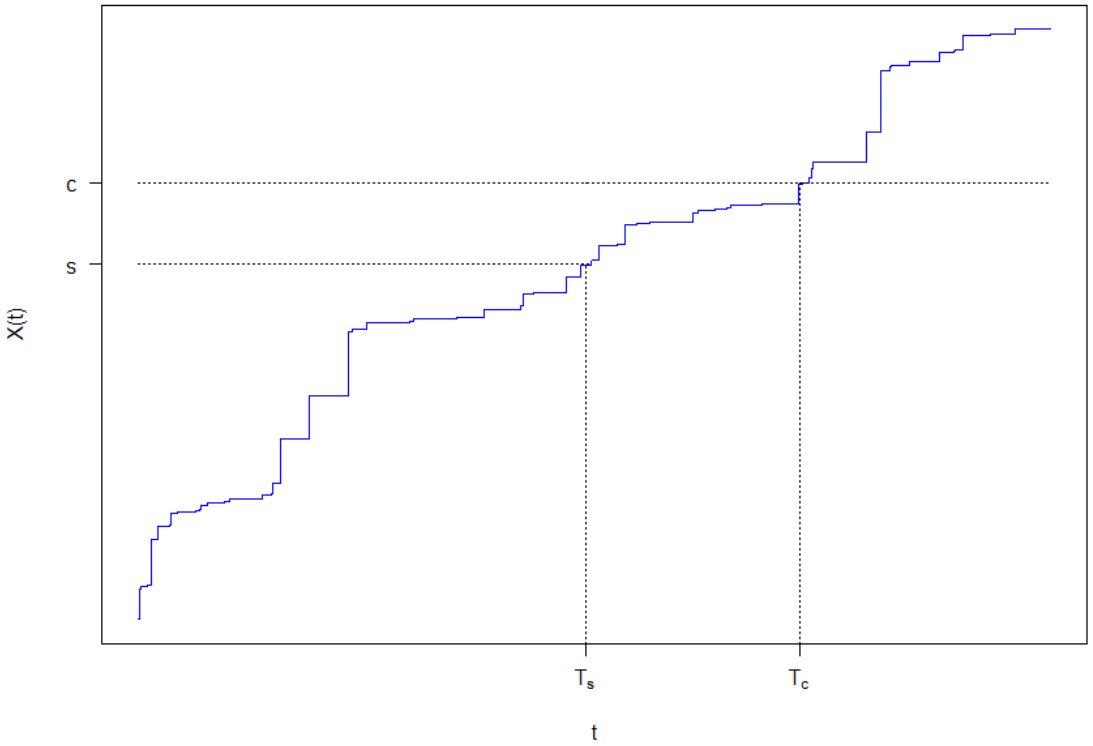

FIGURE 6.1: Relation between the thresholds s and c and the first passage times T_s and T_c in the basic gamma process model

that $Z > X$. This value T_v will never be observed and is only introduced in order to have a completely described joint distribution of (X, Z). Since Z and X originate from the same gamma process, they are dependent.

The probability of observing the emitted signal at level s will be equal to $q = P(Z < X)$. From the definition of random signs censoring in section 3.5, it is clear that the basic gamma process model is a random signs censoring model. The potential failure time X follows the distribution $f_{T_c}(t; v(t), c)$ where $f_{T_c}(\cdot)$ is as defined in equation (5.5). Note that for Z, $f_{T_s}(t; v(t), s)$ is the conditional distribution of Z given $Z < X$, and not the distribution of Z itself. $f_{T_s}(\cdot)$ is also given by equation (5.5). The log-likelihood function for the basic model was deduced in the project thesis, and is included in appendix F.2.

It should be mentioned that in addition to being used in the project thesis, an equivalent model only with Wiener processes has been used before in an article by Lindqvist and Skogsrud [25] and in a Master's thesis by Skogsrud [36]. A similar model has actually also been suggested by Horrocks and Thompson [16]. They considered a Wiener process model for the health status of patients. Upon reaching certain levels, the patients would either be discharged from the hospital or dead.

In addition, when the health status reached an intermediate level, a decision to transfer the patient was made with probability p. As it is pointed out in [25], this corresponds to the situation in our model with PM possibly being done before the level of failure is reached.

6.2 Extending the basic model

A possible extension of the basic gamma process model is to include random effects. One can often observe unexplained differences in the rate of degradation for the items under study. These differences may be obvious, even though the items under observation are identical, receive the same treatment and operate in the same environment. A model that incorporates random effects, for instance frailty models, allows for such unexplained differences.

Random effects are usually integrated in a model by making one or more parameters into a random variable. In the basic gamma process model from the project thesis there are many parameters to choose from. Here we will look more closely at letting the threshold parameter s be random, just like in the article by Lindqvist and Skogsrud [25] for the Wiener process model. We also mentioned it as a possibility for the gamma process model in the project thesis, but there it was not further explored. Of course, we could have chosen to let a different parameter be random, but by choosing s we can compare our results with the ones in [25]. Before we define our specific models, we present some of the work done on gamma process models with a random threshold parameter in the past. There are many other possible extensions of the basic model as well, some of these are discussed in section 10.2.

6.2.1 Gamma process models with random threshold

Models for the first passage time in a gamma process with random threshold is not a new concept, it has been studied by many others. As mentioned in chapter 5, Abdel Hameed was the first to suggest the gamma process as a model for deterioration occurring random in time in 1975. He has since then used the gamma process as a basis to develop multiple mathematical models for optimising time-based maintenance and condition-based maintenance [38]. Among these models, the case of random failure threshold was studied. After Abdel Hameed, several other authors have also studied the problem of finding the first passage time distribution to a random level in the gamma process. A notable example, that is similar to our problem can be found in an article by Paroissin and Salami from 2014 [29]. In their gamma process model there is however only one threshold, which represents item failure, i.e. there is no competing risk. In their article, they assign a probability distribution

to the random threshold variable C. The distribution of C describes at what level of degradation the item in question is most likely to fail. In particular, Paroissin and Salami present models where C is exponentially or gamma distributed, and independent of the gamma process. These distributions were chosen for practical reasons so that an analytical expression for the first passage time distribution could be found [29] (as we will see in section 6.4).

6.3 General description - random S models

Our goal is thereby to extend the basic gamma process model from the project thesis by letting the level s be a random variable S (while the level c is still fixed). This means in effect that the components (or people) under study are heterogeneous with respect to the level S. In contrast, the level s was the same to all components in the basic model. We will in the following refer to this new, extended model as the random S model. Since we may choose several different distributions for S, we will also often use the term random S models - in plural. At the end of the chapter we will return to discussing how we may interpret the distribution of S.

As previously mentioned, we want to model both dependent competing risks and semi-competing risks with the random S model. Now, in both of these settings we have two types of events, non-terminal and terminal. For the semi-competing risks case, this was described in detail in chapter 4. In the competing risks case we still only consider the situation of PM (non-terminal event) versus component failure (terminal event). Even though these two settings are quite different from each other, the following fundamental principles of the model are the same regardless of the setting.

First of all, we assume that the state of an item (or a person) follows a gamma process, $X(t)$, as defined in definition 5.1, with shape function $v(t) = \alpha t^{\beta}$. As in the basic model, we will let the scale parameter $u = 1$. Further, we will let the time to the terminal event equal T_c, the first passage time to the level c. The time to the non-terminal event will equal T_S, the first passage time to the level S, where S is stochastic. If T_S is observed, we set $Z = T_S$. Similarly, if T_c is observed we set $X = T_c$. Furthermore, if we assume that S is independent of the gamma process, we still have a random signs censoring model (in accordance with definition 3.2). The event $Z < X$ now corresponds to $S < c$, which is independent of T_c. Thus, the only difference from the basic model in the project thesis is that S is a random variable rather than a constant.

In the following, we denote the probability density function of S by $f_S(s)$. Just as in the basic model, the time to the terminal event follows the distribution $f_{T_c}(t; v(t), c)$, where $f_{T_c}(\cdot)$ is as defined in equation (5.5). Again we stress that for the time to the non-terminal event, $f_{T_S}(t; v(t), S)$ is the conditional distribution given that $Z < X$, and not the distribution of Z itself.

In addition to being an extension of the basic model, the random S model can be viewed as an extension of the one-threshold problem considered by Paroissin and Salami in [29]. The main difference is of course the extra threshold, but another distinction from that problem is that they used degradation data, where the degradation process is observed at several points in time, whereas we are dealing with lifetime data so that only the event times or first passage times are observed.

6.4 Random S model in the competing risks setting

We will now describe the model in the competing risks setting, and deduce some relevant functions. As previously mentioned, we will consider the possibility of failure at time X vs. preventive maintenance at time Z. We observe $Z = T_s$ if $S < c$ and $X = T_c$ if $S > c$.

For simplicity we begin by considering a situation where there are no censored observations. We will later expand the model description to include the possibility of censoring. Hence, what we observe is the pair (Y, C) where $Y = \min(Z, X)$ and C is the cause variable given by:

$$C = \begin{cases} 1 & \text{if } X \text{ is observed} \\ 0 & \text{if } Z \text{ is observed} \end{cases}$$

We now want to find the joint distribution of (Y, C) in order to evaluate the likelihood function of the model. Since the underlying process is the same for X and Z, they share the same shape function $v(t)$ and scale parameter u.

The contribution to the likelihood function when X is observed is the sub-density function for X, $f_X^*(x)$. To get the expression for this sub-density function, we will follow the same steps as in the project thesis and first find the sub-survival function $S_X^*(x)$ and then differentiate it. Because of the random signs property, the event $X < Z$ is independent of T_c and the expression for $S_X^*(x)$ becomes fairly straight

forward, and similar to what it was in the basic model:

$$\begin{aligned} S_X^*(x) &= P(X > x, C = 1) \\ &= P(X > x, X < Z) \\ &= P(T_c > x, S > c) \\ &= P(T_c > x)P(S > c) \\ &= S_{T_c}(x; v(x), c)(1 - F_S(c)) \end{aligned}$$

By differentiating, we get

$$\begin{aligned} f_X^*(x) &= -S'_{T_c}(x; v(x), c)(1 - F_S(c)) \\ &= f_{T_c}(x; v(x), c)(1 - F_S(c)) \end{aligned} \tag{6.1}$$

where $f_{T_c}(x; v(x), c)$ is the PDF defined in equation (5.5). As it is explained in the thesis by Skogsrud [36], the sub-functions of Z are a little more complicated than what they were in the basic model, since T_S and S are not independent like they were there. We get

$$\begin{aligned} S_Z^*(z) &= P(Z > z, C = 0) \\ &= P(Z > z, Z < X) \\ &= P(T_S > z, S < c) \\ &= \int_0^c P(T_S > z, s \leq S \leq s + ds) \\ &= \int_0^c P(T_s > z, s \leq S \leq s + ds) \\ &= \int_0^c P(T_s > z)P(s \leq S \leq s + ds) \\ &= \int_0^c S_{T_s}(z; v(z), s) f_S(s) ds \end{aligned} \tag{6.2}$$

and by differentiating

$$f_Z^*(z) = \int_0^c f_{T_s}(z; v(z), s) f_S(s) ds \tag{6.3}$$

At this point we face a challenge, as integrating $f_{T_s}(z; v(z), s)$ is no simple task. The problem is similar to that studied by Paroissin and Salami in [29], but at the same time significantly different. The model discussed there does not have any competing risks, so it is the level c that is random and there are no other levels. In

that case one can integrate from 0 to ∞ to find the first passage time distribution to the random level. With ∞ as the upper limit, there are formulas, for instance in [2], that could help us find the exact expression for $f_Z^*(z)$, at least for some specific choices of $f_S(s)$ like the exponential distribution or the gamma distribution.

The problem of finding an expression for $f_Z^*(z)$ was also simpler in the Master's thesis by Skogsrud [36] where a Wiener process model with random level S was considered. In that case, one could calculate the integral in (6.3) relatively easy with certain distributions of S since $f_{T_s}(\cdot)$ was the inverse Gaussian distribution. In our case, the situation seems to be a little more complicated because of the complexity of $f_{T_s}(\cdot)$. A gamma process degradation model was also suggested by Park and Padgett in [28], and they stated that "Although we obtained the exact distribution of the first passage time to the threshold, the distribution is so complicated that it is very difficult to compute in practice"[1]. Therefore we will solve the integral in (6.3) numerically. Even though this leads to some additional uncertainty, the method may still provide good results.

It is however interesting to note that if we were to choose a uniformly distributed S, we can obtain an analytical expression for $f_Z^*(z)$. With $S \sim Unif(0, A)$, it can be shown (Wolfram Alpha [42]) that

$$\int f_{T_s}(z; v(z), s)\frac{1}{A}ds$$

$$= \frac{1}{A}\frac{1}{\Gamma(v(z))}v'(z)\left(-\frac{s^{v(z)+1}{}_2F_2(v(z)+1, v(z)+1; v(z)+2, v(z)+2; -s)}{(v(z)+1)^2}\right.$$

$$+G_{3,4}^{3,1}\left(s\,\middle|\,\begin{matrix}1,2,2\\1,1,v(z)+1,0\end{matrix}\right) - s\Gamma(v(z), s) + \Gamma(v(z)+1, s) + s\log(s)\Gamma(v(z), s)$$

$$\left.-s\Psi(v(z))\Gamma(v(z), s) + \Psi(v(z))\Gamma(v(z)+1, s) + \log(s)\Gamma(v(z)+1, 0, s)\right) + \text{constant}$$

Here, $\Gamma(x)$ is the gamma function as before, and $\Gamma(a, x)$ is the incomplete gamma function. $\Gamma(a, x_0, x_1)$ denotes the generalized incomplete gamma function, $\Gamma(a, x_0) - \Gamma(a, x_1)$. $\Psi(x)$ is still the digamma function and ${}_pF_q(a_1, ..., a_p; b_1, ..., b_q; z)$ is the generalized hypergeometric function of order (p, q). $G_{p,q}^{m,n}\left(z\,\middle|\,\begin{matrix}a_1,...,a_p\\b_1,...,b_q\end{matrix}\right)$ is the Meijer's

[1] Park and Padgett ended up approximating the first passage time distribution of the gamma process by an inverse Gaussion distribution.

G function:

$$G_{p,q}^{m,n}\left(z\,\middle|\begin{matrix}a_1,\ldots,a_p\\ b_1,\ldots,b_q\end{matrix}\right)=\frac{1}{2\pi i}\int_L \frac{\prod_{j=1}^{m}\Gamma(b_j-s)\prod_{j=1}^{n}\Gamma(1-a_j+s)}{\prod_{j=m+1}^{q}\Gamma(1-b_j+s)\prod_{j=n+1}^{p}\Gamma(a_j-s)}\,z^s\,ds$$

This is a very general function that includes many special functions as particular cases. It is even more general than the generalized hypergeometric function [41]. According to [15], the Meijer's G-function is not yet implemented in R. Thus, we are not able use this result directly after all.

We can now take a look at what the likelihood functions look like in the model with random S. With observations of failure times $x_1, ..., x_m$ and observations of time to PM $z_1, ..., z_n$ the likelihood function becomes the product of the sub-densities:

$$\begin{aligned}
L &= \prod_{i=1}^{m} f_X^*(x_i)\prod_{j=1}^{n} f_Z^*(z_j)\\
&= \prod_{i=1}^{m}(1-F_S(c))f_{T_c}(x_i;v(x_i),c)\prod_{j=1}^{n} f_Z^*(z_j)\\
&= (1-F_S(c))^m\prod_{i=1}^{m}\Bigg\{v'(x_i)\left[\Psi(x_i)-\log(c)\right]\left(1-\frac{\Gamma(v(x_i),c)}{\Gamma(v(x_i))}\right)\\
&+ \frac{v'(x_i)}{v(x_i)^2\Gamma(v(x_i))}c^{v(x_i)}\cdot{}_2F_2(v(x_i),v(x_i);v(x_i)+1,v(x_i)+1;-c)\Bigg\}\prod_{j=1}^{n} f_Z^*(z_j)
\end{aligned}$$

where $f_Z^*(z_j)$ is as given by equation (6.3), and will be different for each choice of distribution $f_S(s)$. The corresponding log-likelihood function becomes

$$\begin{aligned}
l &= \ln L\\
&= m\cdot\ln(1-F_S(c))+\sum_{i=1}^{m}\ln\Bigg\{v'(x_i)\left[\Psi(x_i)-\log(c)\right]\left(1-\frac{\Gamma(v(x_i),c)}{\Gamma(v(x_i))}\right)\\
&+ \frac{v'(x_i)}{v(x_i)^2\Gamma(v(x_i))}c^{v(x_i)}{}_2F_2(v(x_i),v(x_i);v(x_i)+1,v(x_i)+1;-c)\Bigg\}+\sum_{j=1}^{n} f_Z^*(z_j)
\end{aligned}$$

Including censoring

For many datasets, there will be some observations that are right censored, so that we observe neither Z nor X, but rather a censoring time τ. This may for instance be due to that items are removed from the study before it has ended, or that the study has ended before PM or failure is observed for an item. The censoring time τ may thereby vary for the different observations. This kind of censoring is assumed

to occur independently of the gamma process. We now assume that we observe failure times $x_1, ..., x_m$, times to PM $z_1, ..., z_n$ and censoring times $\tau_1, ..., \tau_r$. For the censored observations we know that $\min(Z, X) > \tau$, but we do not know if a PM or a failure would have happened had the censoring not occurred. As seen in section A.2.1 in appendix A, the contribution to the likelihood function from the censored observations will be given by the survival function. We get

$$\begin{aligned}
P(X > \tau, Z > \tau) &= P(T_c > \tau, T_S > \tau) \\
&= \int_0^\infty P(T_c > \tau, T_S > \tau, s \leq S \leq s + ds) \\
&= \int_0^c P(T_c > \tau, T_s > \tau) f_S(s) ds + \int_c^\infty P(T_c > \tau, T_s > \tau) f_S(s) ds \\
&= \int_0^c P(T_s > \tau) f_S(s) ds + P(T_c > \tau) \int_c^\infty f_S(s) ds \\
&= \int_0^c S_{T_s}(\tau; v(\tau), s) f_S(s) ds + S_{T_c}(\tau; v(\tau), c)(1 - F_S(c)) \\
&= \int_0^c \left(1 - \frac{\Gamma(v(\tau), s)}{\Gamma(v(\tau))}\right) f_S(s) ds + \left(1 - \frac{\Gamma(v(\tau), c)}{\Gamma(v(\tau))}\right)(1 - F_S(c))
\end{aligned} \tag{6.4}$$

The complete likelihood function with observations $x_1, ..., x_m$, $z_1, ..., z_n$ and $\tau_1, ..., \tau_r$ becomes

$$\begin{aligned}
L &= \prod_{i=1}^m f_X^*(x_i) \prod_{j=1}^n f_Z^*(z_j) \prod_{k=1}^r P(X > \tau_k, Z > \tau_k) \\
&= (1 - F_S(c))^m \prod_{i=1}^m \left\{ v'(x_i) \left[\Psi(x_i) - \log(c)\right] \left(1 - \frac{\Gamma(v(x_i), c)}{\Gamma(v(x_i))}\right) + \frac{v'(x_i)}{v(x_i)^2 \Gamma(v(x_i))} c^{v(x_i)} \right. \\
&\quad \left. \cdot {}_2F_2(v(x_i), v(x_i); v(x_i) + 1, v(x_i) + 1; -c) \right\} \prod_{j=1}^n f_Z^*(z_j) \prod_{k=1}^r P(X > \tau_k, Z > \tau_k)
\end{aligned} \tag{6.5}$$

where both $f_Z^*(z_j)$ and $P(X > \tau_k, Z > \tau_k)$ will be different depending on the distribution of S. The expression for $f_Z^*(z_j)$ is as before given in equation (6.3), while $P(X > \tau_k, Z > \tau_k)$ is given by (6.4). The corresponding log-likelihood function

becomes

$$l = m \cdot \ln(1 - F_S(c)) + \sum_{i=1}^{m} \ln \left\{ v'(x_i) \left[\Psi(x_i) - \log(c) \right] \left(1 - \frac{\Gamma(v(x_i), c)}{\Gamma(v(x_i))} \right) + \frac{v'(x_i)}{v(x_i)^2 \Gamma(v(x_i))} c^{v(x_i)} {}_2F_2(v(x_i), v(x_i); v(x_i)+1, v(x_i)+1; -c) \right\} + \sum_{j=1}^{n} \ln f_Z^*(z_j) + \sum_{k=1}^{r} \ln P(X > \tau_k, Z > \tau_k) \tag{6.6}$$

The log-likelihood function in (6.6) will later be used to estimate the parameters of the model with random S including censoring for competing risks data. Since this function is pretty nasty to differentiate for most of the parameters, we will find the parameter estimates numerically. The function we will use to do this in R is `condSurv()` which again calls the `optim()`-function. It is the same function that was used in the project thesis, but it has been expanded to handle the random S models in addition to the basic model. The `condSurv()` function can be found in appendix D.2.1.

6.4.1 Parametric estimation of conditional sub-survival functions in the gamma process models

In later chapters (the simulation study and the data analysis) we will plot the non-parametric estimates of the conditional sub-survival functions resulting from equation (3.4) to check the random signs property, as described in section 3.5. We will also estimate the parameters of the gamma process models. A way to check how good our parameter estimates are is then to plot the parametrically estimated conditional sub-survival curves and compare them to the non-parametric ones. As noted in equation (3.5) random signs censoring implies that $\tilde{S}_X(t) = S_X(t)$. Thus, we can use $\hat{S}_{T_c}(t, v(t), c)$ as a parametric estimate for $\tilde{S}_X(t)$, where $\hat{S}_{T_c}(\cdot)$ is the distribution defined in equation (5.3) with inserted parameter estimates $\alpha = \hat{\alpha}, \beta = \hat{\beta}$ and $d = \hat{c}$ (and $u = 1$).

For Z there is no equivalent implication as (3.5). However, in section 3.3.2 we saw that the non-parametric estimate of the conditional sub-distribution function was given by

$$\hat{\tilde{F}}_Z(t) = \frac{\hat{F}_Z^*(t)}{\hat{q}} \quad \text{or equivalently} \quad \hat{\tilde{S}}_Z(t) = \frac{\hat{S}_Z^*(t)}{\hat{q}}$$

where $\hat{q} = P(\widehat{Z < X}) = \hat{S}_Z^*(0)$. For our gamma process models we found in equation (6.2) that $\hat{S}_Z^*(t) = \int_0^{\hat{c}} \hat{S}_{T_s}(t; v(t), s) \hat{f}_S(s) ds$. Thus, this estimate will be different for

each choice of distribution for S. Here $\hat{S}_{T_s}(\cdot)$ again is the distribution defined in equation (5.3), this time with with inserted parameter estimates $\alpha = \hat{\alpha}, \beta = \hat{\beta}$ and $d = \hat{s}$ (and $u = 1$). A parametric estimate of $\hat{S}^*_Z(0)$ is $\widehat{P(S < c)} = \hat{F}_S(c)$, the distribution function of S with inserted parameter estimates. Thus,

$$\hat{\bar{S}}_Z(t) = \frac{\hat{S}^*_Z(t)}{\hat{S}^*_Z(0)} = \frac{1}{\hat{F}_S(c)} \int_0^{\hat{c}} \hat{S}_{T_s}(t; v(t), s) \hat{f}_S(s) ds$$

In R we will compute this using the function `subdistrZ()`, which is given in appendix D.2.2.

6.5 Random S model in the semi-competing risks setting

We will now consider model features that are special to the setting of semi-competing risks. What is described in section 6.2 is still the basis of the model. Now, in this situation the first passage time to the level c equals the time to the terminal event and the first passage time to the level S equals the time to the non-terminal event. As we recall from section 4.2, the data structure is different from that of competing risks. This makes the likelihood function more complicated to deduce. If we include the possibility of some observations being right censored, we have four possible scenarios. These were presented in table 4.1, and we have to figure out how the observations in each of these cases contribute to the likelihood.

1. **Observe both Z and X**
   The probability of observing both $Z = T_S$ and $X = T_c$ is equal to the probability that $T_S = t_1$, $T_c = t_2$ and that $S < c$. Then, we have that $t_1 < t_2$ and we can write:

$$P(t_1 \leq T_S \leq t_1 + dt_1, t_2 \leq T_c \leq t_2 + dt_2, S < c)$$

$$= \int_0^\infty P(t_1 \leq T_S \leq t_1 + dt_1, t_2 \leq T_c \leq t_2 + dt_2, S < c | S = s) f_S(s) ds$$

$$= \int_0^c P(t_1 \leq T_S \leq t_1 + dt_1, t_2 \leq T_c \leq t_2 + dt_2 | S = s) f_S(s) ds$$

   Now, because S is independent of the process we can write T_s instead of T_S and delete "$|S = s$":

$$= \int_0^c P(t_1 \leq T_s \leq t_1 + dt_1, t_2 \leq T_c \leq t_2 + dt_2) f_S(s) ds$$

$$= \int_0^c f_{T_s,T_c}(t_1,t_2) f_S(s) ds \tag{6.7}$$

where $f_{T_s,T_c}(t_1,t_2)$ is the joint distribution of (T_s,T_c). This distribution can be found by using the fact that the gamma process has independent increments. One can therefore imagine that after we have observed $T_s = t_1$, a "new" gamma process begins from level s. This second process will determine the distribution of T_c given $T_s = t_1$. Thus,

$$f_{T_s,T_c}(t_1,t_2) = P(t_1 \leq T_s \leq t_1 + dt_1, t_2 \leq T_c \leq t_2 + dt_2)$$

$$= P(t_1 \leq T_s \leq t_1 + dt_1) P(t_2 \leq T_c \leq t_2 + dt_2 | T_s = t_1)$$

Here, $P(t_1 \leq T_s \leq t_1 + dt_1) = f_{T_s}(t_1; v(t_1), s)$ is known from before. From definition 5.1 we know that $X(t_2) - X(t_1)$ is gamma distributed with shape parameter $v(t_2) - v(t_1)$ whenever $0 \leq t_1 \leq t_2$. Furthermore, we can write $T_s = t_1, T_c = t_1 + T_{c-s}$, where T_{c-s} denotes the first passage time to level c when beginning at level s instead of at 0. T_{c-s} follows the same first passage time distribution $f_{T_d}(\cdot)$ as in equation (5.5) with shape function $v(t_2) - v(t_1)$ and $d = c - s$. If we write this out, we get

$$\begin{aligned}
&P(t_2 \leq T_c \leq t_2 + dt_2 | T_s = t_1) = P(t_2 - t_1 \leq T_{c-s} \leq t_2 - t_1 + dt_2) \\
&= f_{T_{c-s}}(t_2 - t_1; v(t_2) - v(t_1), c - s) = \frac{d}{dt_2} \frac{\Gamma(v(t_2) - v(t_1), c - s)}{\Gamma(v(t_2) - v(t_1))} \\
&= v'(t_2) \left[\Psi(v(t_2) - v(t_1)) - \log(c - s) \right] \left(1 - \frac{\Gamma(v(t_2) - v(t_1), c - s)}{\Gamma(v(t_2) - v(t_1))} \right) \\
&+ \frac{v'(t_2)}{(v(t_2) - v(t_1))^2 \Gamma(v(t_2) - v(t_1))} (c - s)^{v(t_2) - v(t_1)} \\
&\cdot {}_2F_2(v(t_2) - v(t_1), v(t_2) - v(t_1); v(t_2) - v(t_1) + 1, v(t_2) - v(t_1) + 1; -(c - s))
\end{aligned}$$

Hence, we have found that

$$f_{T_s,T_c}(t_1,t_2) = f_{T_s}(t_1; v(t_1), s) f_{T_{c-s}}(t_2 - t_1; v(t_2) - v(t_1), c - s) \tag{6.8}$$

where both $f_{T_s}(t_1; v(t_1), s)$ and $f_{T_{c-s}}(t_2 - t_1; v(t_2) - v(t_1), c - s)$ follow the distribution given in equation (5.5) only with different input parameters.

By inserting (6.8) into (6.7), we get that the contribution to the likelihood function if we observe both T_S and T_c is

$$\int_0^c f_{T_s}(t_1; v(t_1), s) f_{T_{c-s}}(t_2 - t_1; v(t_2) - v(t_1), c - s) f_S(s) ds$$

2. **Observe only X**
   The probability of observing only X is equivalent to the probability of $T_S = t_1$, $T_c = t_2$ and that $S > c$. This causes $t_2 > t_1$ so that t_1 is never observed. We can write
   $$P(t_2 \leq T_c \leq t_2 + dt_2, S > c)$$
   As S is independent of the process, we have
   $$= P(t_2 \leq T_c \leq t_2 + dt_2)P(S < c)$$
   $$= f_{T_c}(t_2; v(t_2), c)(1 - F_S(c))$$

3. **Observe only Z and a censoring time τ**
   We observe Z and a censoring time τ if $T_S < \tau < T_c$. Thereby we must have that $S < c$ and $t_1 < \tau$. The probability of observing this is
   $$P(t_1 \leq T_S \leq t_1 + dt_1, T_c > \tau)$$
   $$= \int_0^c P(t_1 \leq T_S \leq t_1 + dt_1, T_c > \tau | S = s) f_S(s) ds$$
   $$= \int_0^c P(t_1 \leq T_s \leq t_1 + dt_1, T_c > \tau) f_S(s) ds$$
   $$= \int_0^c P(t_1 \leq T_s \leq t_1 + dt_1) P(T_c > \tau | T_s = t_1) f_S(s) ds$$
   $$= \left[\int_0^c f_{T_s}(t_1; v(t_1), s) P(T_c > \tau | T_s = t_1) f_S(s) ds \right] dt_1$$
   Here,
   $$P(T_c > \tau | T_s = t_1) = P(T_c - T_s > \tau - t_1 | T_s = t_1)$$
   which, because of independent increments:
   $$= P(T_c - T_s > \tau - t_1)$$
   This we already have an expression for when $c > s$: $P(T_c - T_s > \tau - t_1) = S_{T_{c-s}}(\tau - t_1; v(\tau) - v(t_1), c - s)$. Thus,
   $$P(t_1 \leq T_S \leq t_1 + dt_1, T_c > \tau)$$
   $$= \int_0^c f_{T_s}(t_1, v(t_1), s) S_{T_{c-s}}(\tau - t_1; v(\tau) - v(t_1), c - s) f_S(s) ds$$

4. **Observe only a censoring time τ**
   The probability of only observing a censoring time τ equals the probability that both T_S and T_c are greater than τ. We write

$$\begin{aligned} P(T_S > \tau, T_c > \tau) &= P(T_S > \tau, T_c > \tau, S > c) + P(T_S > \tau, T_c > \tau, S < c) \\ &= P(T_c > \tau, S > c) + P(T_S > \tau, S < c) \\ &= P(T_c > \tau)(1 - F_S(c)) + \int_0^c P(T_s > \tau) f_S(s) ds \\ &= S_{T_c}(\tau; v(\tau), c)(1 - F_S(c)) + \int_0^c S_{T_s}(\tau; v(\tau), s) f_S(s) ds \end{aligned}$$

In each of the four cases we get observations denoted in the following manner:

1. times to the non-terminating event, $z_1, ..., z_n$ and corresponding times to the terminating event $x_{z1}, ..., x_{zn}$
2. times to the terminating event $x_1, ..., x_m$
3. times to the non-terminating event $z_{o1}, ..., z_{os}$ and corresponding censoring times $\tau_{o1}, ..., \tau_{os}$
4. censoring times $\tau_1, ..., \tau_r$

We obtain the complete likelihood function from multiplying the contributions from all of the four cases together:

$$\begin{aligned} L = &\prod_{i=1}^{m} (1 - F_S(c)) f_{T_c}(x_i; v(x_i), c) \\ &\cdot \prod_{j=1}^{n} \int_0^c f_{T_s}(z_j; v(z_j), s) f_{T_{c-s}}(x_{zj} - z_j; v(x_{zj}) - v(z_j), c - s) f_S(s) ds \\ &\cdot \prod_{l=1}^{w} \int_0^c f_{T_s}(z_{ol}; v(z_{ol}), s) S_{T_{c-s}}(\tau_{ol} - z_{ol}; v(\tau_{ol}) - v(z_{ol}), c - s) f_S(s) ds \\ &\cdot \prod_{k=1}^{r} \left[S_{T_c}(\tau_k; v(\tau_k), c)(1 - F_S(c)) + \int_0^c S_{T_s}(\tau_k; v(\tau_k), s) f_S(s) ds \right] \end{aligned} \tag{6.9}$$

We get the log-likelihood function by taking the natural logarithm of the expression in (6.9):

$$\begin{aligned} l = {} & m \cdot \ln(1 - F_S(c)) + \sum_{i=1}^{m} \ln \{f_{T_c}(x_i; v(x_i), c)\} \\ & + \sum_{j=1}^{n} \ln \left\{ \int_0^c f_{T_s}(z_j; v(z_j), s) f_{T_{c-s}}(x_{zj} - z_j; v(x_{zj}) - v(z_j), c - s) f_S(s) ds \right\} \\ & + \sum_{l=1}^{w} \ln \left\{ \int_0^c f_{T_s}(z_{ol}, v(z_{ol}), s) S_{T_{c-s}}(\tau_{ol} - z_{ol}; v(\tau_{ol}) - v(z_{ol}), c - s) f_S(s) ds \right\} \\ & + \sum_{k=1}^{r} \ln \left\{ \left[S_{T_c}(\tau_k; v(\tau_k), c)(1 - F_S(c)) + \int_0^c S_{T_s}(\tau_k; v(\tau_k), s) f_S(s) ds \right] \right\} \end{aligned} \tag{6.10}$$

If there are no censored observations, we only get the first two types of observations, namely times to the non-terminating event, $z_1, ..., z_n$ and corresponding times to the terminating event $x_{z1}, ..., x_{zn}$ or only times to the terminating event $x_1, ..., x_m$. Then, the log-likelihood function is simply

$$\begin{aligned} l = {} & m \cdot \ln(1 - F_S(c)) + \sum_{i=1}^{m} \ln f_{T_c}(x_i; v(x_i), c) \\ & + \sum_{j=1}^{n} \ln \left\{ \int_0^c f_{T_s}(z_j; v(z_j), s) f_{T_{c-s}}(x_{zj} - z_j; v(x_{zj}) - v(z_j), c - s) f_S(s) ds \right\} \end{aligned} \tag{6.11}$$

These are the log-likelihood functions that we will use later in the thesis to find parameter estimates in the gamma process models with random S for semi-competing risks data. To do this we have implemented the function `estSemi`() in R, which in turn calls the `optim`() function. The R-code for `estSemi`() is included in appendix D.2.3.

6.5.1 Parametric estimation of crude and net quantities in the gamma process models

As discussed in section 4.3, we will estimate both crude and net quantities in our analysis of semi-competing risks data. Like [31], we are mainly interested in the quantities associated with the time to the non-terminal event.

Crude quantities

The crude quantities that we will estimate are the cause specific hazard, described by equation (4.2), and the cumulative incidence function from equation (4.1). In section 4.4, non-parametric estimates of these quantities for the non-terminal event

were given in equations (4.3) and (4.4) respectively. We have already presented a parametric expression for the sub-survival function $S_Z^*(t)$ in equation (6.2), so the parametric estimate of the cumulative incidence function is easily found to be

$$\hat{F}_Z^*(t) = 1 - \hat{S}_Z^*(t) = 1 - \int_0^{\hat{c}} \hat{S}_{T_s}(t; v(t), s)\hat{f}_S(s)ds \tag{6.12}$$

Here, $\hat{S}_{T_s}(\cdot)$ and $\hat{f}_S(\cdot)$ denote the distribution in (5.3) and the chosen distribution of S with inserted parameter estimates for α, β and the parameters associated with $f_S(s)$. A parametric estimate for $\lambda_Z^*(t)$ can be found from

$$\hat{\lambda}_Z^*(t) = \frac{\hat{f}_Z^*(t)}{\hat{S}(t)} \tag{6.13}$$

where $\hat{f}_Z^*(t)$ as before is the expression from (6.3) with inserted parameter estimates, and

$$\begin{aligned} S(t) &= P(T_c > t, T_S > t) \\ &= P(T_c > t)(1 - F_S(c)) + \int_0^c P(T_s > t) f_S(s)ds \end{aligned}$$

as shown earlier in this section. Hence

$$\hat{S}(t) = \hat{S}_{T_c}(t; v(t), c)(1 - \hat{F}_S(c)) + \int_0^{\hat{c}} \hat{S}_{T_s}(t; v(t), s)\hat{f}_S(s)ds$$

The cumulative hazard rate $\hat{\Lambda}_Z^*(t)$ is found by integrating $\hat{\lambda}_Z^*(t)$.

Net quantities

We have already got an expression for the marginal survival function of the time to the terminal event, $\hat{S}_X(t) = \hat{S}_{T_c}(t; v(t), c)$. As mentioned in section 4.3.2 it might also be of interest to estimate the marginal distribution of the time to the non-terminal event. The marginal survival function of Z can in our case be found

as

$$\begin{aligned} S_Z(t) &= P(T_S > t) \\ &= \int_0^\infty P(T_S > t|S = s) f_S(s) ds \\ &= \int_0^\infty P(T_s > t) f_S(s) ds \\ &= \int_0^\infty S_{T_s}(t; v(t), s) f_S(s) ds \end{aligned} \tag{6.14}$$

The marginal hazard rate for Z can be expressed as

$$\begin{aligned} \lambda_Z(t) &= \lim_{h\to 0} \frac{P(T_S \leq t + \Delta t, |T_S > t)}{\Delta t} \\ &= \lim_{h\to 0} \frac{P(t \leq T_S \leq t + \Delta t)}{\Delta t P(T_S > t)} \\ &= \frac{f_{T_S}(t; v(t), S)}{S_{T_S}(t; v(t), S)} \\ &= \frac{\int_0^\infty f_{T_S}(t; v(t), S|S = s) f_S(s) ds}{\int_0^\infty S_{T_S}(t; v(t), S|S = s) f_S(s) ds} \\ &= \frac{\int_0^\infty f_{T_s}(t; v(t), s) f_S(s) ds}{\int_0^\infty S_{T_s}(t; v(t), s) f_S(s) ds} \end{aligned} \tag{6.15}$$

Thereby, a parametric estimate of $S_Z(t)$ can be found by inserting parameter estimates into the expression in (6.14), and similarly for $\lambda_Z(t)$ by inserting parameter estimates into (6.15). Following the same steps as above we find that a parametric expression for the marginal hazard rate for X is given by

$$\lambda_X(t) = \frac{f_{T_c}(t; v(t), c)}{S_{T_c}(t; v(t), c)} \tag{6.16}$$

Both the crude and the net quantities will be estimated in R by the script `semiQuant` which is included in appendix D.2.3.

6.6 Choice of distribution $f_S(s)$

The distribution of S will have a different meaning or interpretation depending on if we are in the competing risks setting or in the semi-competing risks setting. In the competing risks case of PM vs. failure, the probability distribution of S describes at what levels of degradation the signal that indicates an emerging failure is most likely to be emitted. Hence, heterogeneity with respect to the level S may

represent differences in maintenance policy. For instance, the distribution of S may be interpreted as a distribution of when (at what levels) the maintenance crew will check the item.

In the semi-competing risks situation the other hand, the distribution of S describes the tendency of the non-terminal event. An example is if we consider the case when the non-terminal event is cancer relapse and the terminal event is death. The distribution of S then represents the tendency a patient has to relapse.

In either setting, one could say that the basic model is a special case of the random S model. In the basic model, the distribution of S is discrete, and contains two point probabilities. S can either take the value s with probability q, or another value that is greater than c with probability $1 - q$. If we denote this second level by v we get the distribution

$$f_S(s) = \begin{cases} s & \text{with probability} \quad q \\ v & \text{with probability} \quad 1-q \end{cases} \tag{6.17}$$

When we are to select a suitable distribution for S in the random S model, we first of all need to have that $P(S > 0) = 1$ for all s, as 0 is the starting point of our degradation process. We should also keep in mind that the distribution should not contain too many parameters, as this will make parameter estimation more difficult. Two very simple distributions that satisfy these criteria are the uniform distribution (with lower limit $= 0$) and the exponential distribution. If S is uniformly distributed, this would mean that it is equally likely for the non-terminal event to occur at all times. An exponentially distributed S on the other hand suggests that it is more likely to occur at the beginning of an item's lifetime. For many applications, neither of these choices are very realistic. If we for instance consider PM as the non-terminal event, then ideally PM should be performed just before a critical failure. In that sense, a gamma distribution or a lognormal distribution may be more suitable choices. They would be more flexible too, as they can take on a range of different shapes. One can use a similar argument if the non-terminal event is a disease recurrence. For most illnesses, this is more likely to occur after some time has passed from treatment.

As mentioned before, in [29] Paroissin and Salami selected the exponential and the gamma distribution to make their PDF of the first passage time to the critical level into an analytically solvable integral. In a similar manner, Skogsrud in her thesis (with ordinary competing risks) [36] suggested two distributions that would make

$f_Z^*(z)$ with the Wiener process into pleasant expressions. She chose the uniform distribution and a truncated normal distribution with mean just before the critical level c. Neither of the Wiener-process models fitted the data very well, but the models where S was normally distributed gave the best fit.

Here, we are not too constricted by having to choose distributions that will make $f_Z^*(z)$ in the competing risks case into a nice expression, as we will solve the associated integral numerically. Still, one should not underestimate the advantage of a simple distribution with few parameters. That is why we have ended up with choosing to test the uniform, exponential, gamma and lognormal distributions. There are probably several other distributions that also would be suitable.

6.6.1 Uniform S

We will first consider the case $f_S(s) \sim Unif(0, A)$, $A > c$. We will from now on refer to this as the uniform model. This situation is illustrated in figure 6.2. There, two different realisations of s are marked. If $s_1 < c$ is drawn, we will observe only T_{s_1} in the competing risks case, but both T_{s_1} and T_c in the semi-competing risks case. If $s_2 > c$ is drawn, we will only observe the failure time T_c. T_{s_2} is never observed.

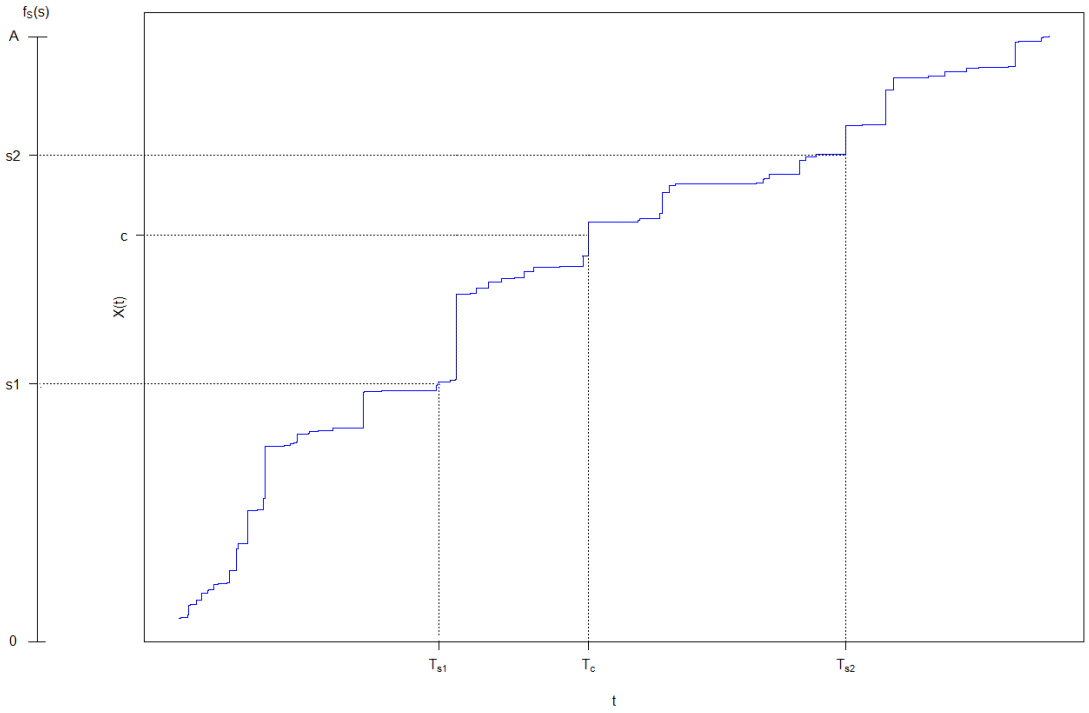

FIGURE 6.2: Illustration of the case of a gamma process $X(t)$ with a fixed level c and a uniformly distributed level S on $[0, A]$

When S is uniformly distributed, the density and distribution functions $f_S(s)$ and $F_S(s)$ are simply

$$f_S(s) = \frac{1}{A} \tag{6.18}$$

$$F_S(s) = P(S \leq s) = \frac{s}{A} \tag{6.19}$$

Thus, $F_S(c) = \frac{c}{A}$.

6.6.2 Exponentially distributed S

We now move on to a case where S is exponentially distributed with parameter λ_S, $S \sim \exp(\lambda_S)$. This situation is illustrated in figure 6.3. We will in the following refer to this as the exponential model. Again, if $s_1 < c$ is drawn, we will observe only T_{s_1} in the competing risks case, but both T_{s_1} and T_c in the semi-competing risks case. If $s_2 > c$ is drawn, we will observe the failure time T_c, not T_{s_2}.

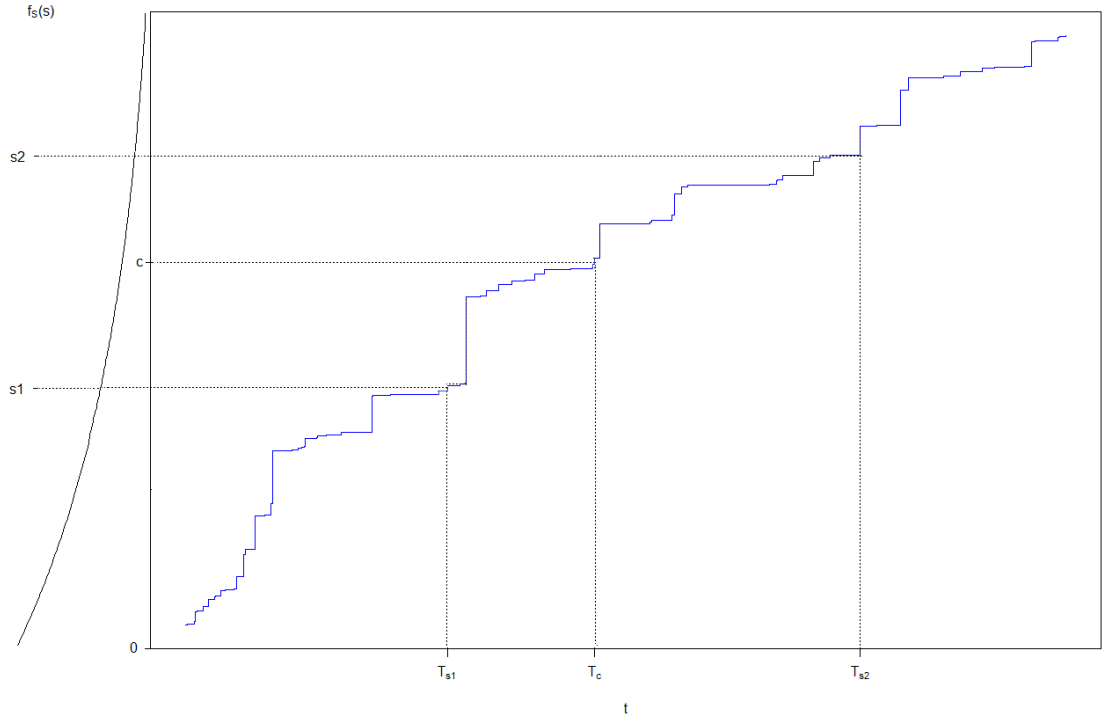

FIGURE 6.3: Illustration of the case of a gamma process $X(t)$ with a fixed level c and an exponentially distributed level S

When S is exponentially distributed, the PDF $f_S(s)$ and CDF $F_S(s)$ are given by

$$f_S(s) = \lambda_S e^{-\lambda_S s} \tag{6.20}$$

$$F_S(s) = 1 - e^{-\lambda_S s} \tag{6.21}$$

so that $F_S(c) = 1 - e^{-\lambda_S c}$.

6.6.3 Gamma distributed S

We now let S be gamma distributed with shape parameter α_S and scale parameter β_S, $S \sim Ga(\alpha_S, \beta_S)$. For the rest of the thesis we will refer to this as the gamma model. An example of how this may look is given in figure 6.4. Here we have chosen a gamma distribution with mean value $= c$ and relatively small variance so that the chance of the non-terminating event occurring very early or very late is small. Just as before, we have marked two different realisations of s. If $s_1 < c$ is drawn, we will observe T_{s_1} in the competing risks case or both T_{s_1} and T_c in the semi-competing risks case. If $s_2 > c$ is drawn, we will observe the failure time T_c, never T_{s_2}.

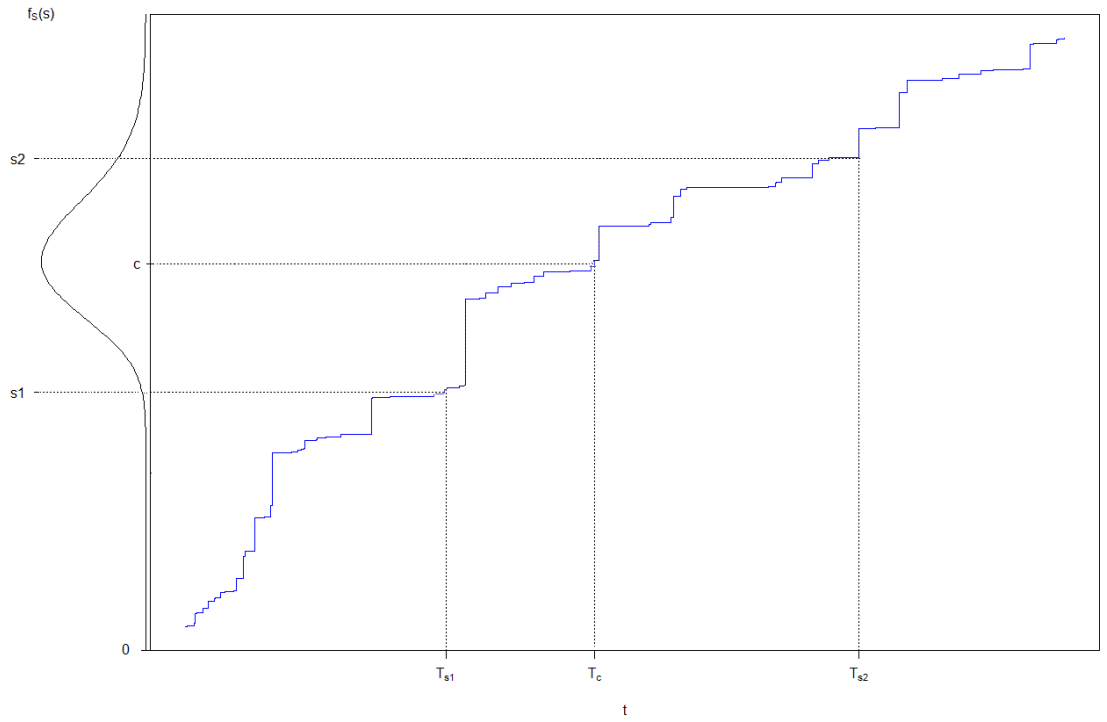

FIGURE 6.4: Illustration of the case of a gamma process $X(t)$ with a fixed level c and a gamma distributed level S

With a gamma distributed S the PDF $f_S(s)$ and CDF $F_S(s)$ are given by

$$f_S(s) = \frac{\beta_S^{\alpha_S}}{\Gamma(\alpha_S)} s^{\alpha_S - 1} \exp\{-s\beta_S\} \tag{6.22}$$

$$F_S(s) = 1 - \frac{\Gamma(\alpha_S, s\beta_S)}{\Gamma(\alpha_S)} \tag{6.23}$$

where $\Gamma(\alpha_S, s\beta_S)$ as before is the upper incomplete gamma function.

6.6.4 Lognormally distributed S

If S is lognormally distributed with shape parameter σ_S and scale parameter μ_S, $S \sim \ln N(\mu_s, \sigma_S^2)$, then the situation may look something like in figure 6.5, depending on the choice of parameter values. Just as in all of the other models we have illustrated two different realisations of s. If $s_1 < c$ is drawn, we will observe T_{s_1} in the competing risks case or both T_{s_1} and T_c in the semi-competing risks case. If $s_2 > c$ is drawn, we will observe the failure time T_c, not T_{s_2}.

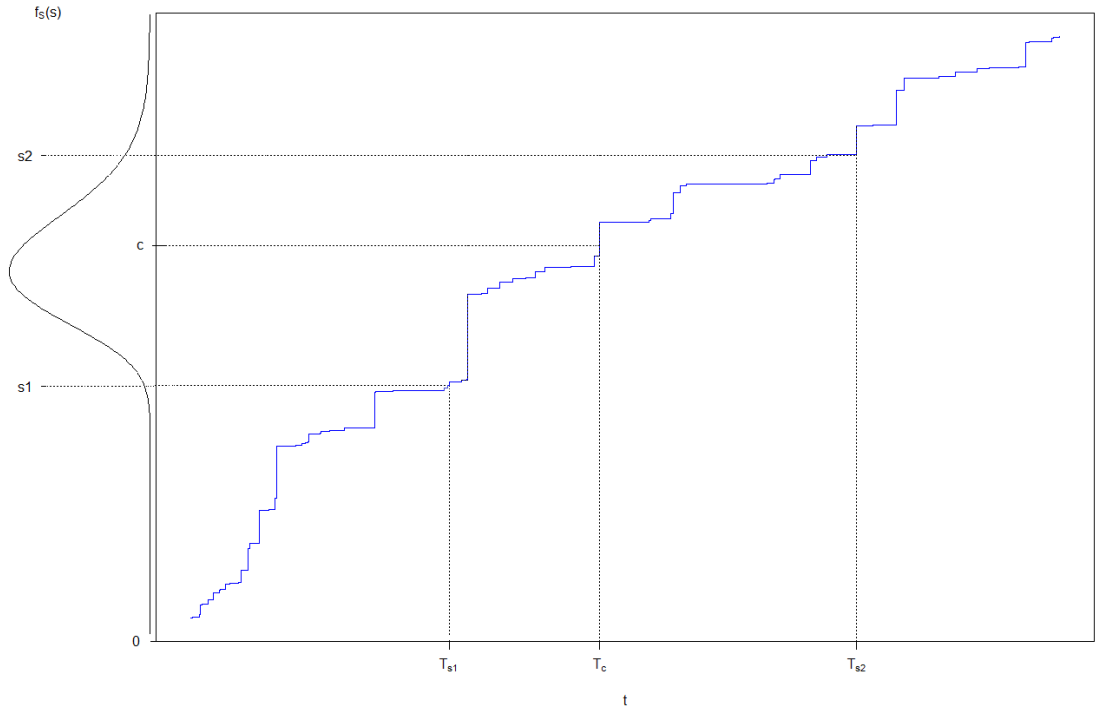

FIGURE 6.5: Illustration of the case of a gamma process $X(t)$ with a fixed level c and a lognormally distributed level S

The PDF $f_S(s)$ and CDF $F_S(s)$ in the lognormal model are given by

$$f_S(s) = \frac{1}{\sqrt{2\pi}\sigma_S s} \exp\left\{-\frac{(\ln s - \mu_S)^2}{2\sigma_S^2}\right\} \tag{6.24}$$

$$F_S(s) = \frac{1}{2} + \frac{1}{2} erf\left[\frac{\ln s - \mu_S}{\sqrt{2}\sigma_S}\right] = \Phi\left(\frac{\ln s - \mu_S}{\sigma_S}\right) \tag{6.25}$$

where $erf(\cdot)$ is the error function, and $\Phi(\cdot)$ is the cumulative distribution function of the standard normal distribution.

Chapter 7

Simulation studies

In this chapter we will conduct simulation studies with the random S models described in chapter 6. The algorithms used to simulate data are also provided in this chapter. For each dataset we generate, we will then estimate back the parameters of the models by maximum likelihood estimation and evaluate the quality of these estimates. We have chosen to only consider datasets including censored observations, because data without censoring is only a special (and simpler) version of the case with censoring.

7.1 Random S models on competing risks data

We will begin by simulating ordinary competing risks data from the random S models. Knowing the original parameter values, we may then see how good the estimates we obtain from our maximum likelihood procedure are. In addition, we will test how well the basic model from the project thesis is able to adapt to data simulated from the random S models.

7.1.1 Simulation algorithm

The simulation procedure in the models with random level S including censoring is quite similar to that of the basic model with censoring in the project thesis [35]. The difference is that instead of having a probability q that you observe a PM and $1-q$ that you observe a failure, you draw an S from the distribution $f_S(s)$ and then if $S < c$ you get a PM and if not you get a failure. The procedure to draw a sample from the random S model in the competing risks case with censoring is shown in algorithm 1.

This algorithm is implemented as the function `simRandomS`() in R, and can be found in appendix D.1.2 (for $f_S(s)$ being the uniform, exponential, gamma or lognormal distribution). To draw from the first passage time distributions $f_{T_s}(z; v(z), s)$ and $f_{T_c}(x; v(x), c)$, the same algorithm as in the project thesis was used (see appendix F.1). This is implemented as the function `simdata`() and given in appendix D.1.1.

Algorithm 1 Algorithm to sample from the model with censoring and random S

$S \sim f_S(s)$
$\tau \sim f_\tau(t)$
if $S < c$ **then**
    $Z \sim f_{T_S}(z; v(z), S)$
    if $Z < \tau$ **then**
        return Z
    else
        return τ
    end if
else
    $X \sim f_{T_c}(x; v(x), c)$
    if $X < \tau$ **then**
        return X
    else
        return τ
    end if
end if

The choice of distribution $f_\tau(\cdot)$ is arbitrary. We have selected a gamma distribution with parameters that make approximately 10 % of the observations censored.

7.1.2 Simulation results - uniform model

In this simulation, we have chosen to simulate data from a gamma process with parameter values $\alpha = 5, \beta = 1$ and $c = 5$ and uniformly distributed S on $[0, A]$, where $A = 10$. This makes $F_S(c) = 0.50$ and $\mathrm{E}[S] = c$. An illustration of how the sub-densities $f_Z^*(t)$ and $f_X^*(t)$ look with these parameter values is given in figure 7.1. These curves are found from equations (6.1) and (6.3) with the expressions from (5.5), (6.18) and (6.19) inserted. To calculate $f_Z^*(t)$ we have used the `integrate`() function in R. The first passage time distribution to the level c, $f_{T_c}(t; v(t), c)$ from equation (5.5), is plotted in the same figure for reference. As we can see, $f_X^*(t)$ is just a scaled down version of $f_{T_c}(t; v(t), c)$, more accurately $0.5 \cdot f_{T_c}(t; v(t), c)$. $f_Z^*(t)$ on the other hand has a shape that is influenced by the choice of distribution $f_S(s)$.

Using the simulation procedure from algorithm 1, we produced a dataset of $N = 1000$ observations. Of these, $m = 441$ were failure times, $n = 455$ were times to PM and $r = 104$ were censoring times. As a simple check of our simulation procedure, we have plotted the histograms of the simulated observations $x_1, ..., x_m$ and $z_1, ..., z_n$ in figure 7.2 (we briefly ignore the censored observations).

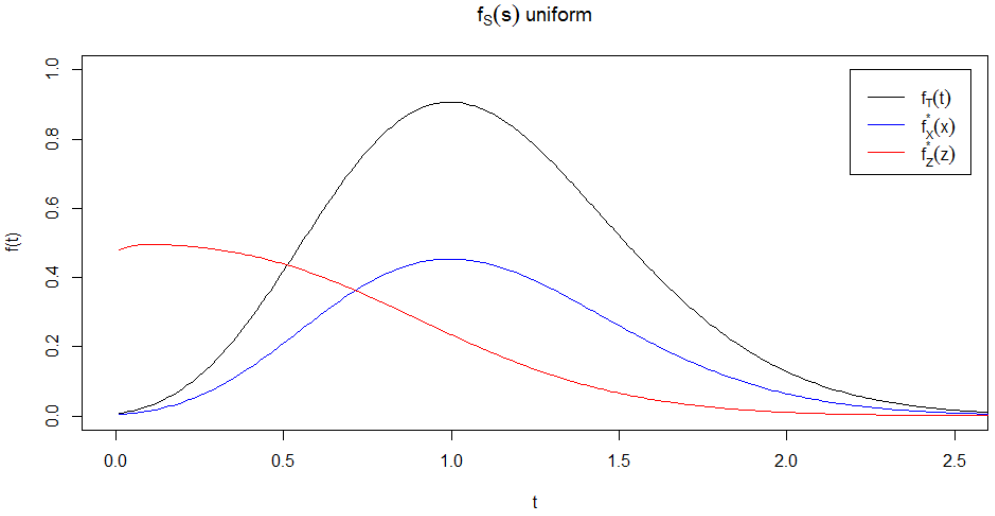

FIGURE 7.1: Sub-densities of X and Z when S is uniformly distributed on $[0, A]$ and the first passage time density to level c

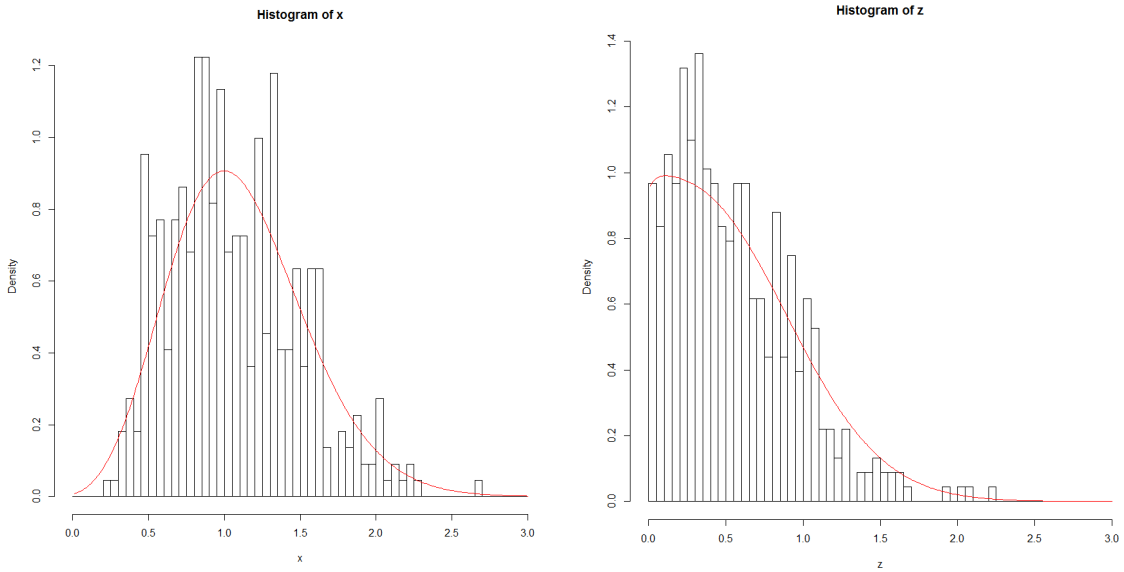

FIGURE 7.2: Histograms of the empirical distributions of X (left) and Z (right) along with the curves of the theoretical distributions in the model with uniform S

We can compare the histograms to the the curves of the corresponding theoretical distributions, which are plotted in red in the same figure. The theoretical distributions are given by the conditional sub-density functions of X and Z respectively. From section 3.2.2, we know that for X this is just

$$\tilde{f}_X(x) = \frac{f^*_X(x)}{P(X < Z)} = \frac{f_{T_c}(x; v(x), c)(1 - F_S(c))}{(1 - F_S(c))} = f_{T_c}(x; v(x), c) \tag{7.1}$$

For Z we get

$$\tilde{f}_Z(z) = \frac{f_Z^*(z)}{P(Z < X)} = \frac{\int_0^c f_{T_s}(z; v(z), s) f_S(s) ds}{F_S(c)} \tag{7.2}$$

As we can see, the histograms and the theoretical curves match fairly well.

In order to fit our gamma process model to the data, we first need to check whether $\hat{\tilde{S}}_Z(t) < \hat{\tilde{S}}_X(t)$ for all t. Therefore, we have plotted the non-parametric estimates of $\tilde{S}_Z(t)$ (thick line) and $\tilde{S}_X(t)$ (thin line) together to the left of figure 7.3. These estimates were found by using equation (3.4). The inequality is clearly fulfilled for most values of t (for t-values greater than ≈ 2.3 it is impossible to tell the curves apart). As we can recall, $\hat{\tilde{S}}_Z(t) < \hat{\tilde{S}}_X(t)$ means that we can fit a random signs censoring model to these data.

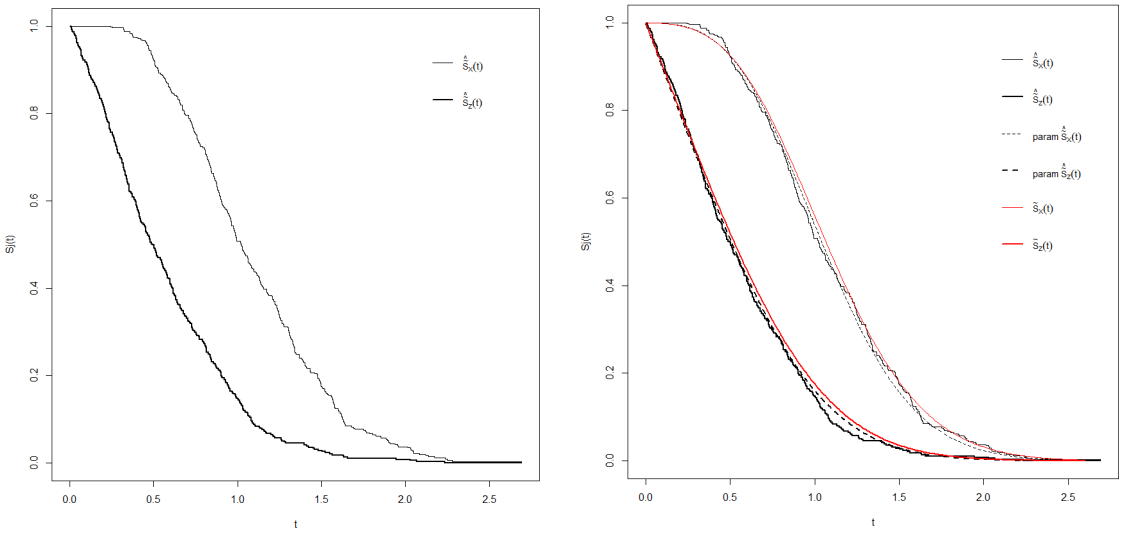

FIGURE 7.3: Parametric and non-parametric estimates of the conditional sub-survival functions for the data generated from the uniform model

To estimate the parameters of the uniform model, we used the `condSurv`() function, which again calls `optim`() to maximize the log-likelihood function in (6.6). `condSurv`() is included in appendix D.2.1. From this function, we get maximum likelihood estimates for all of the parameters, as well as standard deviations calculated from the Hessian matrix and limits for the 95% standard positive confidence interval. The results are displayed in table 7.1, while the complete output from R is given in appendix E.1.1.

From table 7.1 we can see that the differences between the correct and the estimated parameter values are relatively small. The parameters α, c and A all seem to have been a little over-estimated. Still, all of the 95% standard positive confidence

TABLE 7.1: Maximum likelihood estimates of the parameters in the model with uniformly distributed S. In addition, the correct values, standard deviations from the Hessian matrix and 95% standard positive confidence intervals are included

Par.	Correct value	Est.	St. deviation	Lower bound	Upper bound
α	5	5.4524	0.6608	4.2995	6.9143
β	1	0.9923	0.0445	0.9087	1.0835
c	5	5.3411	0.7079	4.1192	6.9254
A	10	10.8534	1.4963	8.2835	14.2207

intervals include the correct parameter values with good margin. β has the smallest estimated standard deviation, as we also saw in the basic model in the project thesis. The standard deviations for α and c are considerably larger, but the standard deviation of A is by far the largest. It is about twice as large as the standard deviations for α and c, which seems logical as the parameter estimate itself is also about twice as large.

Now, to further evaluate the fit of the model, we have plotted the non-parametrically estimated conditional sub-survival curves (solid black lines) together with the parametrically estimated curves (dashed black lines) and the curves of the true conditional sub-survival functions (solid red lines) to the right in figure 7.3. The parametric estimates are found from the expressions described in section 6.4.1. From figure 7.3 we can see that they are quite close to the true and non-parametric ones. This indicates that the estimates are good. For small t there is almost no observable difference in the true and parametrically estimated curves. For higher values of t, the difference seems to be slightly larger for the estimates of $\tilde{S}_X(t)$ than for the estimates of $\tilde{S}_Z(t)$.

Comparison to the fit of the basic model

It would be interesting to check how well the basic model from the project thesis (with fixed level s) fits the data simulated from the uniform S model. It is reasonable to expect that the basic model is not well suited for these data, as the distribution in (6.17) is very different from the uniform distribution. We can compare the estimated values for α, β and c with the true values as before, but the estimate of s will be compared to $\mathrm{E}[S|S < c]$ and the estimate of q will be compared to $F_S(c)$. The calculation of $\mathrm{E}[S|S < c]$ is shown in appendix B.1.1. The maximum likelihood estimation using the log-likelihood function for the basic model with censoring from equation (F.1) was done with the function `condSurv()`. It yielded the results shown in table 7.2. The table contains parameter estimates, standard deviations, and lower

and upper bounds for the 95% standard positive confidence intervals estimated from the Hessian matrix. As before, the complete output from R is given in appendix E.1.1.

TABLE 7.2: Maximum likelihood estimates of the parameters in the basic model for the data simulated with uniform S. In addition: correct values, standard deviations from the Hessian matrix and 95% standard positive confidence intervals

Par.	Correct value	Est.	St. deviation	Lower bound	Upper bound
α	5	3.1295	0.5938	2.1576	4.5393
β	1	1.0570	0.0898	0.8949	1.2485
c	5	2.9403	0.6053	1.9640	4.4918
s	$(\mathrm{E}[S\|S < c]=)2.5$	1.2397	0.3695	0.6913	2.2235
q	$(F_S(c)=)0.5$	0.4917	0.0166	0.4602	0.5252

Comparing the estimated parameter values to the true or expected ones in table 7.2, it is clear that they are quite different from each other. The estimated parameter values for α, c and s are quite far from the true or expected values, and they are all underestimated. For these parameters, the true or expected values are not included in the 95% standard positive confidence intervals. Still, the estimate of β is quite close to its true value, and the estimate of q or $F_S(c)$ is quite accurate.

We can further evaluate the estimated conditional sub-survival curves based on the parameter estimates in table 7.2. These curves are shown in figure 7.4 with dashed black lines, along with the true conditional sub-survival curves from the uniform S model in solid red lines and the non-parametrically estimated curves in solid black. These parametrically estimated curves do not match the non-parametric curves as well as the estimates from the uniform S model in figure 7.3 did. They deviate from both the true and the non-parametric curves for both the smallest and the largest t-values.

One can also compare the maximum log-likelihood values of the two models. The basic model resulted in a value of -1087.523, while the uniform S model had a value of -1064.135. Hence the uniform S model fits the data much better than the basic model, as we expected.

7.1.3 Simulation results - exponential model

In the following simulation study, we have chosen parameter values $\alpha = 5, \beta = 1, c = 7$ and $\lambda_S = 0.1$. This makes $f_S(s)$ equal to the expression in (6.20) and $F_S(s)$ as in (6.21). With our chosen parameter values, $F_S(c) \approx 0.50$ and $\mathrm{E}[S] =$

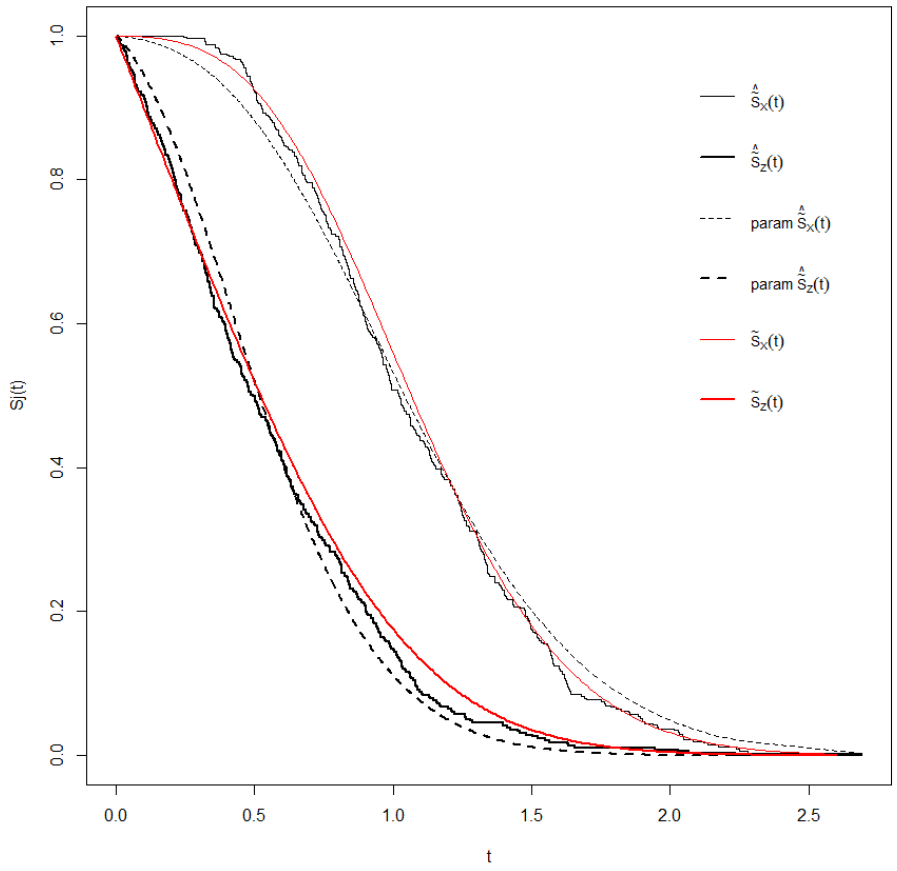

FIGURE 7.4: Parametric and non-parametric estimates of the conditional sub-survival functions for the data generated from the uniform model when the parametric estimates are from the basic model

10. A plot of the sub-density functions $f_X^*(t)$ and $f_Z^*(t)$ together with the first passage time distribution to the level c from equation (5.5) is given in figure 7.5. As we can see, also in this case $f_X^*(t) = 0.5 f_{T_c}(t; v(t), c)$, while the shape of $f_Z^*(t)$ is strongly influenced by the exponential distribution of S. To calculate $f_X^*(t)$ we used expression (6.1), and for $f_Z^*(t)$ we have used the `integrate`() function in R on the expression in equation (6.3).

Using the `simRandomS`() function from appendix D.1.2, we simulated $N = m + n + r = 1000$ observations. In the resulting dataset we got $m = 455$ failure times, $n = 455$ times to PM and $r = 90$ censored observations. The histograms of Z and X are displayed in figure 7.6. In addition, the curves of the theoretical distributions from equations (7.1) and (7.2) are plotted (ignoring the censored observations). They fit the histograms reasonably well, and gives us reason to believe that the data follow the intended distribution.

Next, we check if Cooke's condition for a random signs censoring model holds

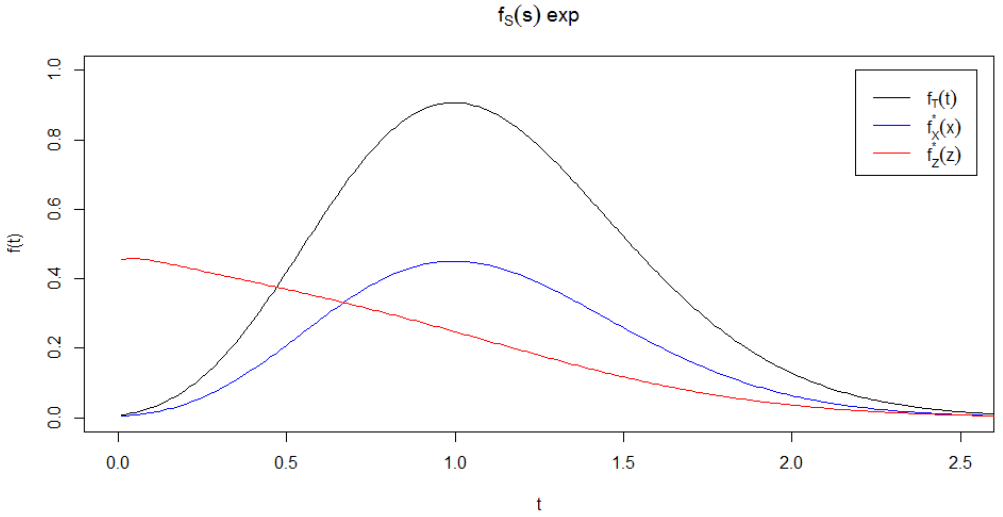

FIGURE 7.5: First passage time density to the level c and sub-densities of Z and X when S is exponentially distributed

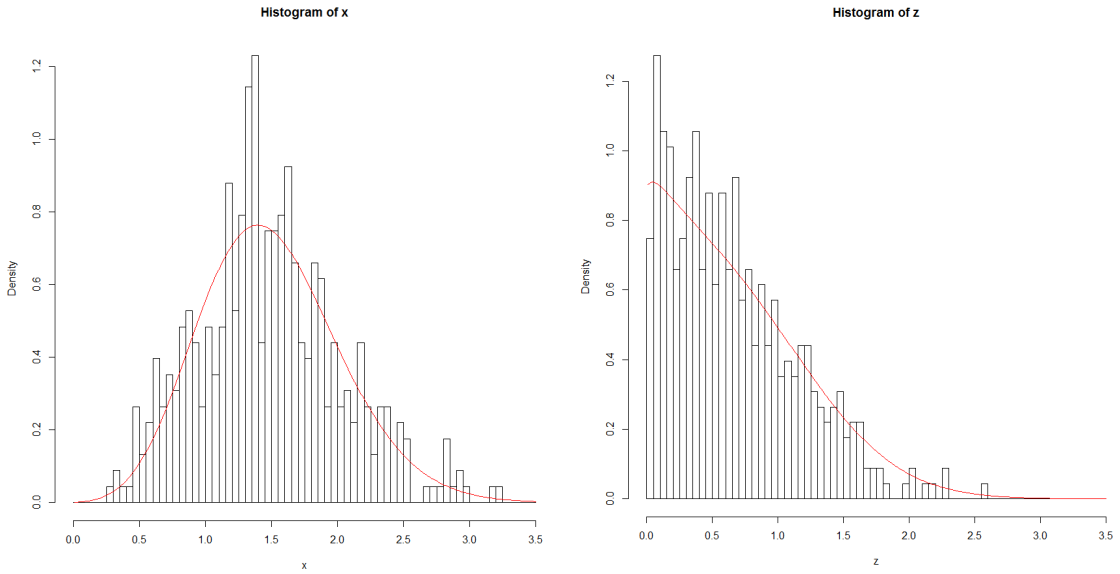

FIGURE 7.6: Histograms of the empirical distributions of X (left) and Z (right) along with the curves of the theoretical distributions in the model with exponential S

for this dataset by plotting the conditional sub-survival curves estimated non-parametrically. The result is presented to the left of figure 7.7. Like in the uniform case, these estimates are made by using equation (3.4). As we can see, $\hat{\bar{S}}_Z(t)$ (thick line) lies below $\hat{\bar{S}}_X(t)$ (thin line) for all t, so the condition holds.

Thus, we can fit our exponential random S model to the data. By using the function `condSurv`() in appendix D.2.1 to maximize the log-likelihood function from equation (6.6) we get the results shown in table 7.3. There, the estimated parameter values

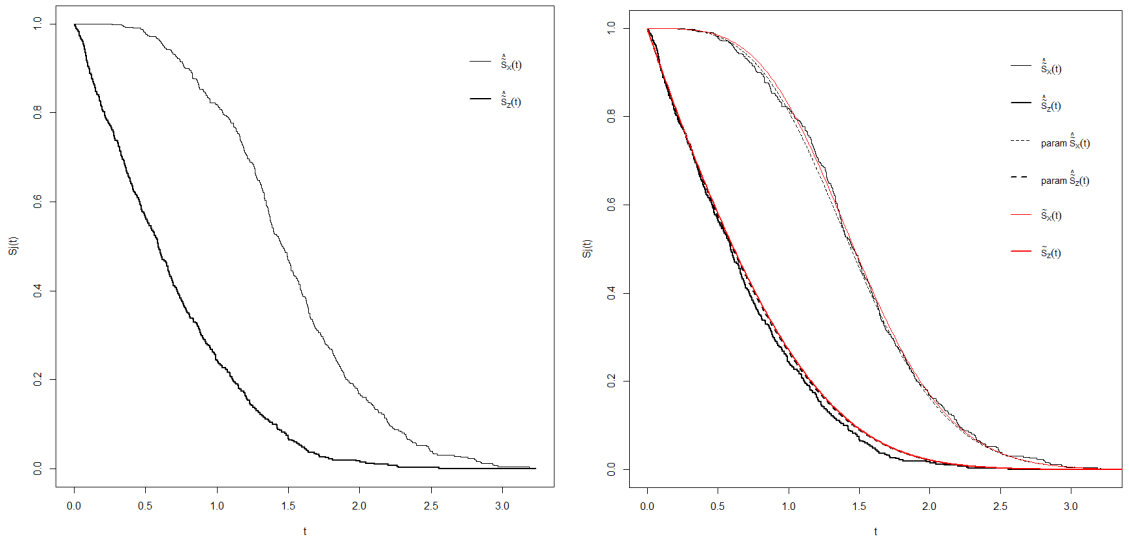

FIGURE 7.7: Parametric and non-parametric estimates of the conditional sub-survival functions for the data generated from the exponential model

are given together with the their true values, the estimated standard deviations and the limits of the 95% standard positive confidence intervals calculated from the Hessian matrix. The complete output from R is included in appendix E.1.2.

TABLE 7.3: Maximum likelihood estimates of the parameters in the model with exponential S. In addition, the correct values, standard deviations from the Hessian matrix and 95% standard positive confidence intervals are included

Par.	Correct value	Est.	St. deviation	Lower bound	Upper bound
α	5	4.8895	0.5747	3.8834	6.1563
β	1	0.9943	0.0401	0.9188	1.0761
c	7	6.7068	0.7728	5.3510	8.4063
λ_S	0.1	0.0993	0.0129	0.0770	0.1280

Considering the parameter estimates provided in table 7.3, the estimation procedure seems to work well also in the exponential model. The estimates are all close to their true values, and well within their respective 95% confidence intervals. As in the uniform model, the standard deviations of α and c are considerably larger than that of β. The standard deviation of λ_S is also very small, but then so is the parameter value itself. It is approximately 10% of the size of the parameter value.

Using the estimated values from table 7.3, we estimate the parametric conditional sub-survival curves by the method explained in section 6.4.1. We plot them in the same figure as the non-parametric and true curves. This is shown to the right of

figure 7.7. As we can see, all of the curves lie closely together and there is almost no difference between the parametrically estimated curves and the true ones.

Comparison to the fit of the basic model

It would be interesting to see how well the basic model would fit to these data simulated from the exponential model. We expect the fit to be quite poor, as an exponential distribution is very different from having the point probabilities from (6.17). Maximizing the log-likelihood function in (F.1) using the function `condSurv`(), we got the results shown in table 7.4. The complete output from R is given in appendix E.1.2 and the calculation of $\mathrm{E}[S|S < c]$ is included in appendix B.1.2.

TABLE 7.4: Maximum likelihood estimates of the parameters in the basic model for the data simulated with exponential S. In addition: correct values, standard deviations from the Hessian matrix and 95% standard positive confidence intervals

Par.	Correct value	Est.	St. dev.	Lower bound	Upper bound
α	5	1.9437	0.3177	1.4109	2.6778
β	1	1.1759	0.0730	1.0411	1.3281
c	7	2.6866	0.4214	1.9756	3.6535
s	$(\mathrm{E}[S\|S < c] =)3.0950$	0.8087	0.2088	0.4875	1.3415
q	$(P(S < c) =)$ 0.5	0.4838	0.0164	0.4527	0.5170

From table 7.4, we can see that the parameters α, c and s are all significantly underestimated compared to the original or expected values. β on the other hand is overestimated. Neither of these four parameters have their true or expected value inside the estimated 95% standard positive confidence intervals. The 95% confidence interval for q is the only one that contains its expected value.

Inserting the parameter estimates from table 7.4 into the functions in section 6.4.1, we can plot the parametric conditional sub-survival curves and compare them to the ones made with the true parameter values and the non-parametric curves. This is done in figure 7.8. Here, it is obvious that the estimates from the basic model do not fit the data very well. The parametrically estimated curves seem to have a different shape than the true and non-parametric ones, both for Z and for X.

The poor fit of the basic model is further confirmed by the maximum log-likelihood values. For the basic model the maximum value became -1314.095 whereas in the exponential model it was -1275.883. In conclusion, the exponential model fits much better, as we expected.

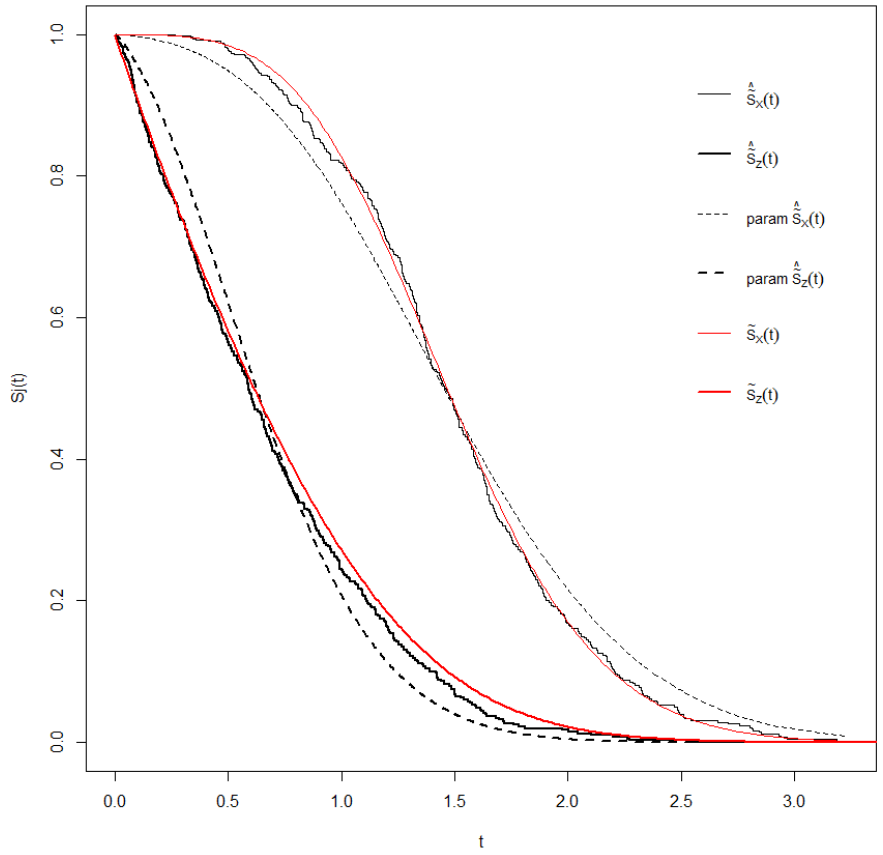

FIGURE 7.8: Parametric and non-parametric estimates of the conditional sub-survival functions for the data generated from the exponential model when the parametric estimates are from the basic model

7.1.4 Simulation results - gamma model

In this case we have chosen parameter values $\alpha = 5, \beta = 1, c = 7, \alpha_S = \frac{49}{4}$ and $\beta_S = \frac{7}{4}$. Hence, we use the expressions for $f_S(s)$ and $F_S(s)$ from (6.22) and (6.23). The chosen parameter values make $F_S(c) \approx 0.54$ and $\mathrm{E}[S] = c$. A plot of the sub-density functions $f_X^*(t)$ and $f_Z^*(t)$ along with $f_{T_c}(t; v(t), c)$ for reference is given in figure 7.9. $f_X^*(t)$ is equal to $0.54 f_{T_c}(t; v(t), c)$, while $f_Z^*(t)$ is influenced by the $\mathrm{Ga}(\alpha_s, \beta_s)$-distribution of S. As we can see, it has a completely different shape than for instance when S was uniformly distributed in figure 7.1. To calculate $f_Z^*(t)$ we have used the `integrate()` function in R on the expression in (6.3). $f_X^*(t)$ is found from equation (6.1).

After simulating a dataset of $N = m + n + r = 1000$ observations we got $m = 443$ failure times, $n = 460$ times to PM and $r = 97$ censoring times. Histograms of the observed x'es and z's are given in figure 7.10 along with curves of the theoretical distributions in red. The theoretical distributions are as before found from the

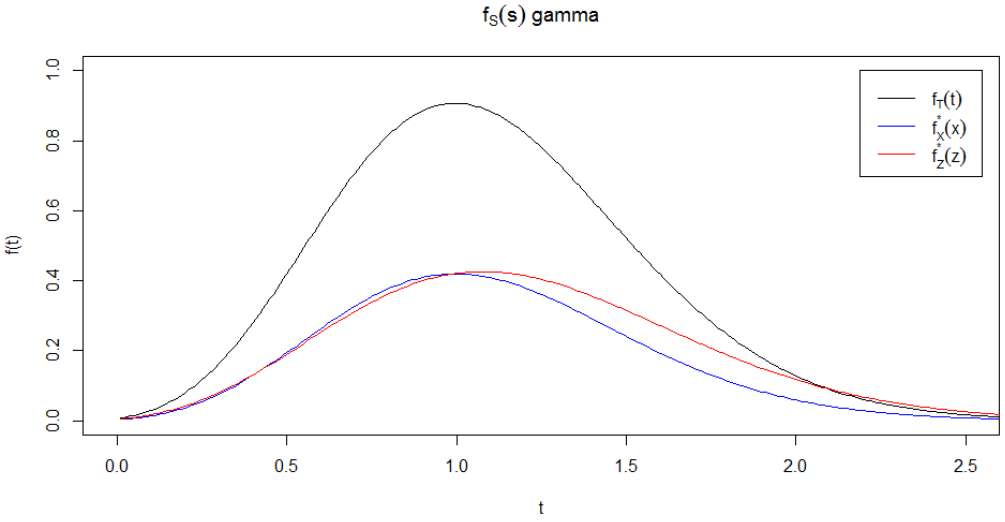

FIGURE 7.9: First passage time density to the level c and sub-densities for Z and X when S is gamma distributed

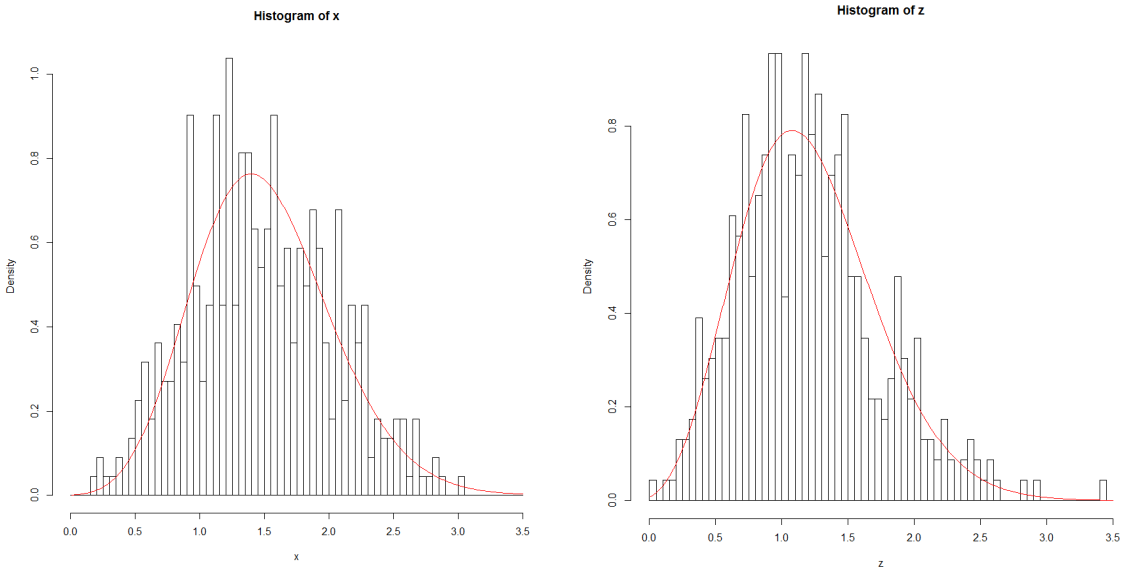

FIGURE 7.10: Histograms of the empirical distributions of X (left) and Z (right) along with the curves of the theoretical distributions in the model with gamma distributed S

expressions in (7.1) and (7.2). In both cases they seem to match the histograms pretty well.

Also in this model we make a plot of the non-parametrically estimated conditional sub-survival curves. These are made from using equation (3.4). The plot is given to the left of figure 7.11. From this we can confirm that $\hat{\bar{S}}_Z(t) < \hat{\bar{S}}_X(t)$ for all t, and so our gamma process model can be fitted to the data.

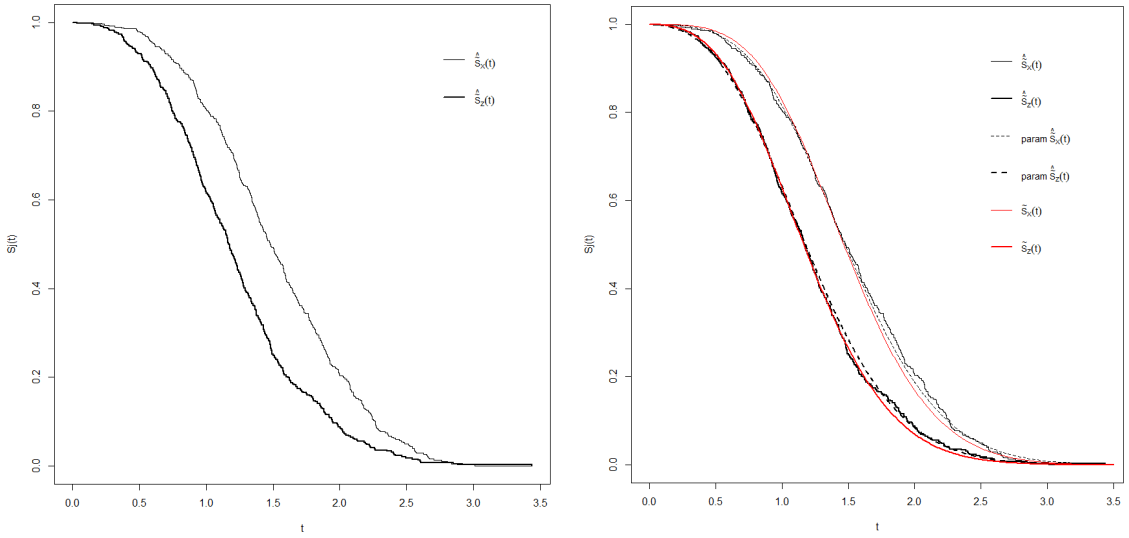

FIGURE 7.11: Parametric and non-parametric estimates of the conditional sub-survival functions for the data generated from the gamma model

The resulting parameter estimates from the maximization of the log-likelihood function in (6.6) by the `condSurv()` function is given in table 7.5. The table also displays the estimated standard deviations and the upper and lower limits of the 95 % standard positive confidence intervals. The output from R is provided in appendix E.1.3.

TABLE 7.5: Maximum likelihood estimates of the parameters in the model with gamma S. In addition, the correct values, standard deviations from the Hessian matrix and 95% standard positive confidence intervals are included.

Par.	Correct value	Est.	St. deviation	Lower bound	Upper bound
α	5	3.8414	1.0978	2.1939	6.7259
β	1	1.0534	0.1230	0.8380	1.3243
c	7	5.4587	1.3639	3.3450	8.9080
α_S	$\frac{49}{4}$=12.25	10.7625	3.5498	5.6384	20.5434
β_S	$\frac{7}{4}$=1.75	1.9160	0.5213	1.1241	3.2657

From table 7.5 we can see that the parameter estimates are quite good. All of the true parameter values are within the estimated confidence intervals. We can however note that the uncertainty in the estimates are larger than in the uniform and the exponential models. This is natural considering that the gamma model requires estimation of an additional parameter. Again the estimate of β has a fairly low standard deviation, while it is larger for α and c. The standard deviations for α_S and β_S are of similar size (relative to the size of the parameter estimates themselves).

To the right of figure 7.11, the parametrically estimated conditional sub-survival curves from section 6.4.1 are plotted as black dashed lines together with the curves made with the true parameter values in solid red. The curves match each other fairly well. There is only a small difference between the true and the estimated curves, and it actually looks like the estimated curves fit the non-parametric ones even better than the true curves.

Comparison to the fit of the basic model:
We follow the same procedure as with the previous models and check how well the basic model fits the data simulated from the gamma model. With a relatively narrow gamma distributed S, we expect the basic model to fit better than to the data with uniform or exponential S. The results from the maximum likelihood estimation made with `condSurv`() are shown in table 7.6. The complete output from R can be found in appendix E.1.3. $\mathrm{E}[S|S < c]$ has been calculated in appendix B.1.3.

TABLE 7.6: Maximum likelihood estimates of the parameters in the basic model from the data simulated with gamma S. In addition: the correct values, standard deviations from the Hessian matrix and 95% standard positive confidence intervals

Par.	Correct value	Est.	St. dev.	Lower bound	Upper bound
α	5	3.5104	0.9537	2.6511	5.9786
β	1	1.0582	0.1158	0.8511	1.3157
c	7	4.9842	1.1823	3.1310	7.9344
s	$(\mathrm{E}[S\|S < c]=)$ 5.5270	3.8441	1.0322	2.2711	6.5066
q	$(P(S < c) =)0.54$	0.5026	0.0166	0.4711	0.5363

The estimates in table 7.6 show the same tendency as we have seen earlier when we have fitted the basic model to data from the random S model: α, c and s all seem to be underestimated if we compare them to the original or expected parameter values. The estimate of β on the other hand, is larger than its true value. This time however, the true or expected values of all of the parameters except q are within the estimated 95% standard positive confidence intervals.

We can further plot the parametrically estimated conditional sub-survival curves and compare them to the true and non-parametric curves. This is done in figure 7.12. The parametric curves are found in the same manner as before, namely by inserting the estimates from table 7.6 into the expressions from 6.4.1. Here, as with the fit of the gamma model to the data, it actually seems like the parametric estimates (in black dashed lines) are a little closer to the non-parametric curves (in black solid lines) than the true ones (in red solid lines). However, if we compare this

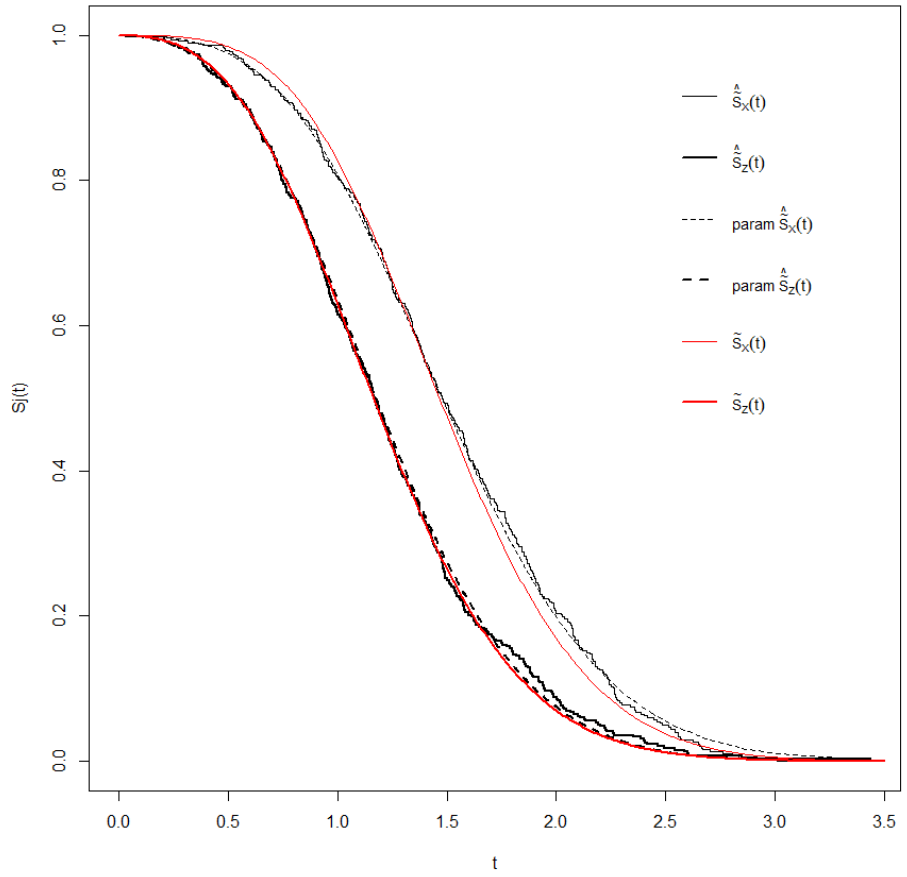

FIGURE 7.12: Parametric and non-parametric estimates of the conditional sub-survival functions for the data generated from the gamma model when the parametric estimates are from the basic model

figure with the one on the right in figure 7.11, the fit of the basic model to the non-parametric curves still looks slightly worse than the fit of the gamma model. But overall, the fit seems to be very good. As we predicted, the basic model performs better on the data from the gamma model than on the data from the uniform or exponential model.

Finally, we compare the maximum log-likelihood values of the two models. The maximum log likelihood value with the basic model was -1378.151, while with the gamma model it was -1377.681. Thus there is only a very small difference in favour of the model with gamma distributed S, which confirms what we saw on figure 7.12.

7.1.5 Simulation results - lognormal model

For the simulation study of the lognormal model, we have chosen parameter values $\alpha = 5, \beta = 1, c = 7, \mu_S = 2$ and $\sigma_S = 0.25$. We will in the following use the expressions for $f_S(s)$ and $F_S(s)$ from (6.24) and (6.25). This makes $F_S(c) \approx 0.41$ and $\mathrm{E}[S] \approx 7.6$.

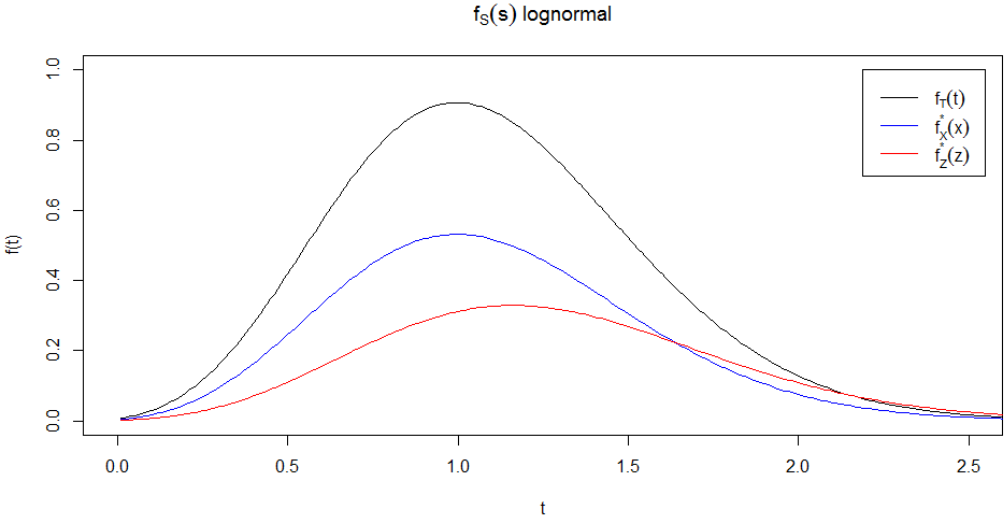

FIGURE 7.13: First passage time density to the level c and sub-densities for Z and X when S is lognormally distributed

In figure 7.13 the sub-densities $f_Z^*(t)$ and $f_X^*(t)$ are plotted along with the first passage time distribution to the level c, $f_{T_c}(t; v(t), c)$. To calculate $f_X^*(t)$ we have as before used equation (6.1) and for $f_Z^*(t)$ we have used the `integrate`() function in R on equation (6.3). $f_Z^*(t)$ looks somewhat similar in shape to what it did in the gamma model, which is logical as the gamma distribution and the lognormal distribution have similar shapes for the parameter values we have chosen.

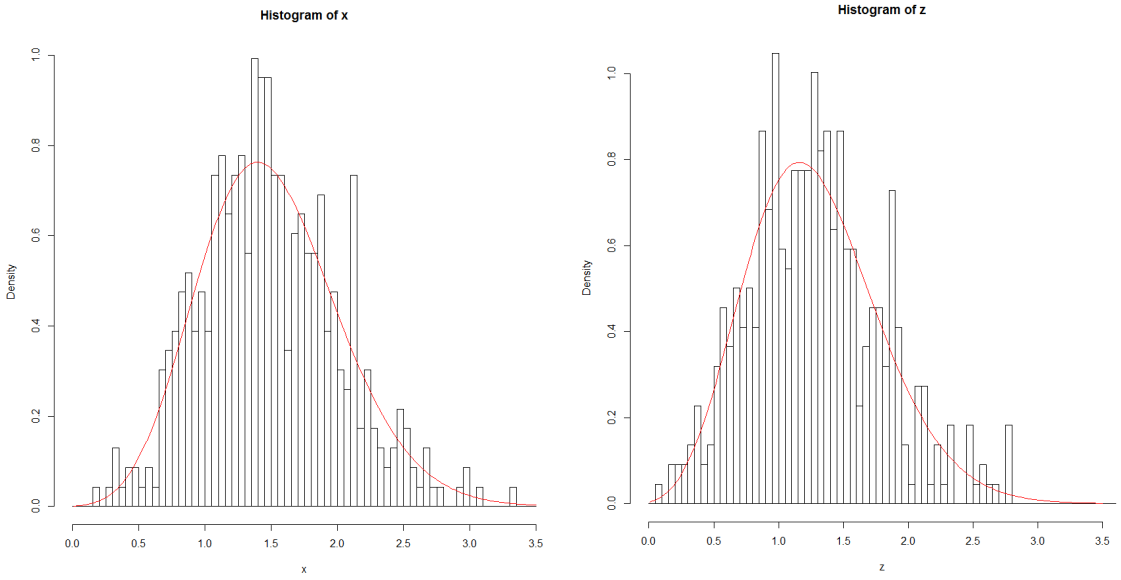

FIGURE 7.14: Histograms of the empirical distributions of X (left) and Z (right) along with the curves of the theoretical distributions in the model with lognormal S

We used algorithm 1 to simulate a dataset consisting of $N = m + n + r = 1000$ observations. We ended up with $m = 463$ failure times, $n = 439$ times to PM and $r = 98$ censoring times. The histograms of $x_1, ...x_m$ and $z_1, ..., z_n$ are displayed in figure 7.14. Ignoring the censored observations, they seem to match their theoretical distributions made from (7.1) and (7.2) plotted in red quite well. Thus, there is no reason to doubt our simulation procedure.

We want to check whether $\tilde{S}_Z(t) < \tilde{S}_X(t)$ holds also in the lognormal model. The non-parametrical estimates made with equation (3.4) are plotted to the left of figure 7.15, and we can see that the inequality is fulfilled for all t. We can thereby proceed to fit the lognormal model to the data.

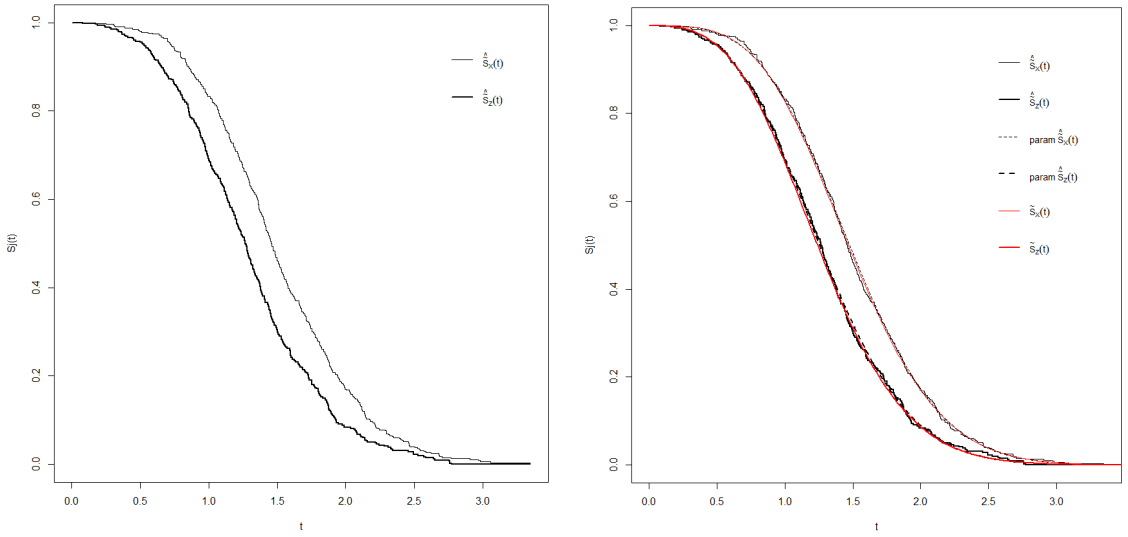

FIGURE 7.15: Parametric and non-parametric estimates of the conditional sub-survival functions for the data generated from the lognormal model

The parameters of the model were estimated by maximizing the log-likelihood function from (6.6) by the function `condSurv`() described in appendix D.2.1. The resulting estimates are displayed in table 7.7. There, the corresponding standard deviations and the 95% standard positive confidence limits are given too. The results as they were provided by R is included in appendix E.1.4.

The estimation procedure seems to work well also in the lognormal model. The parameter estimates in table 7.7 are all fairly close to their true values, and the true values are all contained in their respective 95% confidence intervals. The standard deviations are of approximately the same size as in the gamma model. Also for the parameters of the lognormal distribution, the size of the estimated standard deviations seems to be reasonable (about 10-20% of the size of the parameter values).

TABLE 7.7: Maximum likelihood estimates of the parameters in the model with lognormal S. In addition, the correct values, standard deviations from the Hessian matrix and 95% standard positive confidence intervals are included.

Par.	Correct value	Est.	St. deviation	Lower bound	Upper bound
α	5	4.0681	1.0852	2.4117	6.8621
β	1	1.0851	0.1169	0.8786	1.3401
c	7	5.8791	1.3635	3.7316	9.2625
μ_S	2	1.7815	0.2313	1.3812	2.2979
σ_S	0.25	0.2433	0.0485	0.1646	0.3596

To the right of figure 7.15 the parameter estimates from table 7.7 have been used in the expressions from section 6.4.1 to plot parametric estimates of the conditional sub-survival curves (black dashed lines). In addition, the lines of the true conditional sub-survival functions are plotted (red lines). The curves are all very close to each other and hard to tell apart, indicating a very good fit.

Comparison to the fit of the basic model

Also for this model it is interesting to check how well the basic model would fit the data compared to the lognormal model. The fit should not be too bad, considering that the lognormal distribution we chose is relatively centred around a specific value. The resulting estimates from maximizing the log-likelihood function in (F.1) is given in table 7.8. The table also contains standard deviations as well as upper and lower bounds for the 95% standard positive confidence intervals estimated from the Hessian matrix. The calculation of $\mathrm{E}[S|S < c]$ is shown in appendix B.1.4. The complete output from R is given in appendix E.1.4.

TABLE 7.8: Maximum likelihood estimates of the parameters in the basic model from the data simulated with lognormal S. In addition: correct values, standard deviations from the Hessian matrix and 95% standard positive confidence intervals

Par.	Correct value	Est.	St. dev.	Lower bound	Upper bound
α	5	3.8198	0.9946	2.2931	6.3632
β	1	1.0916	0.1150	0.8880	1.3419
c	7	5.5070	1.2440	3.5370	8.5744
s	$(\mathrm{E}[S\|S < c] =)$ 5.8964	4.5920	1.1295	2.8355	7.4365
q	$(P(S < c) =)$0.41	0.4835	0.0166	0.4520	0.5172

As in all of the previous models, the parameters α, c and s are underestimated. However, like in the gamma model these parameters have their true or expected

value inside the estimated 95% confidence interval. This is true also for the parameters β and q. Thus the fit of the basic model actually seems to be rather good.

We can also evaluate the quality of the model fit of the basic model by comparing the parametrically estimated conditional sub-survival curves with the non-parametric and true ones. These are all plotted together in figure 7.16. As before, the parametric curves are made from plugging the parameter estimates of table 7.8 into the expressions from section 6.4.1. In the figure, we can see that the curves lie very closely together, and it is hard to tell them apart or say that the basic model is worse than the lognormal.

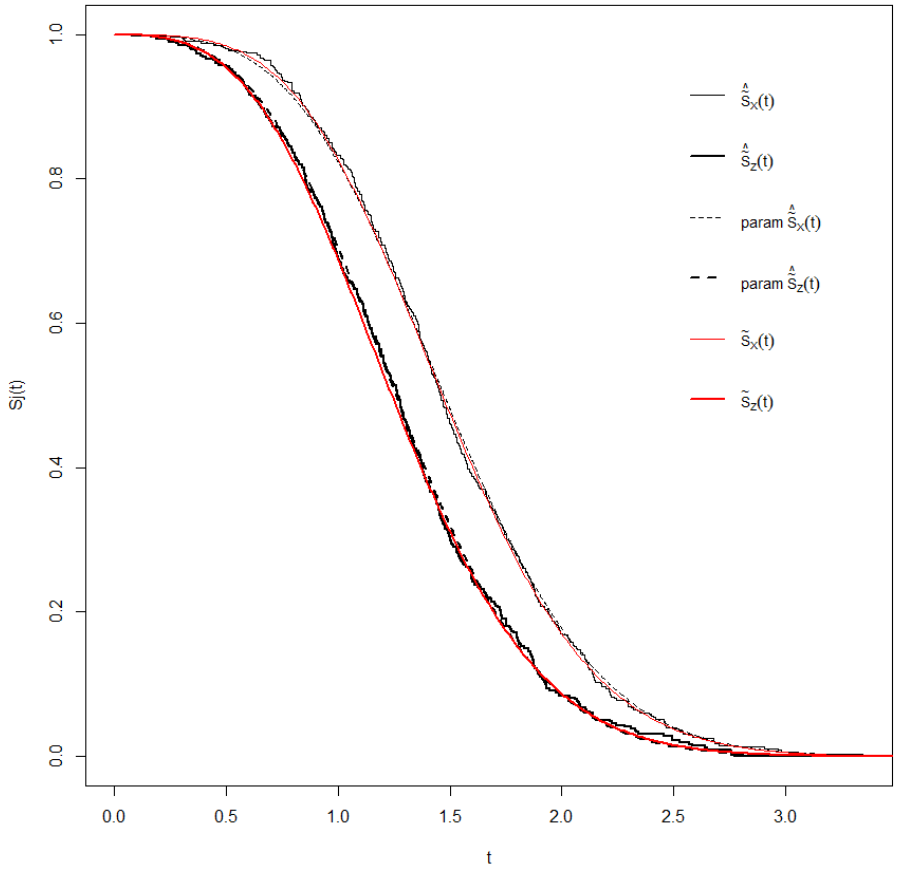

FIGURE 7.16: Parametric and non-parametric estimates of the conditional sub-survival functions for the data generated from the lognormal model when the parametric estimates are from the basic model

Lastly, we compare the maximum log-likelihood values of the two models. With the basic model we got a value of -1322.205, while with the lognormal model it was -1321.492. Hence the models are almost equally good. The lognormal model is only marginally better.

7.1.6 Summary of simulation study - competing risks

From the simulation studies in sections 7.1.2, 7.1.3, 7.1.4 and 7.1.5, we have seen that the parameter estimation worked fine for all of the four random S models. The numerical integration discussed in section 6.3 does not seem to have caused us considerable problems. The true parameter values were always well inside the estimated 95% standard positive confidence intervals. We should thus be able to trust the parameter estimates we obtain by maximizing the log-likelihood functions. We saw that the uncertainty in the estimates were somewhat larger in the lognormal and gamma models than in the uniform and exponential models. This is probably due to the extra parameter.

In all of our simulation cases the model with the random S distribution corresponding to the distribution that the data was simulated from was better than that of constant s. This was especially true for the uniform and exponential models. Thus, if the maintenance policy in fact is to perform PM randomly, and it is equally likely to happen at any time, then you should use a model that takes this into account. For most real applications however, PM is probably most likely to be performed at a time when the component has reached a specific age, just before it is most likely to fail. It should be mentioned that the difference between the fit of the gamma or lognormal models and the fit of the basic model to data simulated from the gamma or lognormal models respectively, was very small. This is probably due to that we had chosen gamma and lognormal distributions with relatively small variance. In figure 7.17 the chosen distributions of S in the four cases are plotted together.

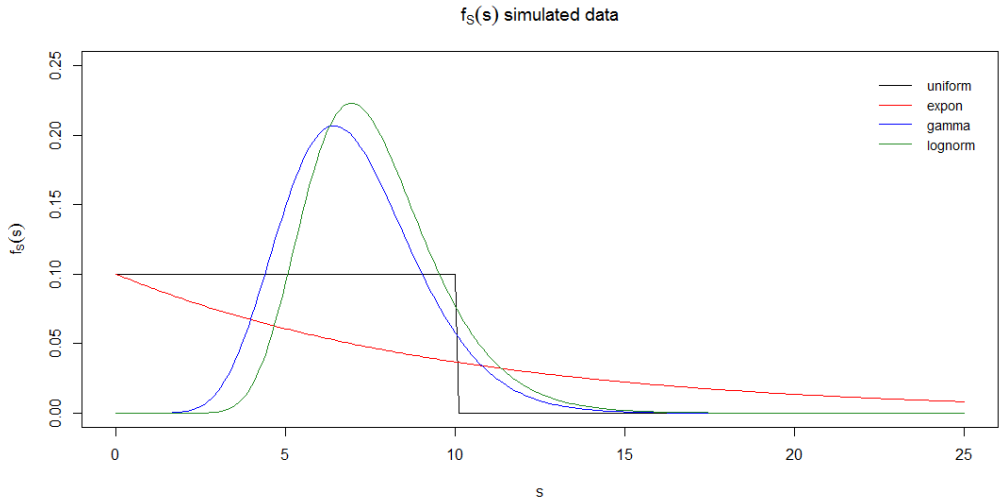

FIGURE 7.17: $f_S(s)$ for the four random S models used in the simulation studies for competing risks data

As an experiment we try to fit the lognormal model to the data simulated from the uniform model. The results from this estimation are provided in table 7.9. There, both parameter estimates, standard deviations estimated from the Hessian matrix and limits for the 95% standard positive confidence intervals are given. The complete output from R is shown in appendix E.1.1.

TABLE 7.9: Maximum likelihood estimates of the parameters in the model with lognormal S. In addition, the correct values, standard deviations from the Hessian matrix and 95% standard positive confidence intervals are included.

Par.	Correct value	Est.	St. deviation	Lower bound	Upper bound
α	5	11.1281	2.3158	7.4008	16.7327
β	1	0.6964	0.0726	0.5676	0.8543
c	5	11.0919	2.3326	7.3451	16.7501
μ_S	-	2.4282	0.2125	2.0455	2.8825
σ_S	-	0.8031	0.0955	0.6361	1.0139

As we can see, the parameter estimates in table 7.9 turned out to be quite different from the original values in the uniform model. This time α and c are significantly larger than in the uniform model, while β is lower. To see how well this estimated lognormal model fits to the data, we plot the parametrically estimated conditional sub-survival curves and compare them to the true curves from the uniform model. Once more we use the expressions provided in section 6.4.1 to make the parametric curves. The resulting plot is shown in figure 7.18.

The fit of the lognormal model to the uniform data actually seems to be very good. It is difficult to evaluate which of the models provides the closest fit to the non-parametric estimates. The black dashed lines of the lognormal model in figure 7.18 or the black dashed lines of the uniform model to the right in figure 7.3. For $\tilde{S}_X(t)$ it looks like the fit of the lognormal model is the best, but for $\tilde{S}_Z(t)$ the difference is not that clear.

Moreover, the uniform model resulted in a maximum log-likelihood value of -1064.135 while the lognormal model gave -1059.603, which is larger. This shows the great flexibility of the lognormal model. It can both attain shapes that are relatively wide and flat, and shapes that are quite narrow. The same is true for the gamma distribution.

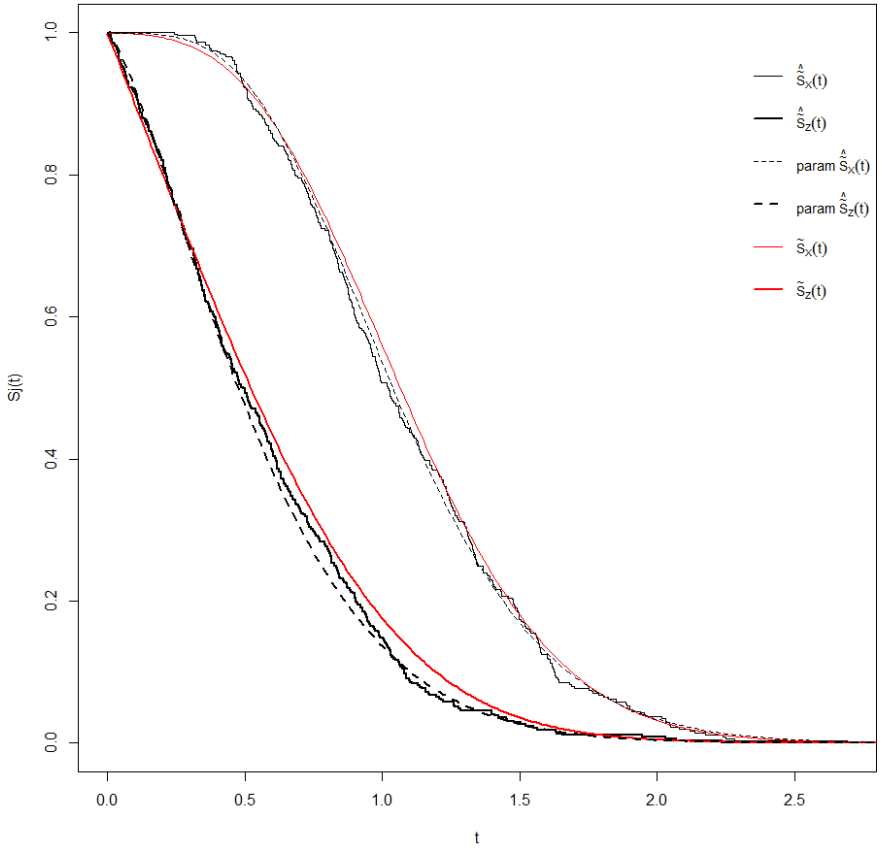

FIGURE 7.18: Parametric and non-parametric estimates of the conditional sub-survival functions for the data generated from the uniform model when the parametric estimates are from the lognormal model

7.2 Random S models on semi-competing risks data

We will now simulate semi-competing risks data from the four random S models. We will then estimate back the parameter values and see how well they match with the original values.

7.2.1 Simulation algorithm

When censoring is included there are four possible types of observations, as described in chapter 4. In each of the four cases we get the following set of observations:

1. times to PM, $z_1, ..., z_n$ and corresponding failure times $x_{z1}, ..., x_{zn}$
2. failure times $x_1, ..., x_m$
3. times to PM $z_{o1}, ..., z_{ow}$ and corresponding censoring times $\tau_{o1}, ..., \tau_{ow}$
4. censoring times $\tau_1, ..., \tau_r$

Algorithm 2 Sample from the semi-competing risks model with random S including censoring

$S \sim f_S(s)$
$\tau \sim f_\tau(t)$
if $S < c$ **then**
    $Z \sim f_{T_S}(z; v(z), S)$
    if $Z < \tau$ **then**
        $[X_Z - Z] \sim f_{T_{c-s}}(x_z - z; v(x_z) - v(z), c - S)$
        $X_Z = Z + [X_Z - Z]$
        if $X_Z < \tau$ **then**
            return Z
            return X_Z
        else
            return $Z_o = Z$
            return $\tau_o = \tau$
        end if
    else
        return τ
    end if
else
    $X \sim f_{T_c}(x; v(x), c)$
    if $X < \tau$ **then**
        return X
    else
        return τ
    end if
end if

A procedure to draw a sample from the random S model with censoring in the semi-competing risks case is shown in algorithm 2. The algorithm is implemented as the function `simSemi`() which can be found in appendix D.1.3. The `simSemi`() function calls `simdata`() to draw from the first passage time distributions $f_{T_c}(x; v(x), c)$ and $f_{T_s}(z; v(z), s)$, and `simdata2`() to draw from the first passage time distribution given $z = t_1$ and starting from level $S = s$, i.e. $f_{T_{c-s}}(t - t_1; v(t) - v(t_1), c - s)$. `simdata`() and `simdata2`() are given in appendix D.1.1. Also this time the distribution for τ is chosen arbitrarily to make a suitable amount of censored observations. We have used a gamma distribution with parameter values that make $w + r$ equal approximately 10% of the total $(m + n + w + r)$. Again, we perform simulation studies for both uniform, exponential, gamma and lognormal S.

7.2.2 Simulation results - uniform model

For this simulation study, we chose $\alpha = 5, \beta = 1, c = 5$ and $A = 10$ as parameter values, just like in the competing risks simulation study. The expressions for $f_S(s)$ and $F_S(s)$ are again given by (6.18) and (6.19). We used algorithm 2 to simulate $N = m + n + w + r = 1000$ observations. In the resulting dataset we had $m = 446$ observations of only the terminal event, $n = 435$ observations of both the non-terminal and the terminal event, $w = 33$ observations of the non-terminal event and a censoring time, and $r = 86$ observations of only the censoring time. As we can recall, the log-likelihood function is now given by equation (6.11). The parameter estimates from the maximum likelihood procedure in the function `estSemi()` from appendix D.2.3, as well as standard deviations and lower and upper bounds for the 95% standard positive confidence intervals calculated from the Hessian matrix, are given in table 7.10. The complete output from R is provided in appendix E.2.1.

TABLE 7.10: Maximum likelihood estimates of the parameters in the model with uniformly distributed S. In addition, the correct values, standard deviations from the Hessian matrix and 95% standard positive confidence intervals are included

Par.	Correct value	Est.	St. deviation	Lower bound	Upper bound
α	5	4.3596	0.4279	3.5966	5.2845
β	1	1.0441	0.0381	0.9720	1.1216
c	5	4.1476	0.4507	3.3520	5.1320
A	10	8.3029	0.9495	6.6357	10.3890

From these results we can see that the estimation procedure seems to work reasonably well also for semi-competing risks data when S is uniformly distributed. All of the estimated parameter values are relatively close to the true ones, though not as close as the estimates in the competing risks case in section 7.1.2. The true parameter values are well within the estimated 95% standard positive confidence intervals. The estimated standard deviations are a little smaller compared to what they were for ordinary competing risks.

We have plotted the parametrically estimated and true marginal survival functions $S_Z(t)$ and $S_X(t)$ together to the left of figure 7.19. As explained in section 6.5.1, the parametric estimates are found from inserting the parameter estimates from table 7.10 into equations (6.14) and (5.3), respectively. We have also plotted the estimated and true marginal hazard functions $\lambda_Z(t)$ and $\lambda_X(t)$ together to the right in the same figure. These were made from the expressions in (6.15) and (6.16). In both of these plots there is some difference between the true curves (in red) and

the estimated ones (in black). Still, they are generally quite close to each other and follow the same shape. It seems like the estimated and true curves of the hazard functions deviate more from each other with growing t. This is probably due to the fact that there are very few observations that are greater than $t \approx 2$. We can also notice that the curves are slightly further apart for Z than for X.

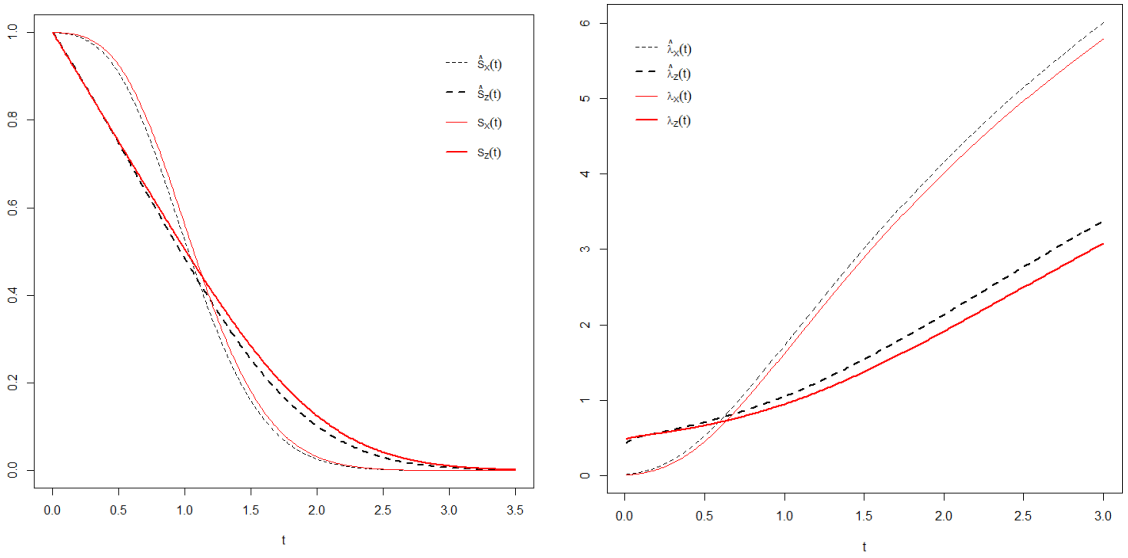

FIGURE 7.19: True and estimated marginal survival functions $S_Z(t)$ and $S_X(t)$ (left) and hazard functions $\lambda_Z(t)$ and $\lambda_X(t)$ (right) with uniform S

In addition, we have plotted the estimated crude quantities described in section 4.4, i.e. the sub-distribution function and cumulative sub-hazard rate of Z, in figures 7.20 and 7.21. The parametrically estimated sub-distribution function for Z, $\hat{F}_Z^*(t)$ is found from the expression in (6.12), while the non-parametric estimate is given by (4.4). As we can see from figure 7.20, the curves all match each other very well. The true curve (red) is almost indistinguishable from the parametrically estimated one (black, dashed). The fit to the non-parametric curve (black, solid) seems to be very good.

When it comes to the cumulative sub-hazard rate, the parametric estimate was found from inserting the parameter estimates into the expression in (6.13) and integrating, while the non-parametric estimate was made by using equation (4.3). Also these curves lie quite closely together, but there is a more distinct difference between the true curve (red) and the parametrically estimated curve (black, dashed). Also here we can notice that the difference between the true and the parametrically estimated curves increases with the value of t. As mentioned earlier, this might be due to that there are few observations for large t. Still, the curves both follow the non-parametric curve quite closely.

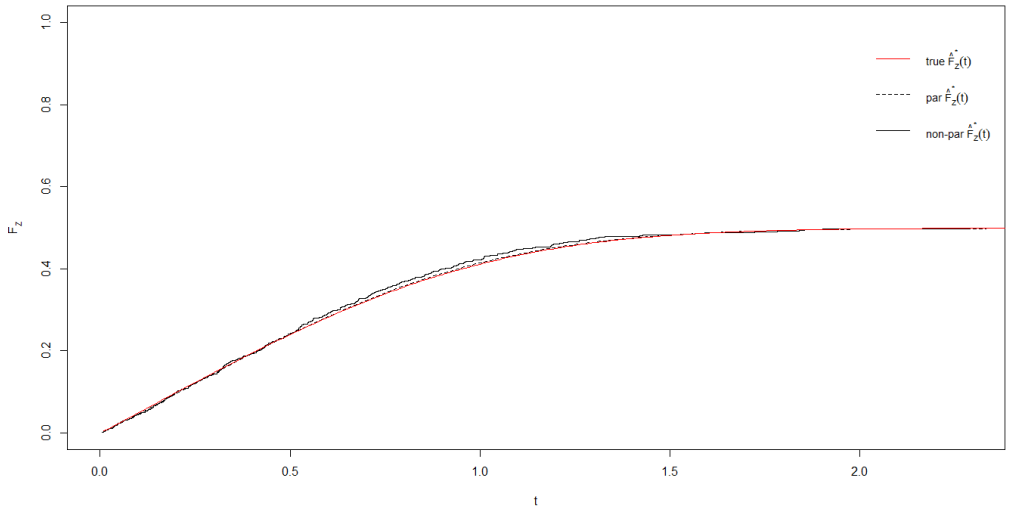

FIGURE 7.20: Parametric and non-parametric estimates of the sub-distribution function for Z, $F^*_Z(t)$, with uniform S

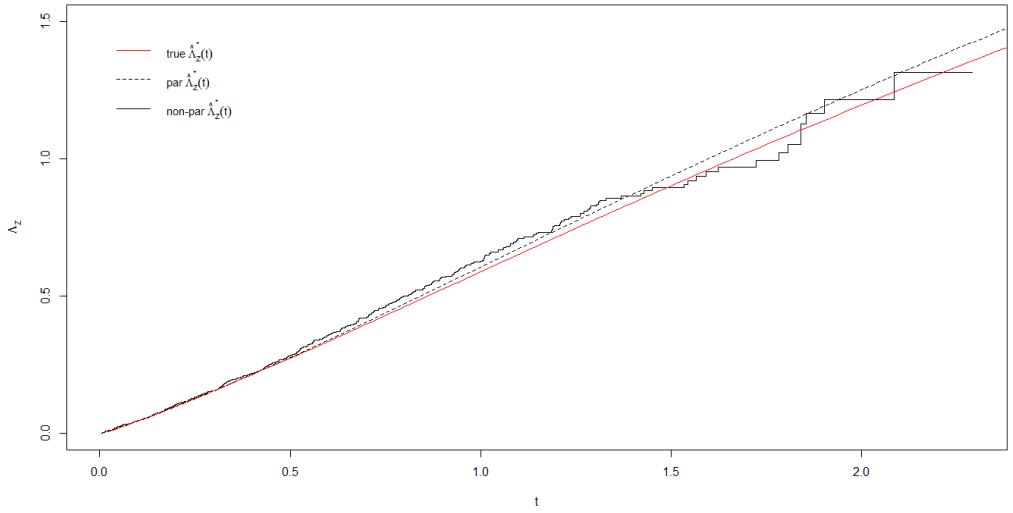

FIGURE 7.21: Parametric and non-parametric estimates of the cumulative sub-hazard rate for Z, $\Lambda^*_Z(t)$ with uniform S

7.2.3 Simulation results - exponential model

As in the competing risks case, we chose $\alpha = 5, \beta = 1, c = 7$ and $\lambda_S = 0.1$ as parameter values. $f_S(s)$ and $F_S(s)$ are then given by equations (6.20) and (6.21). In the resulting dataset from our simulation we had $m = 442$ observations of only the terminal event, $n = 444$ observations of both the non-terminal and the terminal event, $w = 32$ observations of the non-terminal event and a censoring time, and $r = 82$ observations of only a censoring time. To estimate the parameters of the exponential model we used the function `estSemi()` from appendix D.2.3 which again calls the `optim()` function to maximize the log-likelihood function from equation (6.11). Parameter estimates, along with standard deviations as well as lower and

upper bounds for the 95% standard positive confidence intervals calculated from the Hessian matrix are given in table 7.11. The complete output from R can be found in appendix E.2.2.

TABLE 7.11: Maximum likelihood estimates of the parameters in the model with exponential S. In addition, the correct values, standard deviations from the Hessian matrix and 95% standard positive confidence intervals are included

Par.	Correct value	Est.	St. deviation	Lower bound	Upper bound
α	5	5.2065	0.5829	4.1808	6.4840
β	1	0.9677	0.0395	0.8932	1.0483
c	7	7.1727	0.7547	5.8361	8.8154
λ_S	0.10	0.0975	0.0119	0.0768	0.1237

From these results we can see that also with exponential S the estimation procedure seems to work reasonably well for semi-competing risks data. In this case the estimated standard deviations seem have approximately the same size as they did in the competing risks setting in section 7.1.3. The estimated parameter values are quite close to the true ones, and the true values are well within the estimated 95% standard positive confidence intervals.

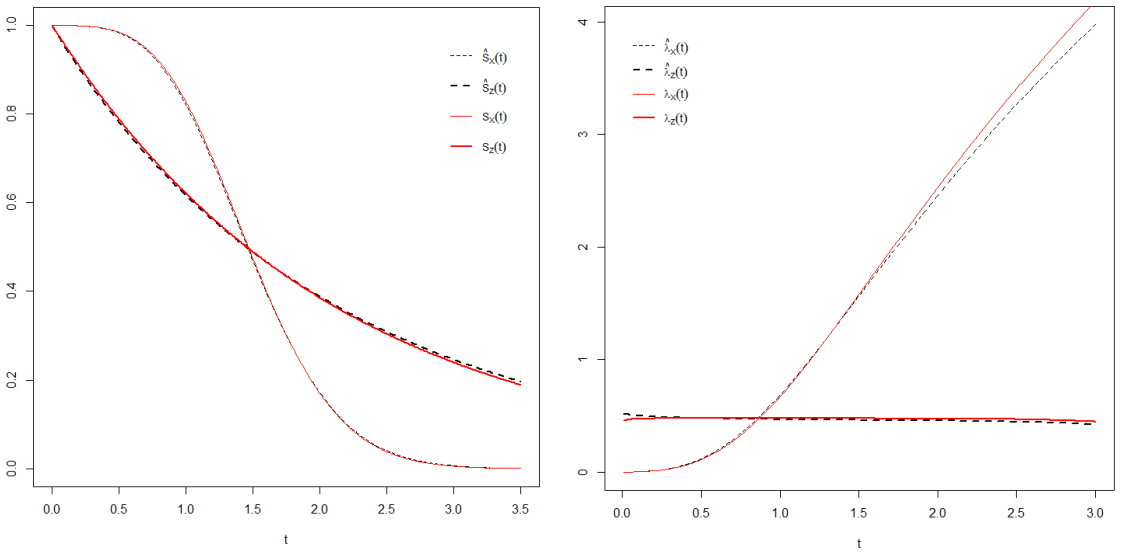

FIGURE 7.22: True and estimated marginal survival functions $S_Z(t)$ and $S_X(t)$ (left) and marginal hazard functions $\lambda_Z(t)$ and $\lambda_X(t)$ (right) with exponential S

We have furthermore plotted the estimated and true marginal survival functions $S_Z(t)$ and $S_X(t)$ together to the left in figure 7.22. These were found from equations (6.14) and (5.3). We have also plotted the estimated and true marginal hazard functions functions $\lambda_Z(t)$ and $\lambda_X(t)$ from equations (6.15) and (6.16) together to

the right in the same figure. As we can see, the difference between the true and the estimated curves, in red and black respectively, is very small. It is smaller than what it was in the uniform model, which is logical considering how close the estimated parameter values are to the original ones in table 7.11. Also here, the difference seems to grow with larger t-values for the hazard functions. It is additionally interesting to notice how the shapes of $\hat{S}_Z(t)$ and $\hat{\lambda}_Z(t)$ change with the distribution of S. They are different from what they were in the uniform model.

We continue by plotting the estimated crude quantities. In figure 7.23, the sub-distribution function for Z is shown. The parametric estimate of $F_Z^*(t)$ is again found from the expression in (6.12), while the non-parametric estimate is given by (4.4). Just as in the uniform model, the curves all match each other very well. Again, the true curve (red) is almost indistinguishable from the parametrically estimated one (black, dashed). Moreover, the fit to the non-parametric curve (black, solid) seems to be very good.

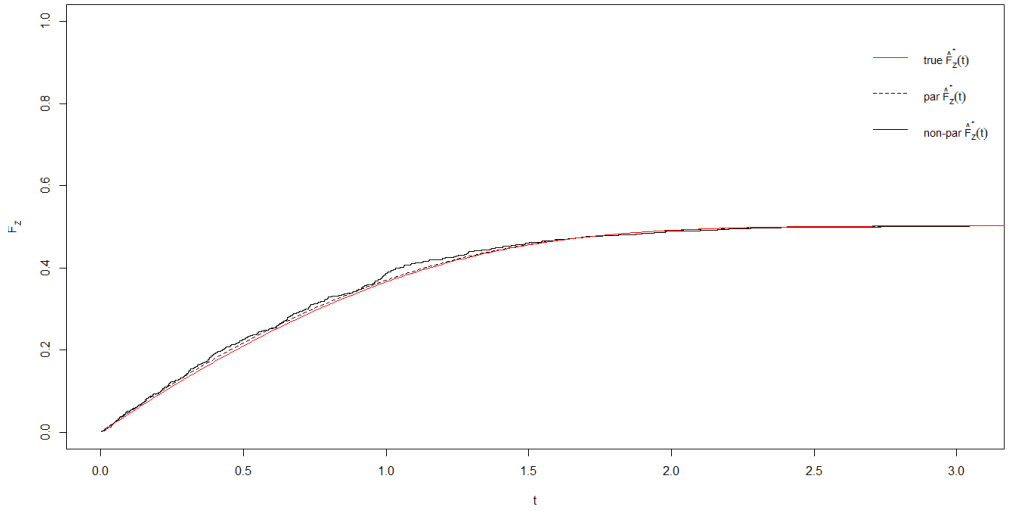

FIGURE 7.23: Parametric and non-parametric estimates of the sub-distribution function for Z, $F_Z^*(t)$ with exponential S

In figure 7.24 the cumulative sub-hazard rate of Z is illustrated. The parametric estimate was once again found by using the expression in equation (6.13), while the non-parametric estimate was made with equation (4.3). As we can see from the figure, the curves all follow each other very closely. The difference between the true curve (red) and the parametrically estimated curve (black, dashed) is not as large as in the uniform model.

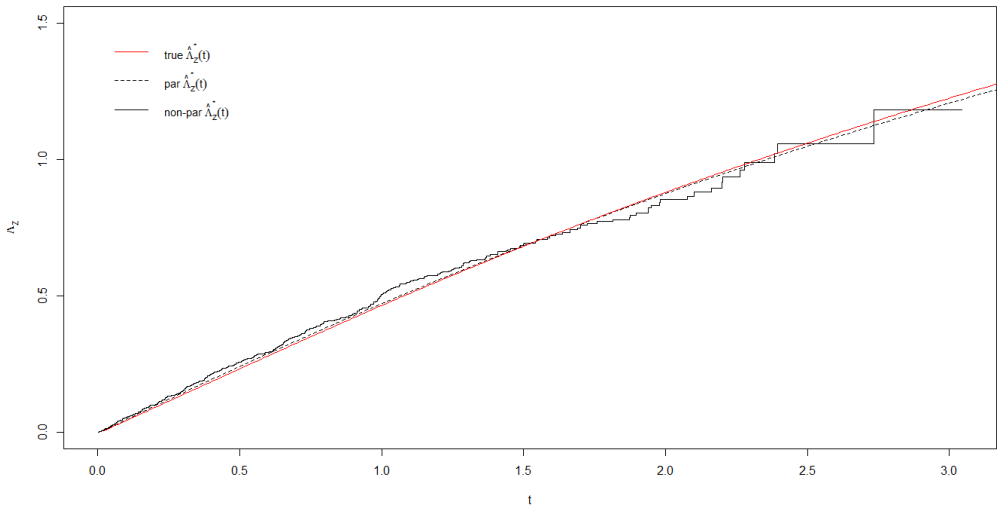

FIGURE 7.24: Parametric and non-parametric estimates of the cumulative sub-hazard rate for Z, $\Lambda_Z^*(t)$ with exponential S

7.2.4 Simulation results - gamma model

For the gamma model we kept using $\alpha = 5, \beta = 1, c = 7, \alpha_S = \frac{49}{4}$ and $\beta_S = \frac{7}{4}$ as parameter values. We used algorithm 2 to simulate a dataset of $N = 1000$ observations. In the resulting data we had $m = 423$ observations of only the time to the terminal event, $n = 463$ observations of both times to the non-terminal event and times to the terminal event, $w = 24$ observations of times to the non-terminal event followed by a censoring time, and $r = 90$ observations of only the censoring time. If we maximize the log-likelihood function from (6.11), with (6.22) and (6.23) inserted for $f_S(s)$ and $F_S(s)$ we get the results displayed in table 7.12. There, parameter estimates, standard deviations and lower and upper bounds for the 95% standard positive confidence intervals calculated from the Hessian matrix are given. For the complete output from the `estSemi`()-function, see appendix E.2.3.

TABLE 7.12: Maximum likelihood estimates of the parameters in the model with gamma distributed S. In addition, the correct values, standard deviations from the Hessian matrix and 95% standard positive confidence intervals are included

Par.	Correct value	Est.	St. deviation	Lower bound	Upper bound
α	5	5.9776	0.9370	4.3964	8.1275
β	1	0.9631	0.0652	0.8435	1.0998
c	7	8.3849	1.1589	6.3951	10.9938
α_S	$\frac{49}{4} = 12.25$	13.0545	1.8600	9.8737	17.2602
β_S	$\frac{7}{4} = 1.75$	1.5483	0.2026	1.1981	2.0010

These results also support that the estimation procedure seems to work well. The estimated parameter values are close to the true ones, in fact closer than the estimates from the ordinary competing risks case in section 7.1.4 were. The estimated standard deviations are in general smaller than than what they were there for competing risks, but larger than what they were in the uniform and exponential models. Anyhow, the true parameter values are contained within the estimated 95% standard positive confidence intervals.

Also in this case we have plotted the estimated and true marginal survival functions $S_Z(t)$ and $S_X(t)$ using the expressions in (6.14) and (5.3). These are shown together to the left in figure 7.25. We have also plotted the estimated and true marginal hazard functions functions $\lambda_Z(t)$ and $\lambda_X(t)$ together to the right in the same figure. These were found using equations (6.15) and (6.16). We can see that in both of these plots the true curves (in red) are very close to the estimated ones (in black). As we have noticed in the previous models, the true and estimated curves of the marginal hazard rates seem to deviate from each other when t grows large, however only for X this time.

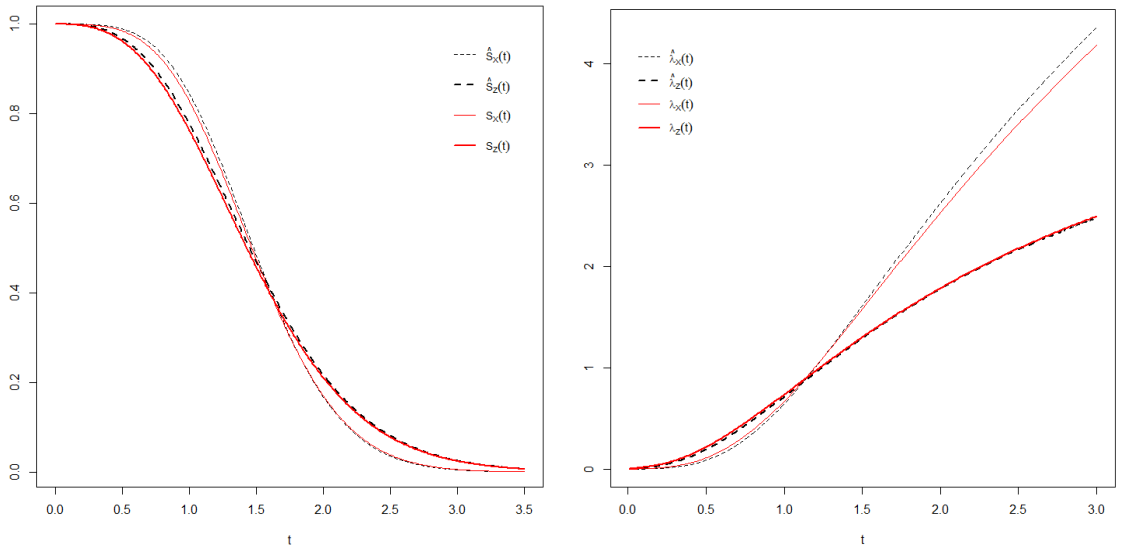

FIGURE 7.25: True and estimated marginal survival functions $S_Z(t)$ and $S_X(t)$ (left) and hazard functions $\lambda_Z(t)$ and $\lambda_X(t)$ (right) with gamma distributed S

Following the same procedure as in the other models, we have also plotted the estimated crude quantities. The true, parametric and non-parametric sub-distribution functions for Z are shown together in figure 7.26. The parametric estimate of $F_Z^*(t)$ is found from the expression in (6.12), while the non-parametric estimate is given by (4.4). Not surprisingly, the curves all match each other very well also for the gamma model.

The true, parametric and non-parametric cumulative sub-hazard functions for Z are shown together in figure 7.27. As before, the parametric estimate was found from inserting the estimated parameter values into the expression in (6.13) and integrating, while the non-parametric estimate was made by using equation (4.3). Also these curves are very similar to each other.

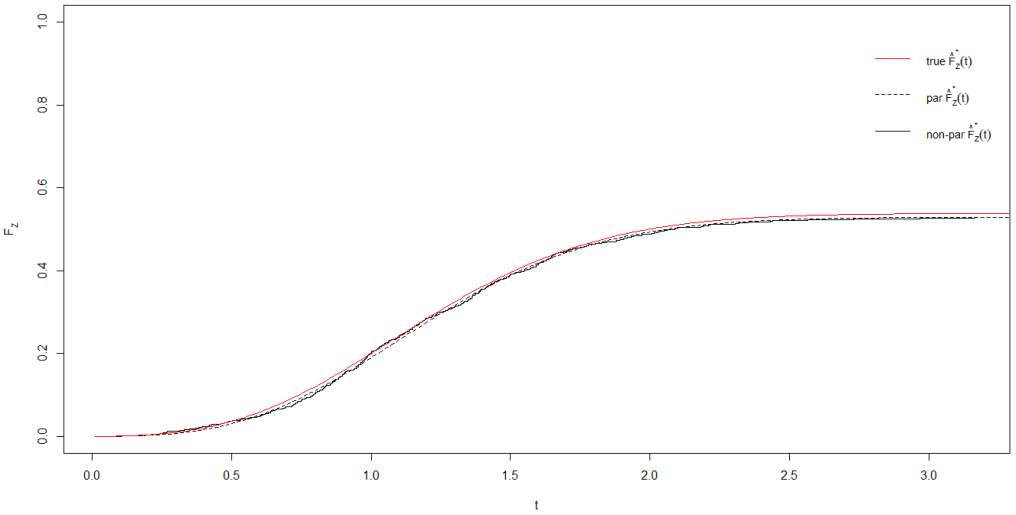

FIGURE 7.26: Parametric and non-parametric estimates of the sub-distribution function for Z, $F_Z^*(t)$ with gamma distributed S

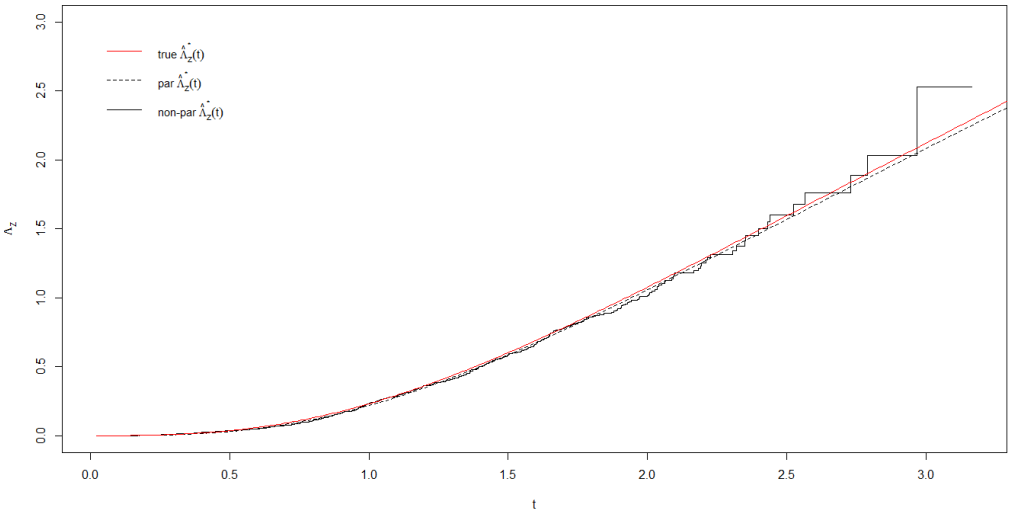

FIGURE 7.27: Parametric and non-parametric estimates of the cumulative sub-hazard rate for Z, $\Lambda_Z^*(t)$ with gamma distributed S

7.2.5 Simulation results - lognormal model

For the lognormal model we have $\alpha = 5, \beta = 1, c = 7, \mu_S = 2$ and $\sigma_S = 0.25$ as parameter values. The PDF and CDF of the distribution of S are then given as in equations (6.24) and (6.25). From the simulation we got $m = 494$ observations of only times to the terminal event, $n = 385$ observations of both times to the

non-terminal and to the terminal event, $w = 16$ observations of times to the non-terminal event and a censoring time, and $r = 105$ observations of only the censoring time. The resulting estimates from the `estSemi()` function, along with standard deviations and lower and upper bounds for the 95% standard positive confidence intervals calculated from the Hessian matrix are given in table 7.13. The complete output from R is given in appendix E.2.4.

TABLE 7.13: Maximum likelihood estimates of the parameters in the model with lognormal S. In addition, the correct values, standard deviations from the Hessian matrix and 95% standard positive confidence intervals are included

Par.	Correct value	Est.	St. deviation	Lower bound	Upper bound
α	5	4.6975	0.5861	3.6784	5.9990
β	1	1.0109	0.0522	0.9135	1.1186
c	7	6.7394	0.7506	5.4176	8.3836
μ_S	2	1.9440	0.1113	1.7376	2.1749
σ_S	0.25	0.2564	0.0195	0.2209	0.2976

We can see that also in this model the estimation procedure with `optim()` seems to work reasonably well. The estimated parameter values are quite close to the true ones, actually closer than what they were in the competing risks case in section 7.1.5. The true values are comfortably inside the estimated 95% standard positive confidence intervals. The standard deviations seem to be only of half the size as those in the competing risks study as well as smaller than the ones in the gamma model.

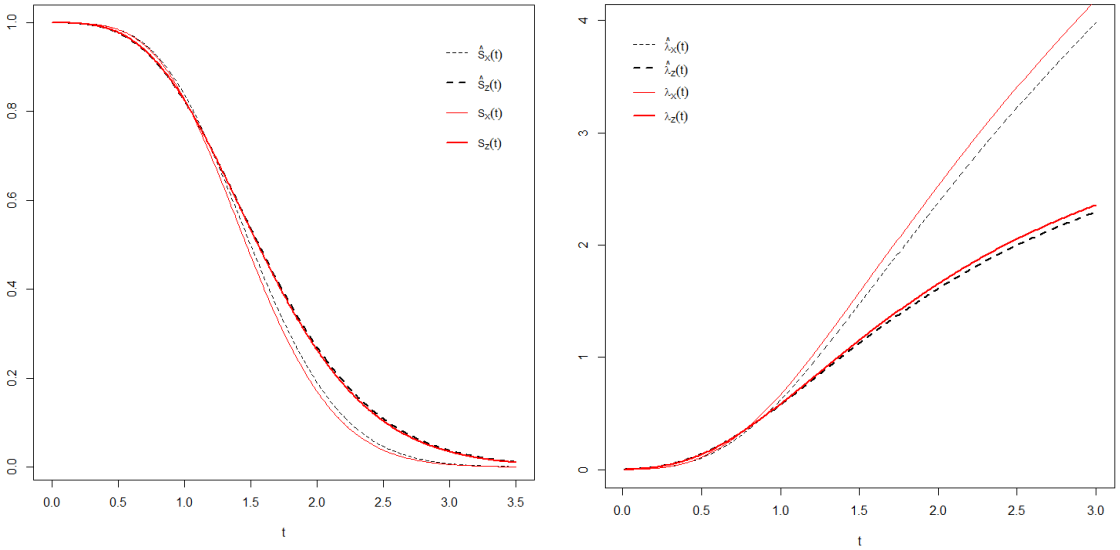

FIGURE 7.28: True and estimated marginal survival functions $S_Z(t)$ and $S_X(t)$ (left) and marginal hazard functions $\lambda_Z(t)$ and $\lambda_X(t)$ (right) with lognormal S

Also in this case we have estimated the net quantities described in section 6.5.1. To the left in figure 7.28 we have plotted the estimated and true marginal survival functions, $S_Z(t)$ from (6.14) and $S_X(t)$ from (5.3), together. We have also plotted the estimated and true marginal hazard functions functions, $\lambda_Z(t)$ from (6.15) and $\lambda_X(t)$ from (6.16), together to the right in the same figure. We can see that the true curves (in red) are close to the estimated ones (in black). The difference between the estimated and true curves seems to be slightly larger for X than for Z.

Next, we have plotted the estimated crude quantities, i.e. the sub-distribution function and cumulative sub-hazard rate of Z in figures 7.29 and 7.30.

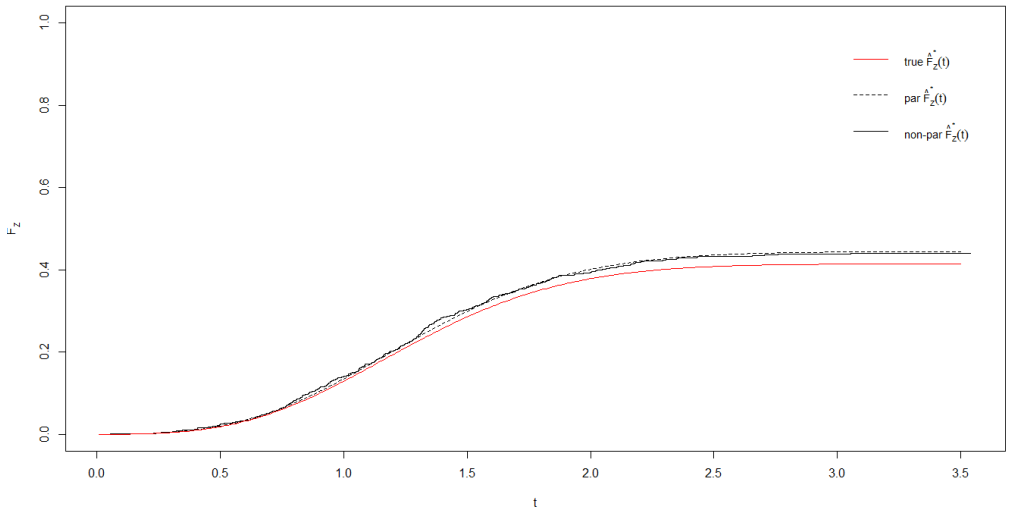

FIGURE 7.29: Parametric and non-parametric estimates of the sub-distribution function for Z, $F^*_Z(t)$ with lognormal S

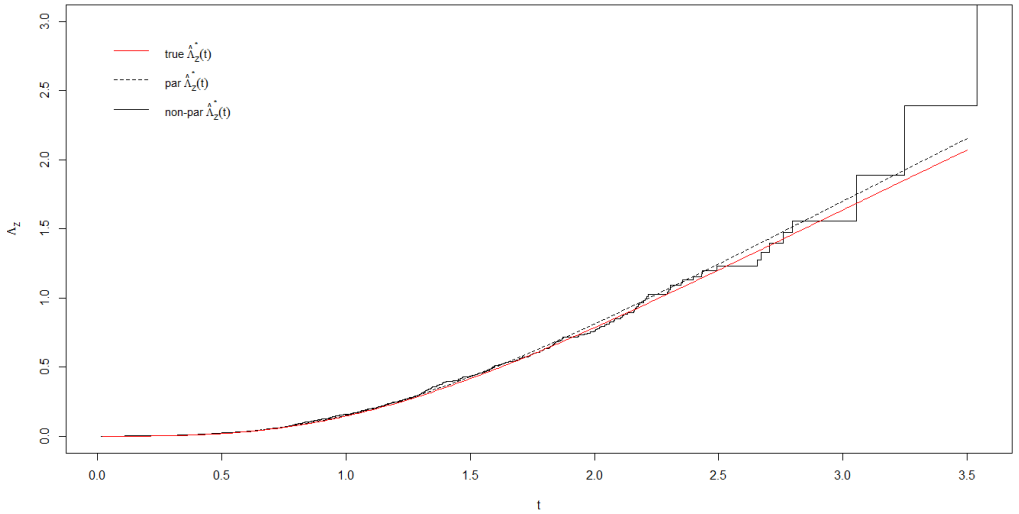

FIGURE 7.30: Parametric and non-parametric estimates of the cumulative sub-hazard rate for Z, $\Lambda^*_Z(t)$ with lognormal distributed S

The parametrically estimated sub-distribution function for Z, $F_Z^*(t)$, is found from the expression in (6.12), while the non-parametric estimate is given by (4.4). When it comes to the cumulative sub-hazard rate, $\Lambda_Z^*(t)$, the parametric estimate was found from the expression in (6.13), while the non-parametric estimate was made by using equation (4.3). As we can see from both figures, the curves all match each other very well. The true curves (red) lie slightly below the parametric and non-parametric estimates (black). For the sub-distribution function in figure 7.29 the parametric estimate seems to fit the non-parametric curve even better than the curve made with the true parameter values. For the cumulative sub-hazard rate in figure 7.30 it is more difficult to tell which of the curves that matches the non-parametric one the best.

7.2.6 Summary of simulation study - semi-competing risks

Considering the simulation studies in sections 7.2.2, 7.2.3, 7.2.4 and 7.2.5, it is clear that also for semi-competing risks data the maximum likelihood estimation with the random S models produced good results. The true parameter values were always well inside their corresponding 95% confidence intervals. As we saw for ordinary competing risks, the estimated standard deviations in the gamma model were slightly larger than in the uniform and the exponential models. For the lognormal model however, the estimated standard deviations were of approximately the same size as in the uniform and exponential models, even though it has one parameter extra.

Furthermore, we have seen that for both crude and net quantities the parametrically estimated curves matched the true curves very well. For the crude quantities we in addition saw that the parametrically estimated curves fitted well to the non-parametrically estimated ones. The deviation from the true curves that occurred for larger t-values in the marginal hazard functions is most likely due to that there are very few observations in that range.

The most important thing to take away from these simulation studies is that the estimation procedure seems to work well, and we should be able to trust the results we get when we apply the method to real data (provided that the datasets are large enough).

Chapter 8

Data analysis - competing risks

In chapter 7 we saw that the estimation of the parameters in the random S model described in chapter 6 worked well on simulated data. It would therefore be interesting to see how well the model will fit to some real data. In this chapter we will use a set of competing risks data which was also used in the project thesis [35]. Thereby we can compare the fit of the random S model to the fit of the basic model. The dataset is included in appendix C.1.

8.1 VHF-data

These data are taken from [27]. The dataset contains the failure times of a commercial airline's ARC-1 VHF communication transmitter receivers. The transmitter receivers were removed and sent to maintenance when it was assumed that they had failed. For those that had failed, we denote the time of removal by X. For those that had not yet failed, we denote the times by Z. In addition, the dataset contains some censored observations. After 630 hours all of the remaining operating units were removed (due to the airline policy). This is a type I censoring. We let this time of removal be the censoring time τ. In total there are $m = 218$ observations of X, $n = 107$ observations of Z and $r = 44$ observations of τ.

As mentioned in the chapter introduction, the basic gamma process model was fitted to this dataset in the project thesis. We will repeat some of the results regarding the fit of the basic model here, but for a complete record we refer to the project thesis [35]. First of all, it was confirmed that Cooke's condition for random signs censoring models from section 3.5 holds for these data. ($\hat{\bar{S}}_Z(t) < \hat{\bar{S}}_X(t)$). Next, the parameters of the model were estimated. A table with these estimates, along with standard deviations and upper and lower bounds for the 95 % standard positive confidence intervals is included in appendix F.3. Upon plotting the parametric estimates of the conditional sub-survival curves together with the non-parametric estimates it was clear that the basic model provided a very good fit to the data. In fact, the model fitted the data much better than any of the Wiener process models had done in [36]. Out of the four Wiener process models tested by Skogsrud, it was the one

with normally distributed S that provided the best fit, significantly better than the basic Wiener process model with constant S. It will therefore be interesting to see the effect of randomizing S in the gamma process model.

8.1.1 Uniform S

We first fitted the uniform S model to the VHF-data. This was done by maximizing the log-likelihood function in equation (6.6), just as in the simulation studies. The resulting estimates are shown in table 8.1, together with the standard deviations and 95% standard positive confidence interval limits. The complete output from R is included in appendix E.3.1.

TABLE 8.1: Maximum likelihood estimates of the parameters in the model with uniform S for the VHF-data. In addition: the correct values, standard deviations from the Hessian matrix and 95% standard positive confidence intervals

Parameter	Estimate	Standard deviation	Lower bound	Upper bound
α	0.0027	0.0003	0.0021	0.0035
β	0.9811	0.0231	0.9368	1.0275
c	0.3552	0.0823	0.2256	0.5593
A	1.1501	0.2856	0.7069	1.8711

Now, we cannot directly tell from this table whether this is a good model fit or not. What we can note however, is that these parameter estimates are very different from the estimates we obtained in the basic model (see table F.1 in appendix F.3). In particular, the values of α and c are a lot smaller, while the estimate of β is significantly larger. By looking at the correlation matrix for the parameters, which can be found in the output from R shown in appendix E.3.1, it is not surprising that if either α or c is much smaller than before, then this will be the case for the other one as well, while the value of β will be much higher. There is an estimated correlation between α and β of -0.596, while between α and c it is 0.372. The correlation between β and c is estimated to be 0.460.

In the simulation studies we saw that trying to fit a model that does not suit the data resulted in parameter estimates that were quite different from the ones in the correct model. Considering how well the basic model fitted the VHF-data in the project thesis, we might therefore suspect that the uniform model does not fit well.

8.1.2 Exponential S

Next we can try to assume that S is exponentially distributed instead. We use the log-likelihood function from (6.6) again on the VHF-data, which when maximized produced the results in table 8.2. The table contains estimated parameter values, standard deviations and 95% standard positive confidence interval limits estimated from the Hessian matrix. The complete output from R is shown in appendix E.3.2.

TABLE 8.2: Maximum likelihood estimates of the parameters in the model with exponential S for the VHF-data. In addition: the correct values, standard deviations from the Hessian matrix and 95% standard positive confidence intervals

Parameter	Estimate	Standard deviation	Lower bound	Upper bound
α	0.0027	0.0003	0.0021	0.0035
β	0.9832	0.0230	0.9391	1.0293
c	0.3653	0.0822	0.2350	0.5679
λ_S	0.9989	0.2473	0.6149	1.6229

From the results in table 8.2, we can note that the estimates for α, β and c are of the same magnitude as those provided by the uniform model. Thereby, these estimates are also quite far from the ones found by the basic model. The estimated standard deviations are in general very similar to those in the uniform model.

8.1.3 Gamma distributed S

We will now fit the model with gamma distributed S to the VHF-data. Table 8.3 provides the results from the parameter estimation using the log-likelihood function in equation (6.6). In addition to the estimated parameter values, standard deviations and bounds for the 95% standard positive confidence intervals are given. The complete output from R can be found in appendix E.3.3.

TABLE 8.3: Maximum likelihood estimates of the parameters in the model with gamma distributed S for the VHF-data. In addition: the correct values, standard deviations from the Hessian matrix and 95% standard positive confidence intervals

Parameter	Estimate	Standard deviation	Lower bound	Upper bound
α	4.6497	0.0244	4.6021	4.6978
β	0.2397	0.0033	0.2334	0.2462
c	16.7151	0.0813	16.5565	16.8752
α_S	121.9852	122.0926	17.1530	867.5086
β_S	6.9725	7.1593	0.9319	52.1692

It is clear that the parameter estimates in table 8.3 are very different from those in the uniform and exponential models. These estimates for α, β and c are much closer to the ones we got with the basic model in the project thesis. Another thing to notice is that the estimated standard deviations for α_S and β_S are quite large. They are of approximately the same magnitude as the parameter estimates themselves. This further results in very wide standard positive confidence intervals for these parameters. In such cases the standard positive interval is not suitable. For this reason, we would like to calculate the confidence intervals in an alternative way, namely by non-parametric bootstrapping as described in section 2.4.2.

TABLE 8.4: Bootstrapping results for the parameters in the gamma model for the VHF data. Includes means, biases, standard deviations, 95% percentile intervals and BC_a intervals estimated by non-parametric bootstrapping

Par.	Mean_B	Bias	SD_B	Perc. int.	BC_a int.
α	4.6708	0.0211	0.1566	(4.3012, 5.0802)	(4.3306, 5.1069)
β	0.2392	-0.0006	0.0064	(0.2237, 0.2544)	(0.2231, 0.2542)
c	16.7190	0.0039	0.0976	(16.4699, 16.8971)	(16.2054, 16.7831)
α_s	121.9531	-0.0321	0.2025	(121.4852, 122.2910)	(121.5666, 122.3566)
β_s	6.9705	-0.0020	0.0419	(6.8913, 7.0500)	(6.8805, 7.0384)

The results from the bootstrapping are shown in table 8.4. In this case we used $B = 500$ bootstrap replications. According to [11] more than 200 replications are seldom necessary, but we have the computer power to generate more. The R-code to generate the bootstrap samples and the BC_a intervals is provided in appendix D.2.5, while the complete output from R is included in appendix E.6.1. As we can see from the results, the non-parametric bootstrapping provides confidence intervals for α_S and β_S that are much smaller than the ones we found by using the Hessian matrix in table 8.3. Also the estimated standard errors are severely reduced, actually almost too much. This may suggest that our maximum likelihood estimation has converged too quickly. The `optim()` function seems to be quite sensitive to which parameter values we pick as initial values and also to the lower and upper limits (which must be set when using the "L-BFGS-B" method as we are doing). It is therefore hard to say how trustworthy these bootstrapping results are. For the parameters α, β and c however, the results are quite similar to those obtained in table 8.3.

8.1.4 Lognormal S

Finally we fit the lognormal model to the VHF-data. Following the same procedure as earlier, we maximized the log-likelihood function from equation (6.6) by using

the `condSurv`() function from appendix D.2.1. The resulting parameter estimates, estimated standard deviations and 95 % standard positive confidence intervals can be found in table 8.5. The complete results are given in appendix E.3.4.

TABLE 8.5: Maximum likelihood estimates of the parameters in the model with lognormal S. In addition, the correct values, standard deviations from the Hessian matrix and 95% standard positive confidence intervals are included.

Parameter	Estimate	Standard deviation	Lower bound	Upper bound
α	4.6831	0.0185	4.6471	4.7194
β	0.2385	0.0028	0.2331	0.2440
c	16.7205	0.0104	16.7002	16.7408
μ_S	2.8592	0.0243	2.8120	2.9072
σ_S	0.0904	0.0467	0.0328	0.2486

As we can see, the parameter estimates of α, β and c from table 8.5 are very close to the ones in the basic and gamma models. Moreover, we can notice that this time all of the standard deviations are estimated to be quite small relative to the size of the parameters, which in turn provides very narrow standard positive confidence intervals. This suggests that the parameter estimates are fairly accurate.

8.1.5 Comparison of model fits - VHF-data

We have seen that the uniform and the exponential models provided two model fits that were very different from the basic model. The parameter estimates of the gamma and the lognormal models on the other hand, were close to those of the basic model, as well as to each other. That they are similar to each other is not so surprising, as the fit of a gamma and a lognormal distribution in many cases will be able to resemble one another, especially for such a modest sample size.

To compare the different model fits to each other, we can begin by examining the maximum log-likelihood values. They are listed in table 8.6. From these values we can confirm what we guessed earlier: the uniform and exponential models do not suit the VHF-data well compared to the other models. Of the two, the uniform model has the lowest maximum log-likelihood value and performs the worst. We can also see that the fit of the basic, gamma and lognormal models are almost equally good. Of the random S models, it is the lognormal model that performs the best, but the basic model with fixed s actually seems to fit the data a tiny bit better.

The difference between the four random S models can be made even more clear by considering the estimated distributions $f_S(s)$. One can for instance plot the four

TABLE 8.6: Comparison of maximum log-likelihood values for the VHF-data from the four random S models as well as the basic model from the project thesis

Model	$\max \log L$
basic	-2377.019
uniform	-2389.049
exponential	-2389.753
gamma	-2377.063
lognormal	-2377.047

distributions of S together in the same figure. This is done in figure 8.1. The parameter estimate for s from the basic model is also marked in the same plot. Yet again, the difference between the estimates in the uniform and exponential models and the gamma and lognormal models is striking. As we can see, the basic estimate of s is smaller than the mean values of S in the gamma and lognormal models. This is natural since s corresponds to $\mathrm{E}[S|S < c]$ not $\mathrm{E}[S]$. The four choices of $f_S(s)$ indicate quite different maintenance policies for the VHF transmitter receivers.

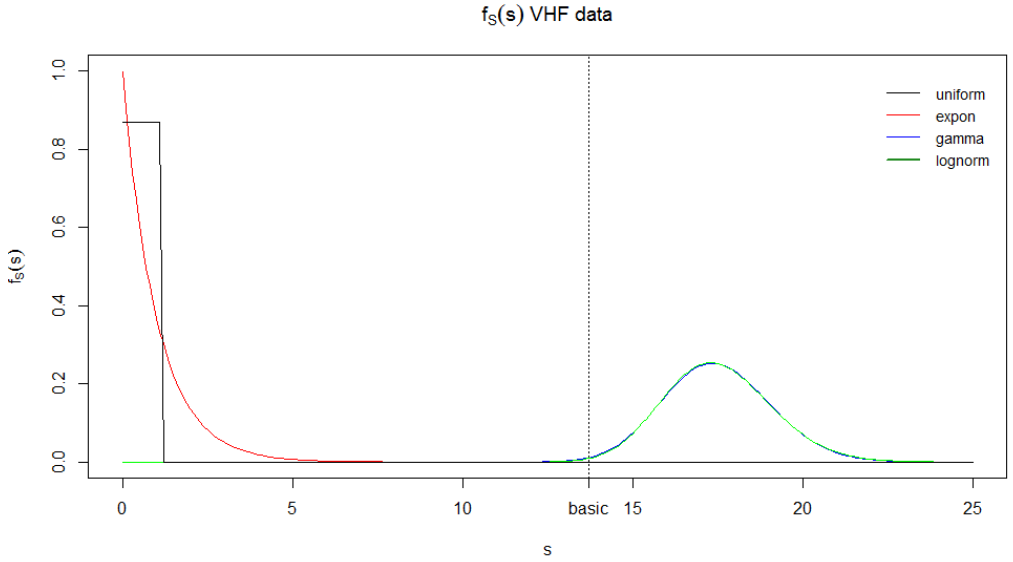

FIGURE 8.1: Estimated distributions of S for the VHF-data in the four random S models as well as the estimate of s from the basic model

We can in addition compare the estimates of $F_S(c)$ made with the different models. The results are listed in table 8.7. There, we can see that all of the estimates of $F_S(c)$ are actually quite similar to each other. Still, the estimates made with the uniform and the exponential models, are a little smaller than the others. From the project thesis we can recall that the non-parametric estimate was 0.3306. Compared to this value, the lognormal model is the closest.

TABLE 8.7: Estimated values of $F_S(c)$ for the VHF-data in the random S models as well as in the the basic model

Model	$\widehat{F_S(c)}$
basic	$(\hat{q} =)$ 0.3159
uniform	0.3089
exponential	0.3057
gamma	0.3190
lognormal	0.3193

To further evaluate the fit of our random S models to the VHF-data we can consider the plots of the parametrically estimated conditional sub-survival curves. From before we know that these curves can be made from the expressions in section 6.4.1. The resulting plots are given in figures 8.2 and 8.3.

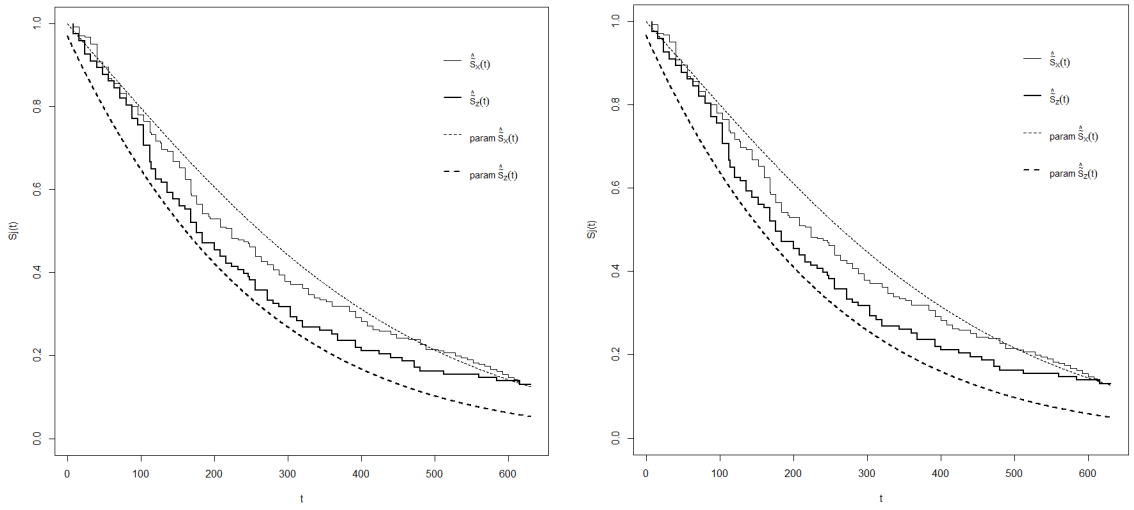

FIGURE 8.2: Parametric and non-parametric estimates of the conditional sub-survival functions from the uniform(left) and the exponential (right) models for the VHF-data

The figures 8.2 and 8.3 confirm what we learned from the table of maximum log-likelihood values (8.6). Considering figure 8.2 first, the parametrically estimated conditional sub-survival curves of the uniform and exponential models are relatively far form the non-parametrically estimated curves, both for Z (thick lines) and for X (thin lines). The curves in the uniform model (to the left) display a slightly closer fit than those in the exponential model (to the right). If we move on to figure 8.3, we can see that the parametrically estimated curves of the gamma and lognormal models are much closer to the non-parametric curves than the uniform and exponential curves were. The difference between the fit of the gamma model (to the left) and the lognormal model (to the right) is almost impossible to see. Both

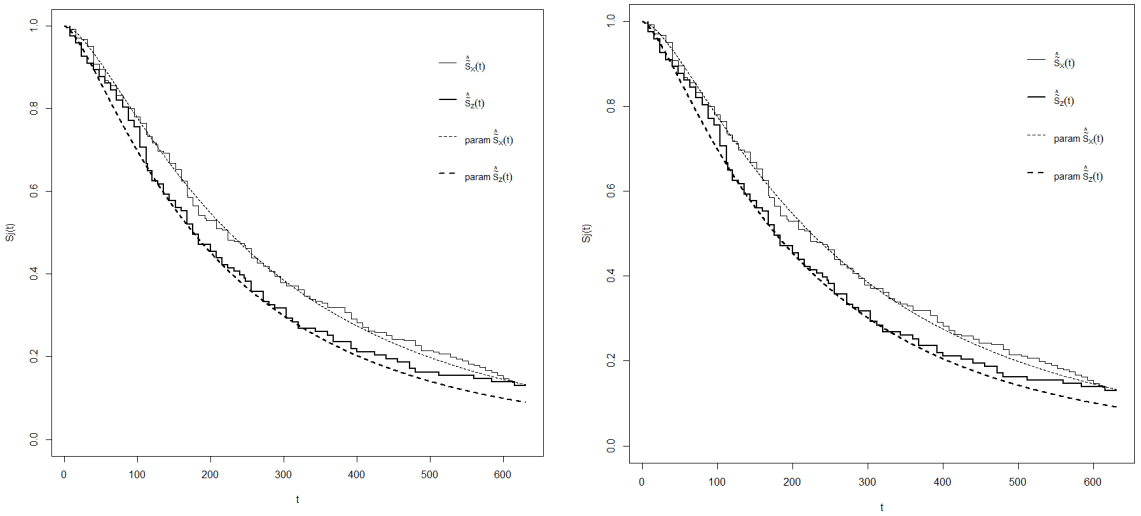

FIGURE 8.3: Parametric and non-parametric estimates of the conditional sub-survival functions from the gamma(left) and the lognormal (right) models for the VHF-data

models seem to fit very well to the data. It is logical that the models with a two-parameter distribution for S are able to fit the data better than the one-parameter distributions, as they are more flexible.

Since it is difficult to say which of the gamma, lognormal and the basic model that suits the data the best, we choose to plot the parametrically estimated conditional sub-survival functions of these three models together in the same figure. The parametrically estimated curves from the basic model were made in the project thesis [35] and is reprinted here in figure 8.4.

As we can see, the three models all seem to fit well to the data. The curves of the lognormal model completely overlap the curves of the gamma model, so these models are both represented by the blue curves. The curves of the basic model are drawn in black dashed lines. The lines of the parametric estimates of $\tilde{S}_X(t)$ are almost identical, and it is hard to tell whether any of the three models is better than the others. When it comes to the parametric estimates of $\tilde{S}_Z(t)$, there is a more distinct difference for $t > 200$. The curve from the gamma and lognormal models seems to lie a little further up (and thus closer to the non-parametric curve) than the line from the basic model. Thus, based on this figure alone one may be tempted to say that the gamma or the lognormal model is the best out of the three. On the other hand, the maximum log-likelihood values suggested that the basic model had a tiny advantage over the others. Overall it is difficult to make a judgement with respect to which of the three models that is the best, since the difference is so small. We should however keep in mind that computationally, the basic model

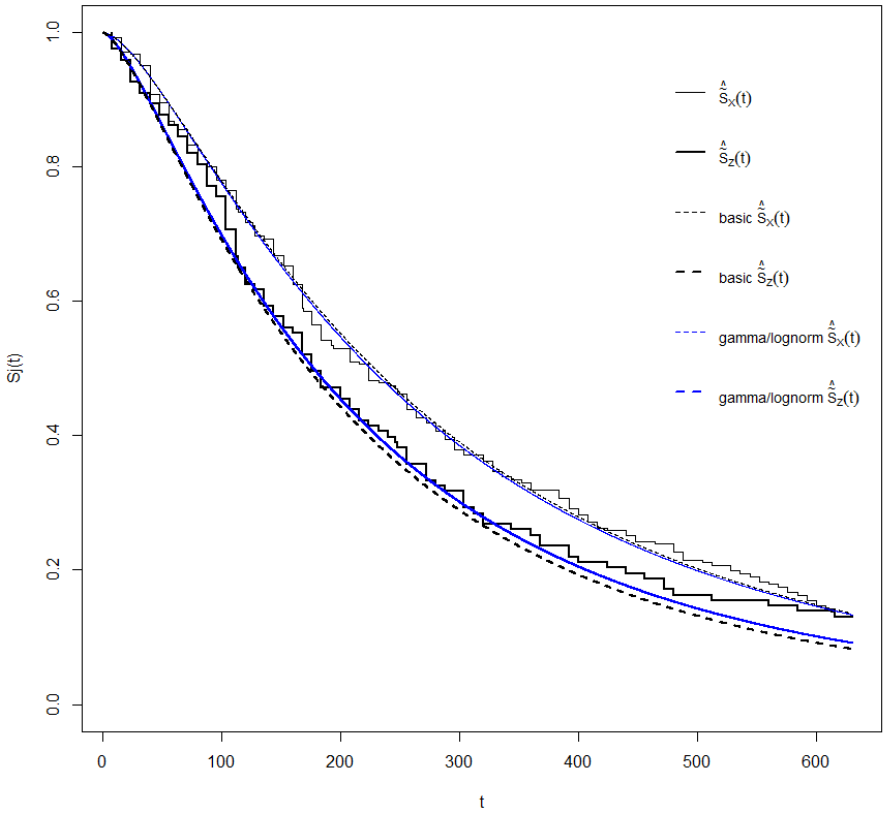

FIGURE 8.4: Parametric and non-parametric estimates of the conditional sub-survival functions from the gamma and lognormal(blue) and the basic (black) models for the VHF-data

is simpler, even though the models have the same number of parameters. This is because no numerical integration is required to calculate the log-likelihood function in the basic model.

To summarize, it actually seems that for the VHF-data it was not necessary, or even beneficiary, to randomize the level S in the gamma process model. In theory, the random S extension of the basic model provides more flexibility, and as mentioned at the beginning of this chapter, randomizing S had a positive effect for the corresponding Wiener process model on the VHF-data in [25]. At the same time, we have to recall that the fit of the basic gamma process model was already very good. The main reason for the ability of the gamma process models to fit so well to the data over for instance the corresponding Wiener process models probably lies in the shape function $v(t)$.

Chapter 9

Data Analysis
- semi competing risks

In chapter 7 we also saw that our estimation method for the random S model worked well for simulated semi-competing risks data. In this chapter we will apply the random S gamma process model to two sets of real semi-competing risks data. The first is related to patients with carcinoma of the lung, and does not contain any censored observations. The second involves data on patients who have undergone bone marrow transplants, and this dataset does contain censored observations. The datasets are included in appendix C.2 and C.3 respectively.

9.1 Carcinoma data

This dataset is taken from [20]. It describes the survival times of people diagnosed with inoperable carcinoma (a type of cancer) of the lungs that were treated by a certain drug. We denote these observations by X. For some patients, treatment was terminated or they started with a different kind of treatment instead (due to specific kinds of disease progression). The time till these types of events are denoted by Z. In addition, the dataset includes the eventual survival times of the patients who ended or changed their treatment. We will denote these times by X_Z. All times are given as number of weeks. There are no censored observations.

The carcinoma data were also used in my project thesis for the basic gamma process model [35]. The data were then treated as ordinary competing risks data, thereby ignoring the information about the times of death for the patients that first experienced the non-terminating event. When the data is treated like competing risks data, and we plot the non-parametric estimates of the conditional sub-survival curves, it is evident that $\hat{\bar{S}}_Z(t) < \hat{\bar{S}}_X(t)$ for all t (see [35]). This suggests that we can fit a random signs censoring model to the data. We will now fit the semi-competing risks versions of the uniform, exponential, gamma and lognormal models without censoring to the carcinoma dataset. In the dataset there are in total $m = 33$ observations of the survival times for the patients on the original treatment and $n = 28$

observations of times of removal from the original treatment and thus also $n = 28$ subsequent death times. Thus the total number of observations is quite small.

9.1.1 Uniform S

We begin by assuming that S is uniformly distributed on $[0, A]$. We next insert this choice of $f_S(s)$ into the log-likelihood function from equation (6.11). This function may then be maximized for the carcinoma data by the `estSemi()` function, which can be found in appendix D.2.3. By doing this, we obtained the parameter estimates shown in table 9.1. In addition, the table presents the estimated standard deviations as well as lower and upper bounds for the 95% standard positive confidence intervals calculated from the Hessian matrix. The complete output from R is shown in appendix E.4.1.

TABLE 9.1: Maximum likelihood estimates of the parameters in the model with uniform S without censoring for the carcinoma data. In addition: standard deviations from the Hessian matrix and 95% standard positive confidence intervals

Parameter	Estimate	Standard deviation	Lower bound	Upper bound
α	0.1676	0.1370	0.0337	0.8322
β	0.6878	0.1045	0.5107	0.9263
c	0.8027	0.6616	0.1596	4.0376
A	1.7488	1.4616	0.3398	8.9988

From the results in table 9.1 we can for instance note that the standard deviations in general are of the same size order as the parameter estimates. We can also see that the estimates of α, β and c are all very different from what they were in the project thesis with the basic model [35]. However, this does not really tell us much, since in that case the data were treated as competing risks data and all of the survival times observed after the non-terminal event were ignored.

As in the simulation studies, we have plotted the estimated marginal survival functions $\hat{S}_Z(t)$ and $\hat{S}_X(t)$. These were made from inserting the parameter estimates from table 9.1 into equations (6.14) and (5.3). The resulting curves are shown together to the left of figure 9.1. We have also plotted the estimated marginal hazard functions $\hat{\lambda}_Z(t)$ and $\hat{\lambda}_X(t)$ together to the right in the same figure. They were found by using the expressions in (6.15) and (6.16). $\hat{S}_Z(t)$ and $\hat{\lambda}_Z(t)$ are plotted in thick lines, while $\hat{S}_X(t)$ and $\hat{\lambda}_X(t)$ are plotted in thin lines.

(The estimated crude quantities will be plotted later, together with the estimates from the other models).

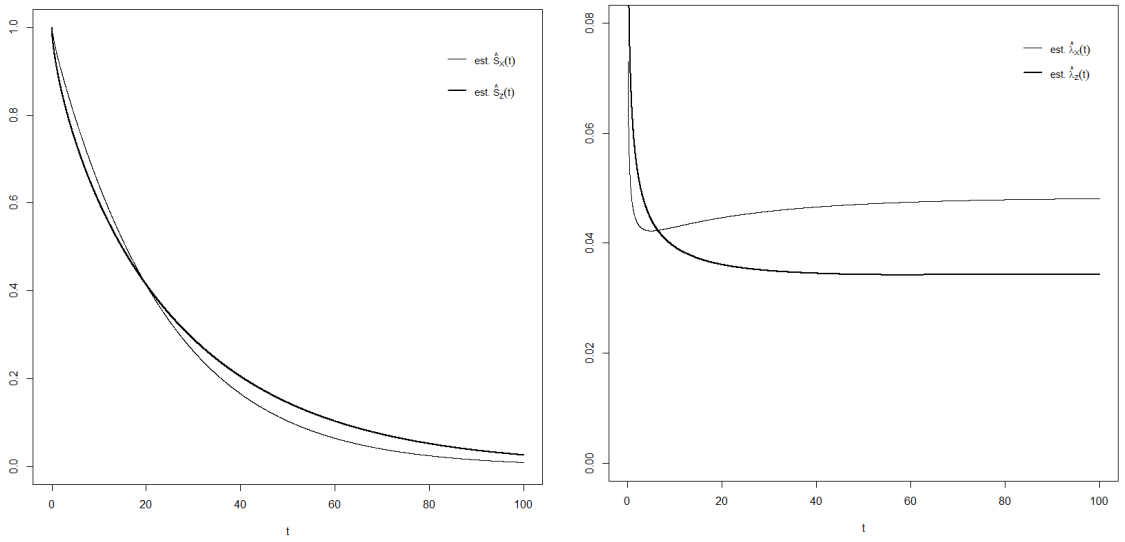

FIGURE 9.1: Estimated marginal survival functions $\hat{S}_Z(t)$ and $\hat{S}_X(t)$ (left) and hazard functions $\hat{\lambda}_Z(t)$ and $\hat{\lambda}_X(t)$ (right) with uniform S

9.1.2 Exponential S

Next, we fit the model with exponential S to the data. The results from the maximum likelihood estimation procedure are shown in table 9.2. The table presents the estimated parameter values and corresponding standard deviations, as well as lower and upper bounds for the 95% standard positive confidence intervals calculated from the Hessian matrix. The complete output from `estSemi()` is shown in appendix E.4.2.

TABLE 9.2: Maximum likelihood estimates of the parameters in the model with exponential S without censoring for the carcinoma data. In addition: standard deviations from the Hessian matrix and 95% standard positive confidence intervals

Parameter	Estimate	Standard deviation	Lower bound	Upper bound
α	0.1383	0.0962	0.0354	0.5405
β	0.7144	0.0920	0.5550	0.9196
c	0.6806	0.5028	0.1600	2.8957
λ_S	0.8824	0.6797	0.1950	3.9932

The estimated parameter values for α, β and c in table 9.2 are quite close to those obtained in the uniform model in table 9.1. Also the corresponding estimated standard deviations are of approximately the same size as they were there.

As in the simulation studies, we have plotted the estimated marginal survival functions $\hat{S}_Z(t)$ and $\hat{S}_X(t)$ together. This is shown to the left in figure 9.2. The parametric estimate of $S_Z(t)$ was made from equation (6.14), while for $S_X(t)$ it is given

by (5.3). We have also plotted the estimated marginal hazard functions $\hat{\lambda}_Z(t)$ and $\hat{\lambda}_X(t)$ together to the right in the same figure. Like before, these functions were estimated by using equation (6.15) and (6.16) respectively. In shape, the estimated functions are quite similar to the ones estimated in the uniform model.

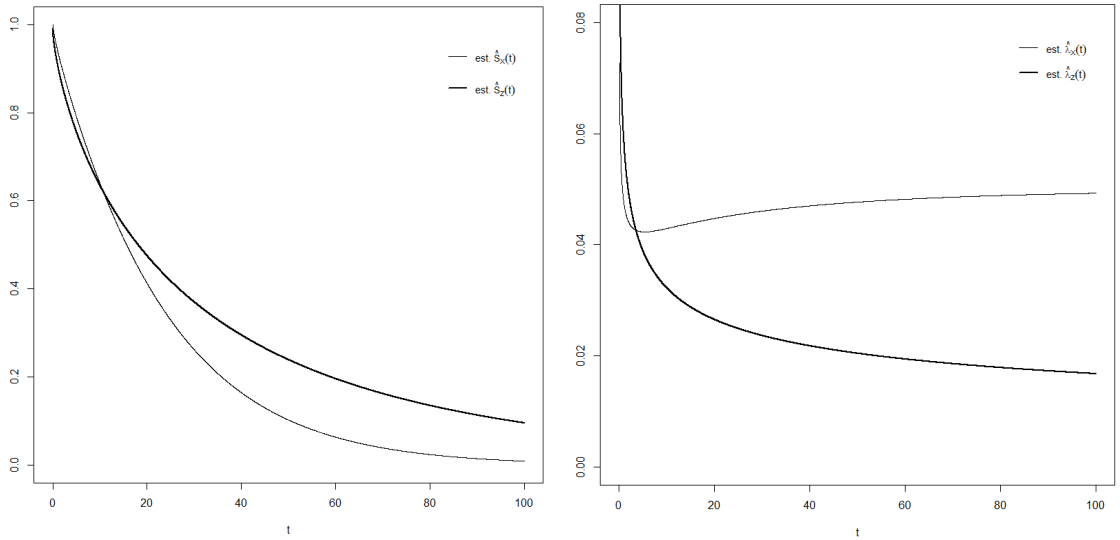

FIGURE 9.2: Estimated marginal survival functions $\hat{S}_Z(t)$ and $\hat{S}_X(t)$ (left) and hazard functions $\hat{\lambda}_Z(t)$ and $\hat{\lambda}_X(t)$ (right) with exponential S

9.1.3 Gamma distributed S

We will now move on to the gamma model. Following the same procedure as before, we can maximize the log-likelihood function from equation (6.11) by the `estSemi()` function. The results are shown in table 9.3. The table presents the estimated parameter values and corresponding standard deviations, as well as lower and upper bounds for the 95% standard positive confidence intervals calculated from the Hessian matrix. The complete output from R is given in appendix E.4.3.

TABLE 9.3: Maximum likelihood estimates of the parameters in the model with gamma S without censoring for the carcinoma data. In addition: standard deviations from the Hessian matrix and 95% standard positive confidence intervals

Parameter	Estimate	Standard deviation	Lower bound	Upper bound
α	2.1788	3.5832	0.0868	54.7135
β	0.3620	0.1850	0.1329	0.9858
c	5.6606	6.7956	0.5382	59.5333
α_s	5.9301	4.5708	1.3091	26.8631
β_s	0.9499	0.7051	0.2217	4.0692

As we can see from table 9.3, the estimates of α, β and c are all quite different from the ones obtained by the uniform and exponential models. The values of α and c

are a great deal larger, while the value of β is a lot smaller than in the two previous models. Just as in the VHF data analysis, this has to do with the fact that all of these parameters are strongly correlated to each other, so that a change in one of them will generate a change in the other two as well. From the correlation matrix in appendix E.4.3, we can for instance see that the correlation between α and β in this case is estimated to be -0.992 and between α and c it is 0.997.

The standard deviations are also in this case estimated to be of the same magnitude as the parameter estimates themselves. Since the parameter estimates here are considerably larger than in the two previous models, this causes the 95% standard positive confidence intervals to be very wide. Recall that this also happened in the competing risks gamma model when we analysed the VHF-data. As we did there, we have used non-parametric bootstrapping to calculate alternative confidence intervals. The results are shown in table 9.4. The output from R is given in appendix E.6.2.1.

TABLE 9.4: Maximum likelihood estimates of the parameters in the gamma model for the carcinoma data. In addition: means, biases, standard deviations, 95% percentile intervals and BC_a intervals from non-parametric bootstrapping

		Non-parametric				
Par.	Est.	Mean_B	Bias	SD_B	Percentile int.	BC_a int.
α	2.1788	3.0834	0.9045	2.4543	(0.1685, 8.1359)	(0.1124, 7.4117)
β	0.3620	0.3902	0.0282	0.1371	(0.2230, 0.6874)	(0.2291, 0.7246)
c	5.6606	7.1056	1.4450	4.4335	(0.8508, 14.9927)	(0.5248, 14.1988)
α_s	5.9301	6.6894	0.7593	3.1743	(1.7220, 14.1156)	(1.4160, 12.7197)
β_s	0.9499	1.1262	0.1763	0.6313	(0.3515, 2.8402)	(0.3442, 2.8628)

Both the percentile intervals and the BC_a intervals obtained in table 9.4 are narrower than the standard positive intervals from table 9.3. However, they are still quite wide. This suggests that there is considerable uncertainty regarding the parameter estimates of the gamma model. The range of estimated parameter values in the bootstrap samples is quite wide. We can further notice that all of the parameters have an estimated positive bias. Many of these are quite large compared to the size of the estimated standard errors.

As previously, we have plotted the estimated marginal survival functions $\hat{S}_Z(t)$ from (6.14) and $\hat{S}_X(t)$ from (5.3). This is done to the left in figure 9.3. We have also plotted the estimated marginal hazard functions $\hat{\lambda}_Z(t)$ from (6.15) and $\hat{\lambda}_X(t)$ from (6.16) together to the right in the same figure. Again, $\hat{\lambda}_Z(t)$ and $\hat{S}_Z(t)$ are plotted in thick lines while $\hat{\lambda}_X(t)$ and $\hat{S}_X(t)$ are plotted in thin lines.

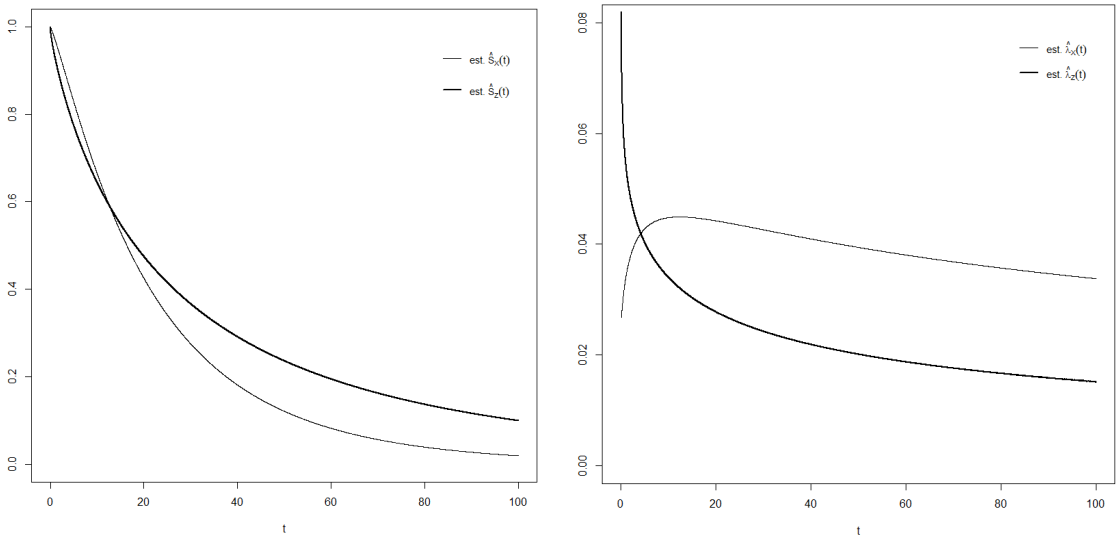

FIGURE 9.3: Estimated marginal survival functions $\hat{S}_Z(t)$ and $\hat{S}_X(t)$ (left) and hazard functions $\hat{\lambda}_Z(t)$ and $\hat{\lambda}_X(t)$ (right) with gamma distributed S

9.1.4 Lognormal S

Finally, we fit the lognormal model to the carcinoma data. The results from the maximum likelihood estimation procedure done with `estSemi`() are shown in table 9.5. The table presents the estimated parameter values and corresponding standard deviations, as well as lower and upper bounds for the 95% standard positive confidence intervals calculated from the Hessian matrix. For the complete output (as given by R), see appendix E.4.4.

TABLE 9.5: Maximum likelihood estimates of the parameters in the model with lognormal S without censoring for the carcinoma data. In addition: standard deviations from the Hessian matrix and 95% standard positive confidence intervals

Parameter	Estimate	Standard deviation	Lower bound	Upper bound
α	1.8396	3.0683	0.0700	48.3595
β	0.3809	0.1923	0.1416	1.0248
c	5.0005	6.0727	0.4627	54.0431
μ_S	1.6562	1.2040	0.3984	6.8854
σ_S	0.4786	0.2054	0.2064	1.1099

These estimates are not that far from the ones obtained in the gamma model. Also here the standard deviations are quite large and the standard positive confidence intervals are very wide. Therefore, we have used non-parametric bootstrapping in this model as well. The results are given in table 9.6, while the complete output from R can be found in in appendix E.6.2.2.

TABLE 9.6: Maximum likelihood estimates of the parameters in the lognormal model for the carcinoma data. In addition: means, biases, standard deviations, 95% percentile intervals and BC_a intervals from non-parametric bootstrapping

Par.	Est.	Mean_B	Bias	SD_B	Percentile int.	BC_a int.
α	1.8396	3.1258	1.2861	2.6428	(0.1997, 7.9788)	(0.1501, 7.8040)
β	0.3809	0.3941	0.0133	0.1394	(0.2253, 0.6768)	(0.2302, 0.7067)
c	5.0005	7.1056	2.1051	4.7934	(1.0459, 15.3990)	(0.8575, 14.7481)
μ_s	1.6552	1.7284	0.0722	0.8365	(0.1154, 2.8295)	(0.0292, 2.7658)
σ_s	0.4786	0.4937	0.0151	0.1478	(0.2818, 0.8637)	(0.3051, 0.9843)

In table 9.6 we can see many of the same tendencies as we saw in the gamma model. The non-parametric bootstrapping provides us with confidence intervals that are smaller than the standard positive ones from table 9.5, but they are still quite wide. There is also a positive bias associated with each of the parameters that for some of the parameters is quite large compared to the estimated standard errors.

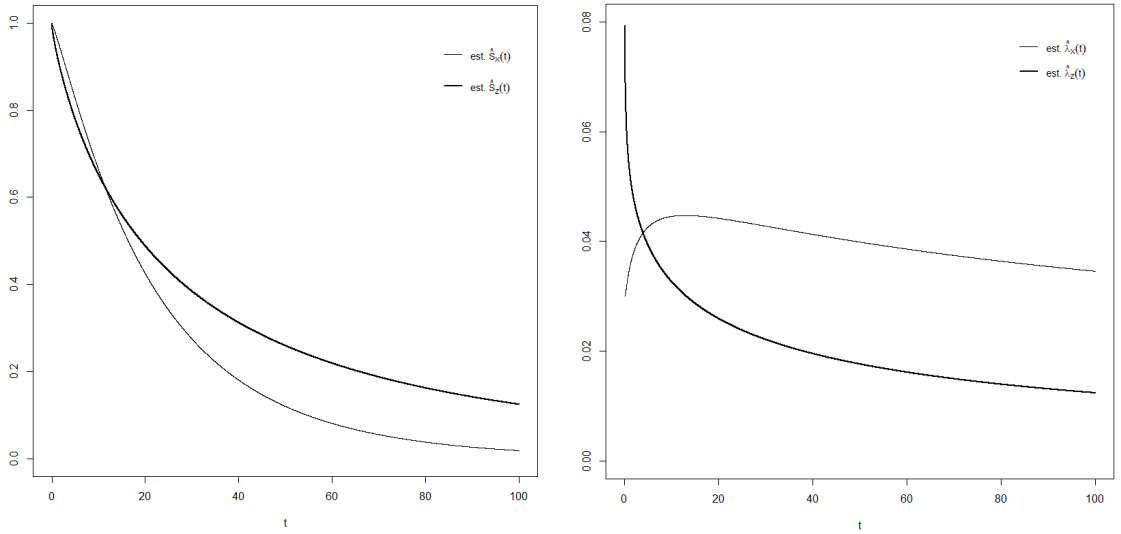

FIGURE 9.4: Estimated marginal survival functions $\hat{S}_Z(t)$ and $\hat{S}_X(t)$ (left) and hazard functions $\hat{\lambda}_Z(t)$ and $\hat{\lambda}_X(t)$ (right) with lognormal S

As in the other cases, we have plotted the estimated marginal survival functions $\hat{S}_Z(t)$ and $\hat{S}_X(t)$. The estimates were found using the expressions (6.14) and (5.3). The curves are shown together, in thick and thin lines respectively, to the left in figure 9.4. We have also estimated the marginal hazard functions using (6.15) and (6.16). $\hat{\lambda}_Z(t)$ (thick line) and $\hat{\lambda}_X(t)$ (thin line) are plotted together to the right in the same figure. The curves look very similar to the ones obtained by the gamma model.

9.1.5 Comparison of model fits - carcinoma data

We will in this section compare the four model fits to each other. To get an indication of which of the models that fits the carcinoma data the best, we can consider the maximum log-likelihood values displayed in table 9.7. From this table we can see that the maximum value is obtained with the lognormal model. The gamma model is very close to having the same value, while the uniform and exponential maximum log-likelihood values are a little lower, but still relatively close to the others.

TABLE 9.7: Comparison of maximum log-likelihood values in the four random S models for the carcinoma data

Model	$\max \log L$
uniform	-382.4325
exponential	-382.9794
gamma	-380.8140
lognormal	-380.7409

To find out more with respect to the differences of the model fits, we can plot the estimated marginal survival function for the time to the non-terminal event, $\hat{S}_Z(t)$, from all four models together. This is done in figure 9.5.

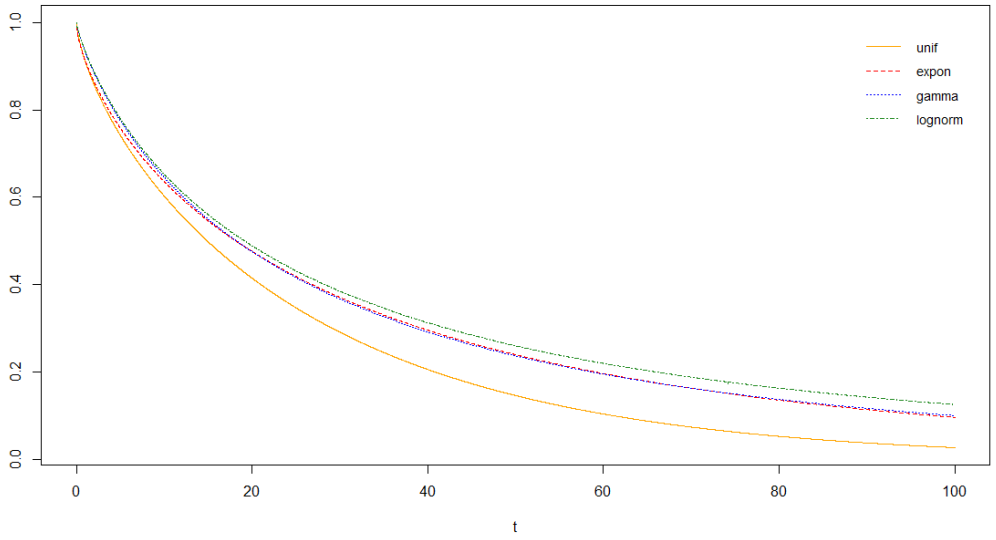

FIGURE 9.5: Comparison of parametrically estimated $\hat{S}_Z(t)$ in the random S models for the carcinoma data

As we can see, the uniform curve (orange) stands out from the others, and decreases the fastest. The curves of the exponential model (red) is almost identical to that

from the gamma model (blue), while the curve of the lognormal model (green) lies a little further up than the rest.

The reason for the similarity in the model fits becomes more evident when we plot the estimated distributions of S together. In figure 9.6 this is done. As we can see, the main parts of the four densities lie much closer together than what they did for the VHF data in figure 8.1. They also lie closer to zero. Moreover, the densities of the gamma and lognormal distributions (in blue and green) are much wider than what they were for the VHF-data. Thereby the uniform and exponential models (in black and red), who have to be larger than 0 at $t = 0$, are not as different from the gamma and lognormal distributions for the carcinoma data.

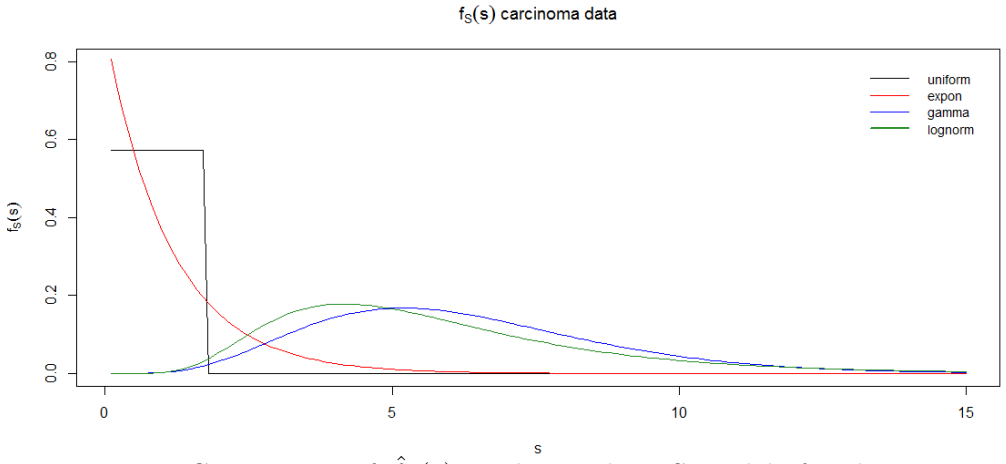

FIGURE 9.6: Comparison of $\hat{f}_S(s)$ in the random S models for the carcinoma data

Also for these data it might be of interest to compare the estimates of $F_S(c)$ made with the different models. These are shown in table 9.8. The estimates are very similar to each other, even though the exponential estimate is a little lower than the others. From the project thesis we can recall that the non-parametric estimate of q was 0.4581, which also is very close to the estimates in table 9.8.

TABLE 9.8: Comparison of the estimates of $F_S(c)$ in the four random S models for the carcinoma data

Model	$F_S(c)$
uniform	0.4590
exponential	0.4515
gamma	0.4617
lognormal	0.4612

We can further plot both the parametric and the non-parametric estimates of the crude quantities $F_Z^*(t)$ and $\Lambda_Z^*(t)$. As we can recall from section 6.5.1, the parametric estimates can be found by equation (6.12) and integrating equation (6.13). How to find the non-parametric estimates was given by (4.4) and (4.3). We plot these parametric and non-parametric estimates together in the same figure. In figures 9.7 and 9.8 this is done with the four random S models.

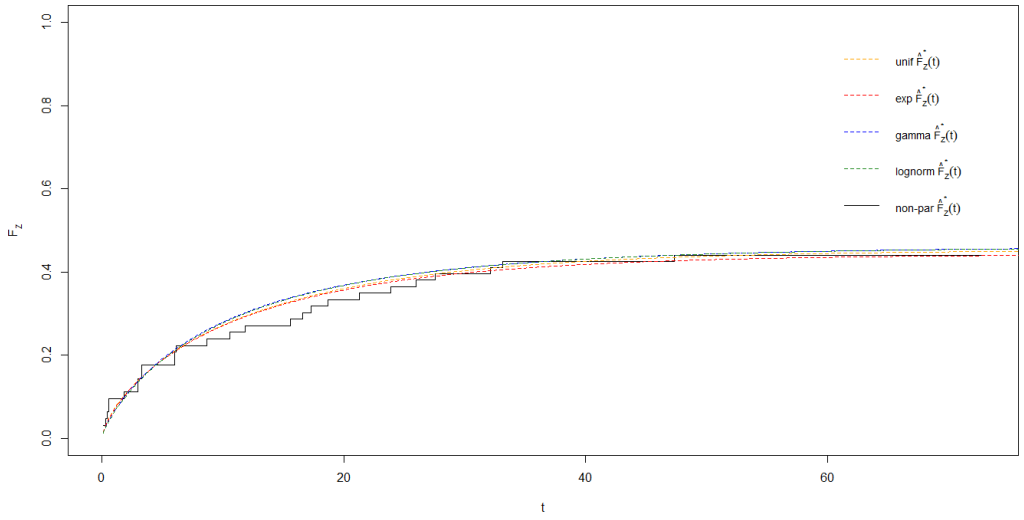

FIGURE 9.7: Parametric and non-parametric estimates of the sub-distribution function for Z, $F_Z^*(t)$, for the carcinoma data

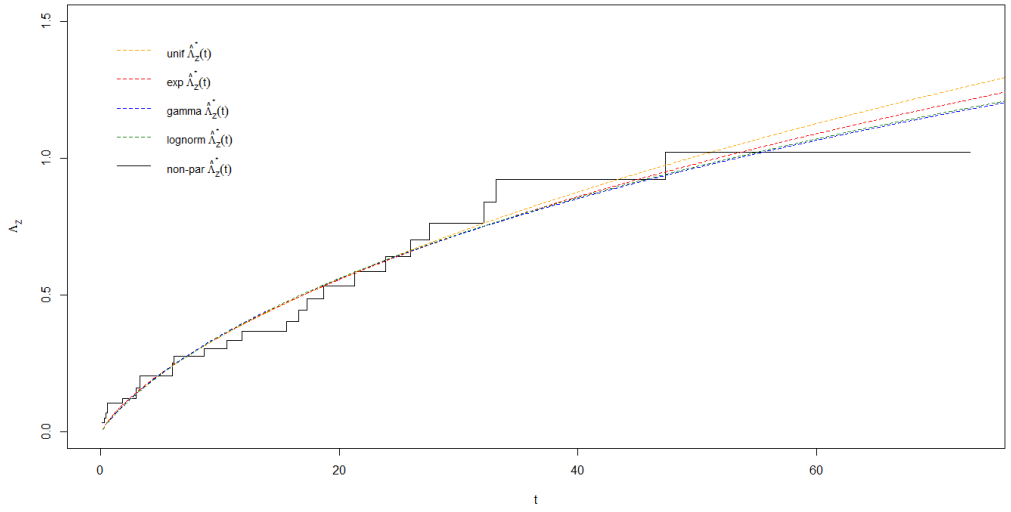

FIGURE 9.8: Parametric and non-parametric estimates of the cumulative sub-hazard rate for Z, $\Lambda_Z^*(t)$, for the carcinoma data

For both the sub-distribution function and the cumulative sub-hazard rate, the fit of the parametric curves seems to very good to the non-parametric estimate (in black). The four parametric lines lie very close together, although in figure 9.8 they seem to begin to spread as t grows and the number of observations gets lower. It is

hard to tell from these figures if any of the models fit better than the others to the data.

To summarize, we have seen that the four random S models seem to fit the carcinoma data almost equally well. In spite of having quite different parameter estimates for α, β and c, the uniform and exponential models seem to fit only slightly worse to the data than the gamma and lognormal models, considering the maximum log-likelihood values. We have also seen that the uncertainty in the parameter estimates is relatively large in all four models. Many of the standard deviations are of the size of the parameter estimates themselves. This causes large standard positive confidence intervals. Also when bootstrapping is applied, we see that there is considerable uncertainty in the estimates. Perhaps the dataset is too small to make any proper inferences. Still, the estimates do not seem to be too bad, as the parametric estimates of $F_Z^*(t)$ and $\Lambda_Z^*(t)$ fit quite well to the non-parametric estimates.

9.2 Bone marrow transplant data

This dataset is collected from [19], and is also included in appendix C.3. The data are originally from a study by Copeland et al [8]. The dataset contains observations of 137 patients that have undergone allogeneic bone marrow transplantation as treatment for acute leukemia. The terminal event in this case is death, while the non-terminal event is cancer relapse. For more information about the data and the study, see [19]. As described in chapter 4, there are four possible types of observations when we are dealing with semi-competing risks data including censorings. For this dataset we may observe:

1. Only X - time until death (from any cause)
2. Both Z and X_Z - time to cancer relapse, and subsequent time of death
3. Both Z_O and τ_O - time to cancer relapse, and then a censoring time
4. Only τ - censoring time

All times are measured in days from the time of transplantation. In the dataset there are $m = 41$ observations of just death times, $n = 40$ observations of both times to relapse and times to subsequent death, $w = 2$ observations of times to relapse and then censoring times, and $r = 54$ observations of only censorings. Note that also in this case the total number of observations is quite small compared to the simulation studies, and that the percentage of censored observations is higher.

If we first treat the data as ordinary competing risks data, and ignore the extra information that comes with semi-competing risks data, we can plot the non-parametric

conditional sub-survival curves of Z and X in the same manner as before, using equation (3.4). The results are shown in figure 9.9. There we can see that $\hat{\bar{S}}_Z(t)$ (thick line) $< \hat{\bar{S}}_X(t)$ (thin line) for most t-values except the smallest.

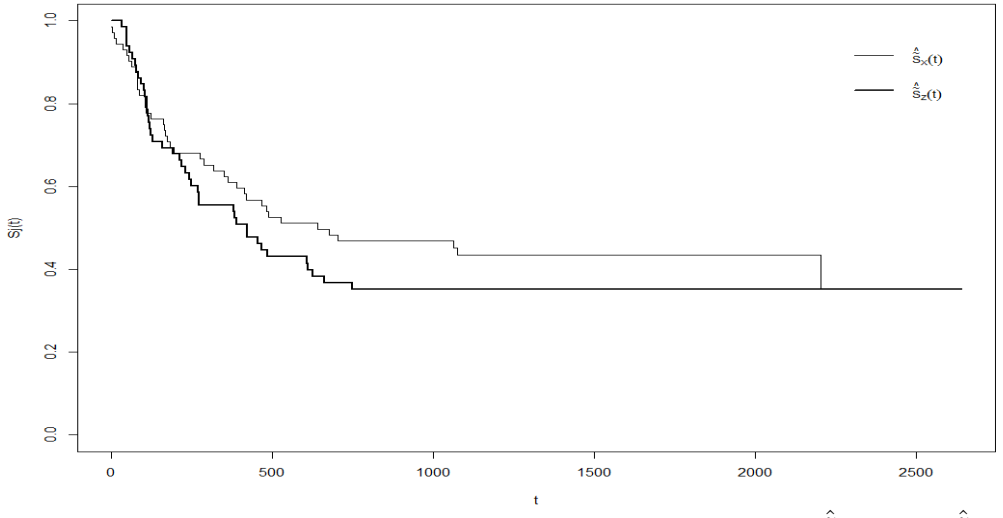

FIGURE 9.9: Non-parametric conditional sub-survival functions $\hat{\bar{S}}_Z(t)$ and $\hat{\bar{S}}_X(t)$ for the bone marrow transplant data

Since $\hat{\bar{S}}_Z(t) < \hat{\bar{S}}_X(t)$ is a necessary condition for a random signs censoring model (theorem 3.3), we want to investigate this further. We can make a plot of $\hat{\bar{S}}_X(t) - \hat{\bar{S}}_Z(t)$. This is done in figure 9.10. There, it is confirmed that $\hat{\bar{S}}_Z(t) < \hat{\bar{S}}_X(t)$ holds for $t > 100$. Since this is the majority of the t-values, our conclusion is that a random signs censoring model can at least not be excluded from possibility.

The bone marrow transplant data were also studied in a semi-competing risks setting by Fine, Jiang and Chappell in [12]. To handle the dependent censoring, it was assumed that the bivariate distribution of X and Z was a known copula, more specifically a gamma frailty copula. Among other quantities, they estimated the marginal survivor function for the time to relapse. The resulting function is plotted (solid line) in figure 9.11. The dashed lines represent the 95% confidence interval limits, while the dotted line is the ordinary Kaplan-Meier estimate.

It will be interesting to see how the fit of our random S models will be compared to this. Also in this case we will test the uniform, exponential, gamma and lognormal distributions as $f_S(s)$.

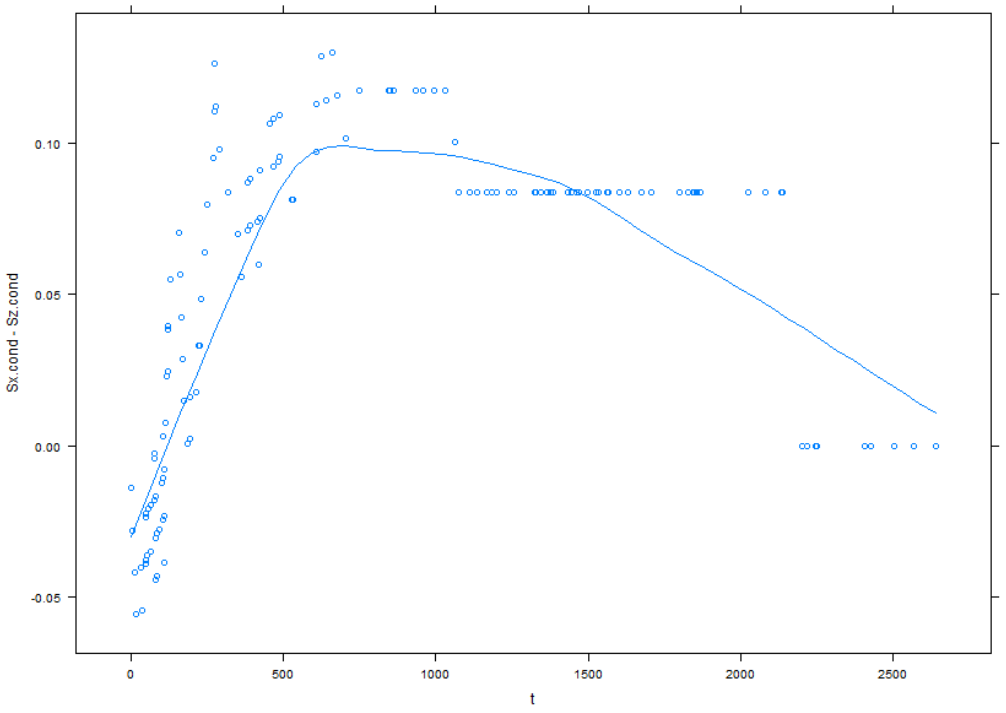

FIGURE 9.10: The difference $\hat{\tilde{S}}_X(t) - \hat{\tilde{S}}_Z(t)$ plotted as a function of t for the bone marrow transplant data

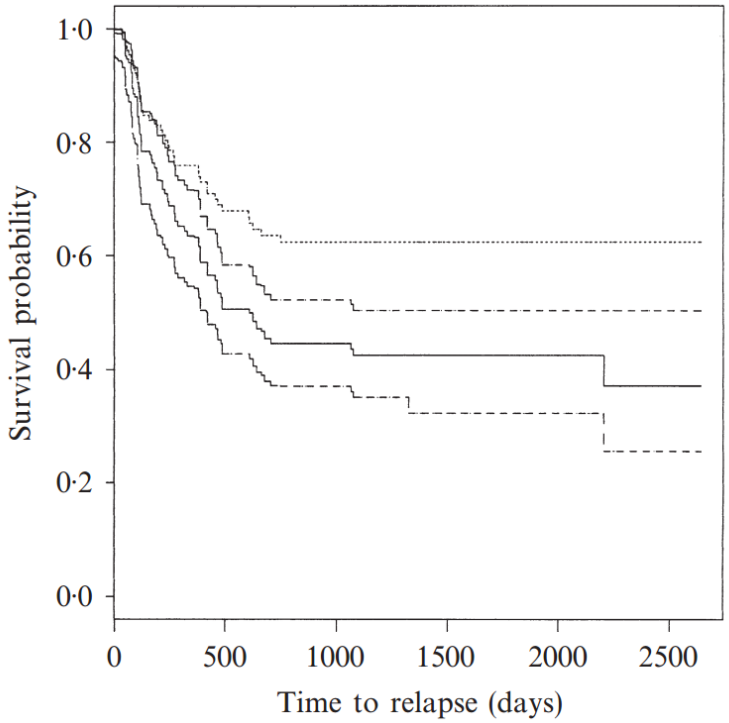

FIGURE 9.11: Estimate from [12] of the survivor function for the time to relapse along with 95% confidence interval limits and the ordinary Kaplan-Meier estimate

9.2.1 Uniform S

Just as with the other datasets, we begin by fitting the uniform model to the data. We have used the function `estSemi()` to maximize the log-likelihood function from equation (6.10). The results from the estimation procedure are shown in table 9.9. The table presents the estimated parameter values and corresponding standard deviations, as well as lower and upper bounds for the 95% standard positive confidence intervals calculated from the Hessian matrix. For the complete output from R, see appendix E.5.1.

TABLE 9.9: Maximum likelihood estimates of the parameters in the model with uniform S for the bone marrow data. In addition, standard deviations from the Hessian matrix and 95% standard positive confidence intervals are included

Parameter	Estimate	Standard deviation	Lower bound	Upper bound
α	0.0140	0.0064	0.0057	0.0343
β	0.5642	0.0466	0.4799	0.6633
c	0.2767	0.1310	0.1095	0.6996
A	0.6576	0.3352	0.2422	1.7859

Since this is the first model we fit to this dataset, we do not really have anything to compare the parameter estimates to. Overall, the parameter estimates seem to be quite small, a tendency that we have seen for the uniform model also for the other datasets. None of the estimated standard deviations are particularly large compared to the values of the parameters.

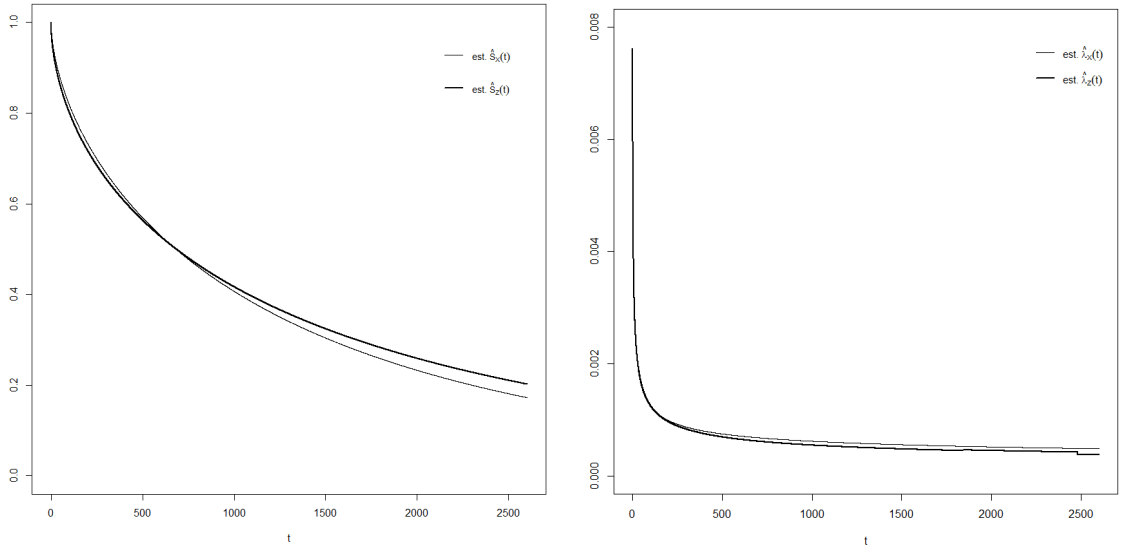

FIGURE 9.12: Estimated marginal survival functions $\hat{S}_Z(t)$ and $\hat{S}_X(t)$ (left) and hazard functions $\hat{\lambda}_Z(t)$ and $\hat{\lambda}_X(t)$ (right) with uniform S

As in the simulation studies, we have plotted the estimated marginal survival functions $\hat{S}_Z(t)$ and $\hat{S}_X(t)$ using equations (6.14) and (5.3). The curves are shown together to the left in figure 9.12. We have also plotted the estimated marginal hazard functions $\hat{\lambda}_Z(t)$ and $\hat{\lambda}_X(t)$ from equations (6.15) and (6.16). These are displayed together to the right, also in figure 9.12, in thick and thin lines respectively.

9.2.2 Exponential S

Next, we fit the exponential model to the data. The resulting parameter estimates from the maximum likelihood procedure are shown in table 9.10. In addition, the table contains the estimated standard deviations and lower and upper bounds for the 95% standard positive confidence intervals calculated from the Hessian matrix. The complete output form `estSemi()` can be found in appendix E.5.2.

TABLE 9.10: Maximum likelihood estimates of the parameters in the model with exponential S for the bone marrow data. In addition, standard deviations from the Hessian matrix and 95% standard positive confidence intervals are included

Parameter	Estimate	Standard deviation	Lower bound	Upper bound
α	0.0134	0.0057	0.0058	0.0310
β	0.5695	0.0459	0.4863	0.6669
c	0.2727	0.1179	0.1168	0.6364
λ_S	1.8243	0.8866	0.7037	4.7294

From the results in table 9.10 we can notice that the parameter estimates for α, β and c are very close to those in the uniform model. This is not surprising, since we have seen this happen both with the VHF-data and the carcinoma data. Also the standard deviations in table 9.10 are of the same size as those in the uniform model.

We continue by considering the net quantities. We plot the functions from (6.14) and (5.3) with the parameter estimates from table 9.10, i.e. the estimated marginal survival functions $\hat{S}_Z(t)$ and $\hat{S}_X(t)$. They are shown together to the left in figure 9.13. To the right in the same figure we plot the estimated marginal hazard functions $\hat{\lambda}_Z(t)$ and $\hat{\lambda}_X(t)$ from (6.15) and (6.16). The functions for Z are plotted in thick lines, while the functions for X are plotted in thin lines.

9.2.3 Gamma distributed S

We now move on to consider the gamma model. By inserting the $Ga(\alpha_S, \beta_S)$ distribution as $f_S(s)$ in the log-likelihood function (6.10) and maximizing it numerically, we get the results shown in table 9.11. The table displays the estimated parameter

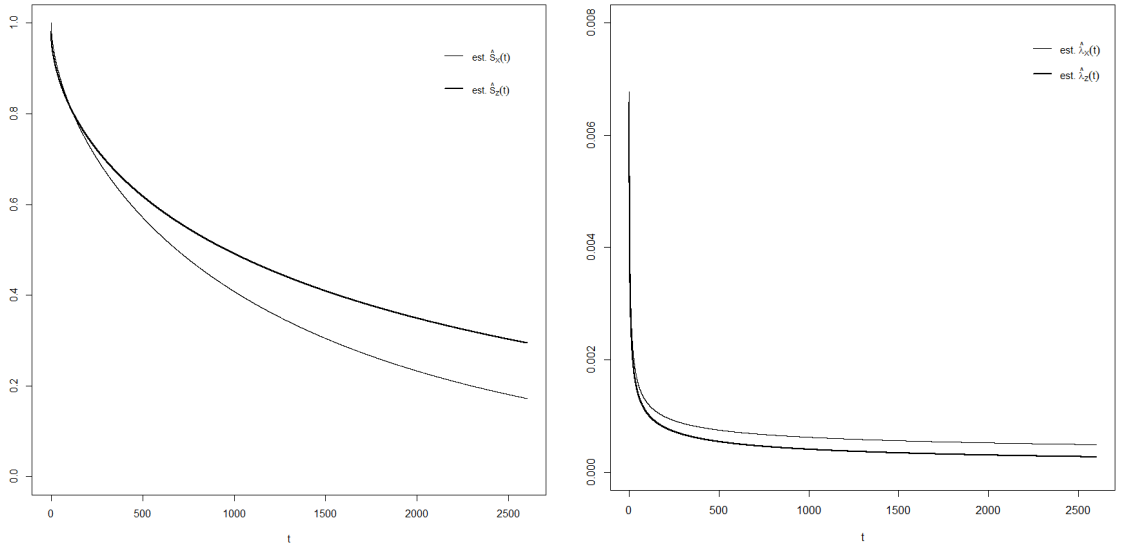

FIGURE 9.13: Estimated marginal survival functions $\hat{S}_Z(t)$ and $\hat{S}_X(t)$ (left) and hazard functions $\hat{\lambda}_Z(t)$ and $\hat{\lambda}_X(t)$ (right) with exponential S

values and corresponding standard deviations, as well as lower and upper limits for the 95% standard positive confidence intervals calculated from the Hessian matrix. The complete output from R is given in appendix E.5.3.

TABLE 9.11: Maximum likelihood estimates of the parameters in the model with gamma S for the bone marrow data. In addition, standard deviations from the Hessian matrix and 95% standard positive confidence intervals are included

Parameter	Estimate	Standard deviation	Lower bound	Upper bound
α	4.8864	0.4992	3.9997	5.9697
β	0.1386	0.0125	0.1162	0.1653
c	12.0845	0.3303	11.4540	12.7496
α_S	576.9046	17.6879	543.2574	612.6357
β_S	47.6163	1.2183	45.2874	50.0651

Also here, we can see the same tendency as we have seen both in the VHF-data analysis and in the carcinoma data analysis: the parameter estimates in the gamma model are very different from those in the uniform and exponential models. At a first glance, $\hat{\alpha}_S$ and $\hat{\beta}_S$ may seem very large. However, they make $\widehat{\mathrm{E}[S]} \approx 12.12$ and $\widehat{\mathrm{Var}[S]} \approx 0.25$, which does not seem implausible. The estimated standard deviations for all of the parameters are relatively small.

Again we have plotted the estimated marginal survival functions $\hat{S}_Z(t)$ and $\hat{S}_X(t)$ using (6.14) and (5.3). They are displayed to the left in figure 9.14 in thick and thin lines respectively. As we can see, they are almost identical. We have also plotted

the estimated marginal hazard functions $\hat{\lambda}_Z(t)$ (6.15) and $\hat{\lambda}_X(t)$ (6.16) to the right in the same figure.

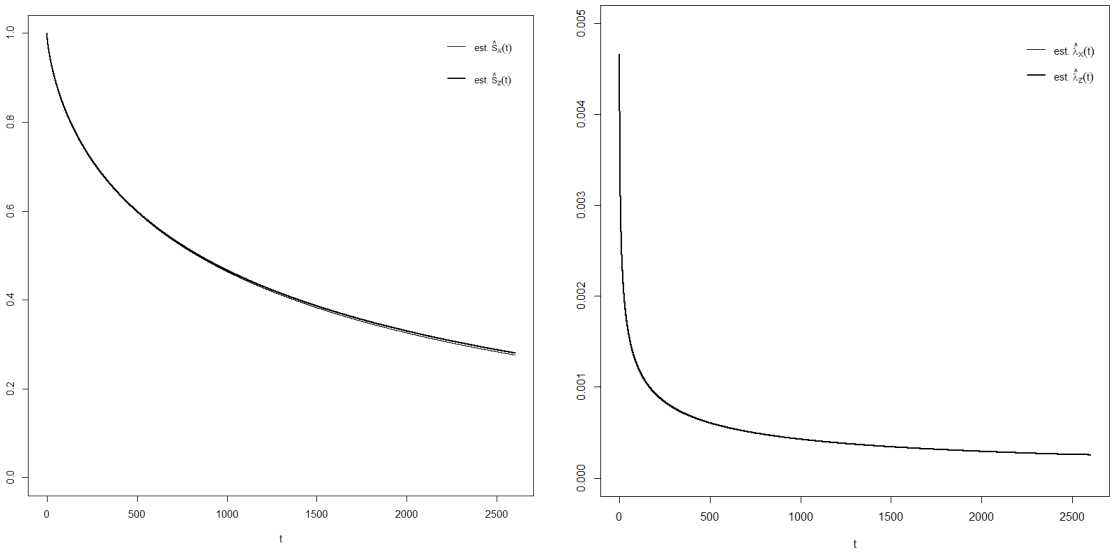

FIGURE 9.14: Estimated marginal survival functions $\hat{S}_Z(t)$ and $\hat{S}_X(t)$ (left) and hazard functions $\hat{\lambda}_Z(t)$ and $\hat{\lambda}_X(t)$ (right) with gamma distributed S

9.2.4 Lognormal S

Finally, we fit the lognormal model to the data. The results from the maximum likelihood estimation procedure are shown in table 9.12. In addition to the actual parameter estimates, the table presents the estimated standard deviations and lower and upper bounds for the 95% standard positive confidence intervals calculated from the Hessian matrix. The complete output from the `estSemi()` function can be found in appendix E.5.4.

TABLE 9.12: Maximum likelihood estimates of the parameters in the model with lognormal S for the bone marrow data. In addition, standard deviations from the Hessian matrix and 95% standard positive confidence intervals are included

Parameter	Estimate	Standard deviation	Lower bound	Upper bound
α	4.9248	0.4273	4.1546	5.8377
β	0.1374	0.0126	0.1148	0.1645
c	12.0893	0.0570	11.9781	12.2016
μ_S	2.4940	0.0073	2.4797	2.5084
σ_S	0.0401	0.0120	0.0223	0.0723

Not surprisingly, these estimates of α, β and c are not that far from those obtained in the gamma model. The estimated standard deviations are however a little smaller

(relative to the size of the parameter estimates). This is also consistent with what we have noticed earlier in the thesis.

The estimated marginal survival functions $\hat{S}_Z(t)$ and $\hat{S}_X(t)$ from (6.14) and (5.3) are plotted together to the left in figure 9.15. We have also plotted the estimated marginal hazard functions $\hat{\lambda}_Z(t)$ and $\hat{\lambda}_X(t)$ using (6.15) and (6.16) together to the right in the same figure. Also here it is very difficult to tell the difference between the functions for Z (thick lines) and the function for X (thin lines).

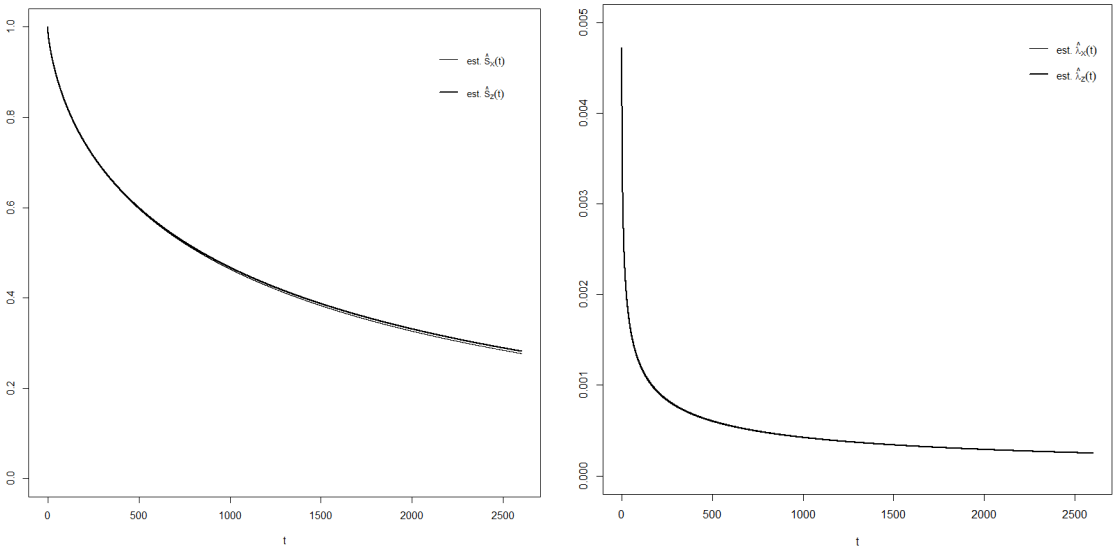

FIGURE 9.15: Estimated marginal survival functions $\hat{S}_Z(t)$ and $\hat{S}_X(t)$ (left) and hazard functions $\hat{\lambda}_Z(t)$ and $\hat{\lambda}_X(t)$ (right) with lognormal S

9.2.5 Comparison of model fits - bone marrow transplant data

We will now compare the fits of the four random S gamma process models to each other. As before, we can use the maximum log-likelihood values to get an indication of which models fit the bone marrow data the best. Table 9.13 contains the maximum log-likelihood values from each model .

TABLE 9.13: Comparison of maximum log-likelihood values from the random S models for the bone marrow data

Model	$\max \log L$
uniform	-1000.186
exponential	-1002.776
gamma	-955.5955
lognormal	-955.6274

From the values presented in table 9.13 we can see that the model with the largest log-likelihood value is the gamma one. The lognormal model has a value that is just

barely a little lower, while the values in the uniform and the exponential models are considerably lower. This implies that the uniform and exponential models do not fit the bone marrow transplant data as well as the gamma and lognormal models. The difference between the models can be further studied graphically by plotting the estimated $f_S(s)$ distributions. This is done in figure 9.16. In this case, the distribution of S describes the tendency of experiencing a relapse. From the plot we can see that the lognormal and gamma models (in green and blue, respectively) have the main part of their densities far from where the exponential and uniform models (red and black) have theirs. We can also notice that the variance in the gamma and lognormal distributions is quite small.

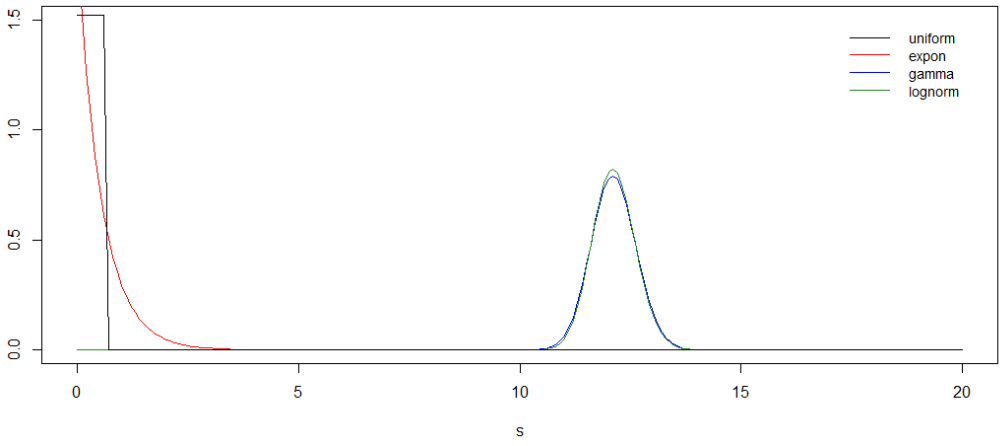

FIGURE 9.16: Comparison of $\hat{f}_S(s)$ in the random S models for the bone marrow data

The shape of the estimated gamma and lognormal distributions suggests that a normal distribution should fit the data quite well. This was also the case for the VHF data we saw in figure 8.1. As we can recall from the discussion on suitable distributions for S in chapter 6, we did not select the normal distribution mainly because it may take on values < 0. A truncated normal distribution, like the one used by Skogsrud and Lindqvist, was also discussed, but we chose to test the gamma and the lognormal distributions instead, since they more naturally suited our requirements. However, seeing the shape of the estimated distribution in figure 9.16, it would be interesting to let $f_S(s) \sim N(\mu_S, \sigma_S)$ after all. For the gamma and lognormal distributions, the shapes in figure 9.16 are somewhat atypical, as they more usually are skewed to some degree. The results obtained using the normal distribution as $f_S(s)$ and maximizing the log-likelihood function in (6.10) are displayed in table 9.14. The complete results from R are included in appendix E.5.5.

As expected, the results in table 9.14 are very similar to those we got using the gamma and lognormal ones. The values for α, β and c are very close to what they

TABLE 9.14: Maximum likelihood estimates of the parameters in the model with normal S for the bone marrow data. In addition, standard deviations from the Hessian matrix and 95% standard positive confidence intervals are included

Parameter	Estimate	Standard deviation	Lower bound	Upper bound
α	4.9218	0.6373	3.8187	6.3436
β	0.1382	0.0061	0.1267	0.1508
c	12.1376	1.3475	9.7641	15.0881
μ_S	12.1620	1.3453	9.7914	15.1066
σ_S	0.5000	0.0799	0.3656	0.6839

were there, and the estimated parameters in $f_S(s)$ result in a distribution of S that is almost identical to the gamma and lognormal curves in figure 9.16. We can furthermore note that the maximum log-likelihood value with normally distributed S is -955.5186. This is better than what we got from the gamma and lognormal models in table 9.13, but still quite close to those values.

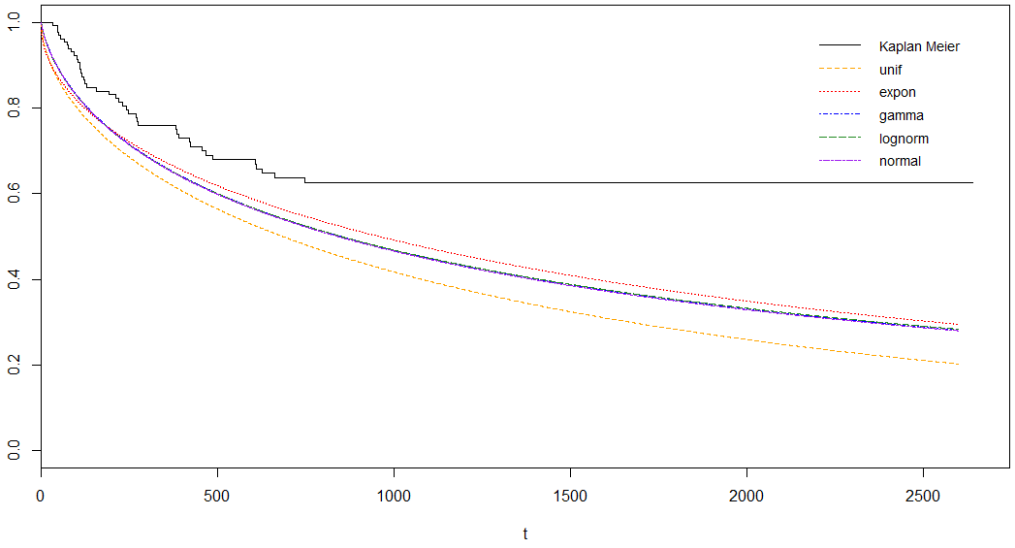

FIGURE 9.17: $\hat{S}_Z(t)$ in each of the five random S models for the bone marrow transplant data compared to the Kaplan-Meier estimate

The estimated marginal survivor functions for the time to relapse, $\hat{S}_Z(t)$, from all five models are plotted together in figure 9.17. Like for the carcinoma data, the uniform curve (in orange) seems to decrease faster than the other curves. The curves of the gamma (blue), lognormal (green) and normal (purple) models are almost identical and difficult to tell apart. The exponential curve (in red) is slightly above the other parametrically estimated curves. The Kaplan-Meier curve (in black) is far above all of the parametric curves. It is obviously not a good estimate of the marginal

survival function for the time to relapse, as it does not take into account the strong dependency between the time to relapse and the time to death.

One should also perhaps discuss what we actually mean by the marginal survival function for the time to relapse. It corresponds to a situation where death from any other cause than leukaemia is preventable. Thus, the situation is not realistic, but in order to evaluate the actual efficacy of the treatment, we need to estimate the disease-free survival time in the absence of other failure types than relapse [12].

We can furthermore make plots of the parametric and non-parametric estimates of the crude quantities $F_Z^*(t)$ and $\Lambda_Z^*(t)$. As before, we make the respective non-parametric estimates by using the expressions in equations (4.4) and (4.3), while the corresponding parametric estimates are found from equation (6.12) and from integrating equation (6.13). We make these estimates for the five random S models and plot the results in in figures 9.18 and 9.19.

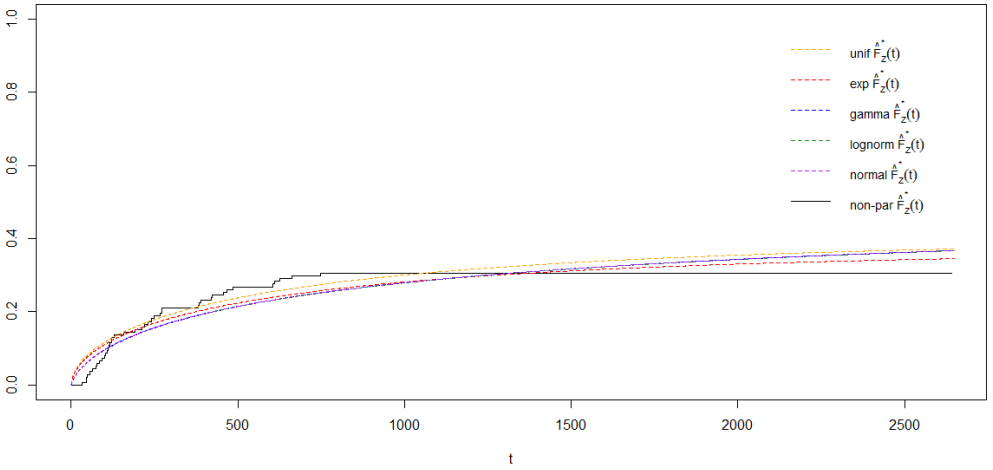

FIGURE 9.18: Parametric and non-parametric estimates of $F_Z^*(t)$ for the bone marrow transplant data

In general, the parametric curves seem to fit fairly well to the non-parametric curves. We can see that the non-parametric curves for both $F_Z^*(t)$ and $\Lambda_Z^*(t)$ flatten out relatively early, since the majority of the observations are for smaller t-values. Hence, it will be difficult for our parametric estimates to match the non-parametric ones very closely. This also makes it difficult to evaluate which of the curves that matches the non-parametric functions the best. We can further notice that the curves of the gamma (blue), lognormal (green) and normal (purple) models lie on top of each other and follow the same trajectory.

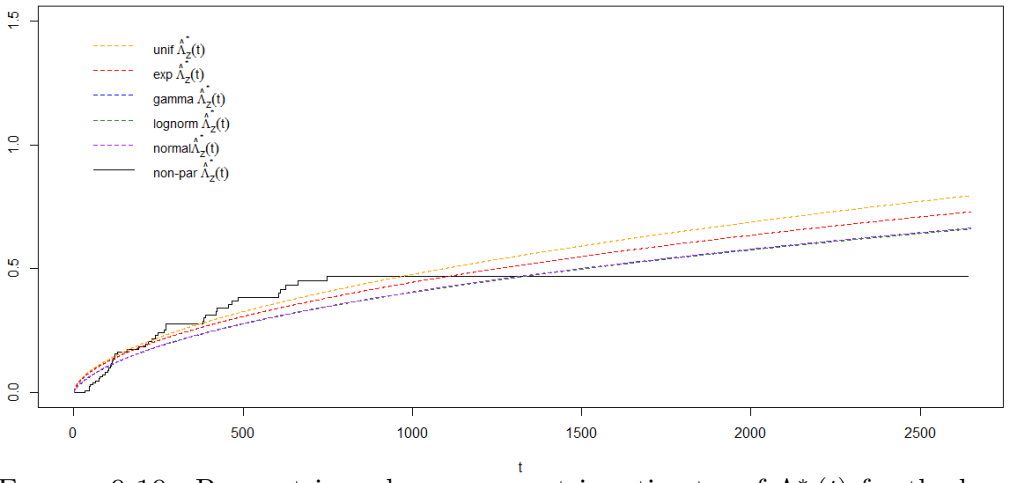

FIGURE 9.19: Parametric and non-parametric estimates of $\Lambda^*_Z(t)$ for the bone marrow transplant data

Also for these data it might be of interest to compare the estimates of $F_S(c)$ made with the different models, even though we do not have any non-parametric estimate to compare them to. These are shown in table 9.15.

TABLE 9.15: Comparison estimates of $F_S(c)$ in the random S models for the bone marrow transplant data

Model	$P(S < c)$
uniform	0.4208
exponential	0.3919
gamma	0.4809
lognormal	0.4831
normal	0.4805

In contrast to earlier, the estimates of $F_S(c)$ now seem to be quite different for the respective models. Mainly, the estimates made with the uniform and exponential models are considerably smaller than the ones made with the gamma, lognormal and normal models.

Overall, we suspect that the gamma, lognormal and normal models provide an overall better fit to the data than the exponential or uniform model, considering the high maximum log-likelihood values. We therefore choose to compare these three estimates more in detail to the one from the article by Fine, Jiang and Chappell [12] by drawing them in the same plot. This is done to the left of figure 9.20. Still, it is impossible to tell the curves of the gamma, lognormal and normal models from each other, so they are all represented by the blue curve. The survival function estimated by Fine, Jiang and Chappell (black) flattens out more than the curves

estimated with our gamma, lognormal or normal S models. This is probably due to that there are very few data points left when $t > 1000$. Still, our estimated function is almost always inside the estimated 95% confidence interval in the black dashed lines.

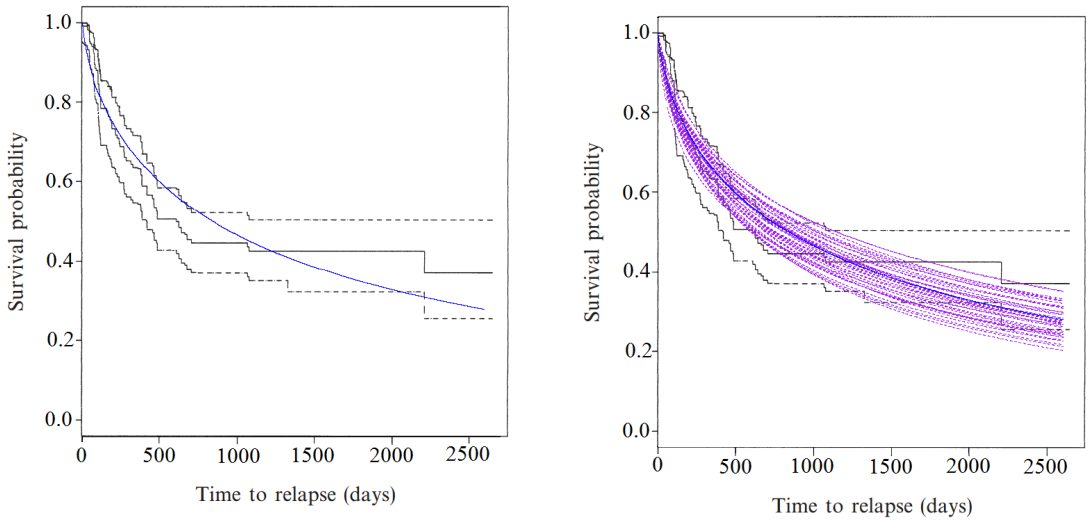

FIGURE 9.20: Estimate from the gamma, lognormal and normal models of the marginal survival function for the time to relapse compared to the estimate from [12] along with 95% confidence interval limits

To the right of figure 9.20, an indication of the variance in our estimates for the normal model is shown. These purple curves are made by drawing the estimates of $S_Z(t)$ obtained from 50 non-parametric bootstrap samples. As we can see, we get a range of curves that is approximately as wide as the estimated confidence interval by Fine, Jiang and Chappell in the black dashed lines.

In summary, the gamma, lognormal and normal models seem to suit the data almost equally well and considerably better than the uniform and exponential models. This is probably mostly because the uniform and exponential distributions have to have non-zero probabilities at t close to zero. Additionally, these two models only have one parameter, so they have limited flexibility. Out of the three best models, it seems that the normal model has a slightly better overall fit than the two others. It had the largest maximum log-likelihood value, and considering the plots of the estimated distributions $f_S(s)$, a normal distribution would intuitively fit very well. Still, to say exactly how well the normal model fits the data in general is difficult. The comparison of the parametric and non-parametric estimates for $F_Z^*(t)$ and $\Lambda_Z^*(t)$ gave us some indication that the fit is at least not completely off, but the low number of observations, in particular for $t > 1000$, made direct comparison tricky. At the same time, we saw that our estimate for the marginal survival function for the time

to relapse was consistent with that obtained by Fine, Jiang and Chappell in [12]. This may be interpreted as a confirmation that the model is quite good.

In the competing risks setting we saw that the basic model with a fixed level s performed just as well, or even better, than the gamma and lognormal models. This raises the question of whether such a model would do better also in the semi-competing risks case. However, a constant s model in the semi-competing risks setting, while possible, does not provide a satisfactory interpretation or description of reality. In the bone marrow example, a fixed level s would mean that there is a certain, constant probability q of relapsing and a probability of $1-q$ of dying before a potential relapse. This in itself might not be such a bad assumption, but if we keep in mind that in semi-competing risks we are often interested in the marginal distribution of the time to relapse, then we face a problem. We then have to choose a value $v > c$ where relapse occurs if death is not present. A random S model fits much better intuitively, letting the potential relapse occur at any time.

Chapter 10

Concluding remarks

In this final chapter, the main results of the thesis are discussed and summarized. In addition, some suggestions for future work will be presented.

10.1 Discussion and main results

In this Master's thesis we have studied a way to model both dependent competing risks and semi-competing risks by means of first passage times in a gamma process. The model we have considered is an extension of a model studied in my Master's project [35]. In both the competing risks case and the semi-competing risks case there is a non-terminal event and a terminal event. These events are considered to happen at specific points of some underlying degradation process. In the gamma process model, the time to the non-terminal event is equal to the first passage time to a stochastic level S. The time to the terminal event is represented by the first passage time to a fixed level c. We have assumed that S is independent of the gamma process, i.e. the age of the item under study. That means that we have random signs censoring. As possible distributions for the level S we have tested the uniform, the exponential, the gamma, the lognormal and (for one dataset) also the normal distribution.

In the competing risks case, our motivation or main application has specifically been the situation where there are two competing events, either failure or preventive maintenance. If $S < c$ we would get a PM, but if $S > c$ we would get a failure. The distribution of S represents when the maintenance crew is most likely to do a PM. This was also modelled by Lindqvist and Skogsrud with Wiener processes in [25]. We first applied our random S gamma process model to simulated data in order to evaluate the quality of the parameter estimates found by maximum likelihood estimation. For all four choices of $f_S(s)$ the estimation worked well, and produced estimates that were close to the true parameter values.

The model was later applied to a real dataset, the VHF data. Out of the four random S cases, the lognormal and gamma models seemed to fit the data the best. In comparison, the uniform and exponential models did not seem to fit to the data

well. The interpretation of this aligns with our intuition: the maintenance crew should check an item at a time close when it is expected to fail, not at the beginning of its lifespan. What is perhaps surprising, is that randomization of S did not seem to yield an improvement of the model fit, even though the random S model is more advanced. In the data analysis of the VHF data we saw that the basic model with constant level S resulted in a maximum log-likelihood value that was slightly higher than those from the gamma and lognormal models. This indicates that there are not that big individual differences with regards to the maintenance policy of the items in the VHF-data.

These results are consistent with what we observed in the simulation studies. For the datasets that we simulated with relatively narrow gamma and lognormal distributions for S, the basic model also seemed to fit the data quite well. Even though the differences between the fit of the gamma, lognormal and basic models were very small, the basic model was in our case preferred because of its simplicity. In contrast, in Skogsrud's thesis the conclusion for the same dataset was that randomization of S was beneficial, but that was with models based on Wiener processes. It is the shape function $v(t) = \alpha t^{\beta}$ in the gamma process that seems to provide a great deal of flexibility to our model. The gamma process is furthermore better suited to model gradual accumulation of damage over time because of its non-negative increments.

The gamma process model was applied to semi-competing risks data as well. Semi-competing risks data frequently arise in medical research and clinical trials. Typically the data consist of times to a non-terminal event like recurrence of some disease, and times to a terminal event, like death. Thus, we are not considering the events PM or failure any more, and the interpretation of the model is obviously not the same as in the competing risks case. Semi-competing risks data have mostly been analysed through copula models in the past. We noticed that the semi-competing risks data also seem to fit Cooke's criteria for random signs censoring when the conditional sub-survival curves are plotted as if the data were ordinary competing risks data. Therefore it should not be a problem to apply our gamma process model to semi-competing risks. In that way, we model the dependency of the time to the non-terminal event and the time to the terminal event through the gamma process instead of a copula. The tendency of the non-terminal event is modelled through the distribution of S.

In the same manner as in the competing risks case, we first tested the model on simulated data. Also here, the method of maximum likelihood estimation worked

well for all of the four distributions of S. We then applied the model to two real datasets. The first was the carcinoma dataset, and the second was the bone marrow transplant data. For both of these datasets, the gamma and lognormal models seemed to have the best fit to the data. The uniform and exponential models performed poorly on the bone marrow transplant data, similar to what we saw in the competing risks case with the VHF data. For the carcinoma data however, they actually did not seem to be that bad. Looking more closely at the estimated distributions for S, we saw that the expected values of S were closer together than what they were for the other datasets, and that the gamma and lognormal distributions had quite large variances. Recall that the non-terminal event in this case was to be taken off a certain drug due to disease progression. It seems plausible that this can happen quite early and at a wide range of times.

For the bone marrow transplant data, the estimated gamma and lognormal distributions looked strikingly like an ordinary normal distribution. For this reason, we chose to fit the gamma process model with normally distributed S to the data as well. The resulting parameter estimates were very similar to those obtained by the gamma and lognormal models, but the maximum log-likelihood value was a little higher for the normal model. Visually, there were almost no observable differences in the estimates made with the three models. It is however difficult to say exactly how well the models fit the data overall. It was hard to directly compare parametric estimates of the crude quantities to non-parametric ones, because for a large part of the range of t there were very few observations. Still, the fit seemed to be relatively good. More convincing is the fact that our estimate for the marginal survival function for the time to relapse looked like it was consistent with that obtained by Fine, Jiang and Chappell in [12].

Since we only have tested the gamma process model on a few datasets, we cannot say anything definite about how good the model is. Moreover, we have judged the fit of the models mostly by looking at plots. There are more formal ways to measure goodness-of-fit. However, the results we have seen are promising. In general, they imply that the gamma process model with random level S is a good way to model both competing and semi-competing risks. The main strength of the gamma process model is its flexibility, mainly due to the power-law shape function, but also the possibility of choosing different probability distributions for S. For most purposes we would recommend using a flexible distribution for S with (at least) two parameters, like the gamma, lognormal or normal distributions. We have seen in the simulation study that for data that actually stem from a model where S is uniformly distributed, the lognormal model is still very good. Of course, to find

this model useful one has to accept the concept of random signs censoring. Like all other approaches, random signs censoring is based on non-testable assumptions.

10.2 Further work

In this thesis we chose to extend the basic gamma process model from the project thesis by letting the level S be random. Here, we present and discuss some other possible extensions of the basic gamma process model.

10.2.1 Random level c

As mentioned in chapter 6, we could let a different parameter than s be random. A possibility that would flip the situation around, is to let s be fixed while C is random. This problem would be very similar to the one we have considered with random S, but the model interpretation would be different. For instance, if we look at the competing risks setting of PM vs. failure, the individual heterogeneity would no longer be with respect to different maintenance policies. The situation now would be that C can range from 0 to ∞, and if it is below s we observe a failure, while if it is above s we observe a PM. Thus, the distribution of C would describe at what level of degradation the item in question is most likely to fail.

Opposite random signs?
As previously explained, Z is a random signs censoring of X if the event $Z < X$ is independent of X. This was clearly fulfilled in the case with random S as $P(Z < X) = P(S < c)$ and S was independent of the process. Now, with random C instead, Z will no longer be a random signs censoring of X, since the event $X < Z$ is dependent of X. But in this case we may choose to consider X to be a random signs censoring of Z instead. Then it would be the marginal distribution of Z that is identifiable. That would mean that the conditional sub-survival curve of Z needs to dominate that of X.

It is more difficult to find realistic applications of this model, especially within a reliability context. One example (provided by Cooke [7]) is from a medical cohort study. Cooke suggests that this may be a suitable model if we let Z denote the time from a patient enters a study till the study ends. All of the patients have been treated for some life-threatening disease. The study will end at a predetermined time that is the same for all patients under observation, but they will all have entered the study at different times. Let X denote the time till death for those patients that died. Now, it is plausible that whether the patients respond to the

treatment (and thus do not die during the course of the study) is independent of the time they entered the study.

10.2.2 Random scale

A different way of incorporating a random effect into the model, is to replace the scale parameter u by $w \cdot u$ where w is a random effect. Actually, this is not that different from letting one of the thresholds c or s be random, since u always appears together with these thresholds in the likelihood function as the product $u \cdot c$ or $u \cdot s$. The difference is that u appears in both $f_Z^*(z)$ and $f_X^*(x)$ not just one of them as c or s does. Thus, the speed of the overall process is the random quantity here. Letting the scale parameter u in a gamma process be a random variable is also done by Lawless and Crowder in [21]. In their example, they let w be gamma distributed, $w \sim Ga(\gamma^{-1}, \delta)$. This implies that w has mean $\frac{\delta}{\gamma}$ and variance $\frac{\delta}{\gamma^2}$. We will in the following denote the gamma process by $Y(t)$, and follow the same steps as in [21]. Then, the conditional density $f(y|w) \sim Ga(y; wu, v(t))$, and it follows that

$$f(y) = \int_0^\infty f(y|w)f(w)dw = B(v(t), \delta)^{-1}(\gamma u^{-1})^\delta \frac{y^{v(t)-1}}{(y + \gamma u^{-1})^{v(t)+\delta}}$$

where $B(v(t), \delta) = \frac{\Gamma(v(t))\Gamma(\delta)}{\Gamma(\delta+v(t))}$ is the beta function.

Furthermore, one can note that the mean and variance of $Y(t)$ is given by

$$\mathrm{E}[Y(t)] = \frac{\gamma v(t)}{u(\delta - 1)} \quad \text{for} \quad \delta > 1$$

$$\mathrm{Var}[Y(t)] = \frac{\gamma^2 v(t)(v(t) + \delta - 1)}{u^2(\delta - 1)^2(\delta - 2)} \quad \text{for} \quad \delta > 2$$

We can now make a note about parameterization: γ and u always appear together as $\gamma \cdot u^{-1}$. We may therefore choose to just write u, and in the case of no covariates, let $u = 1$ as in the basic model. Alternatively, one could let $\gamma = \delta$ and include the constant term γ in u. In this case w will have mean 1 and variance δ^{-1}. This is often what is done in gamma frailty models, and we choose to do it here as well, so

$$f(y) = B(v(t), \delta)^{-1}(\delta u^{-1})^\delta \frac{y^{v(t)-1}}{(y + \delta u^{-1})^{v(t)+\delta}}$$

An interesting side-note: In [29] Paroissin and Salami also suggest a model with random effects and refer to this paper by Lawless and Crowder, but they let u itself

be gamma distributed, and never introduce w. Then, u disappears from the PDF, while both γ and δ are kept.

Now, we are interested in the distribution of the first passage time to a certain level y^F:

$$F_T(t) = P(T \leq t) = P(y \geq y^F) = 1 - F\left(\frac{uy^F}{v(t)}\right)$$

(At this point we stop following the steps in [21]). If we now denote $\frac{uy^F}{v(t)}$ by x, F is the distribution

$$F_{2v(t),2\delta} = I_{\frac{2v(t)x}{2v(t)x+2\delta}}\left(\frac{2v(t)}{2}, \frac{2\delta}{2}\right) = I_{\frac{v(t)x}{v(t)x+\delta}}\left(v(t), \delta\right)$$

Here, I is the regularized incomplete beta function, so that:

$$I_{\frac{v(t)x}{v(t)x+\delta}}\left(v(t), \delta\right) = \frac{B\left(\frac{v(t)x}{v(t)x+\delta}, v(t), \delta\right)}{B\left(v(t), \delta\right)}$$

where $B(z, a, b) = \int_0^z t^{a-1}(1-t)^{b-1}dt$ is the incomplete beta function. Inserting back for x we get

$$I_{\frac{uy^F}{uy^F+\delta}}\left(v(t), \delta\right) = \frac{B\left(\frac{uy^F}{uy^F+\delta}, v(t), \delta\right)}{B\left(v(t), \delta\right)}$$

Again we can see that the scale parameter u only appears together with the random threshold y^F. We may therefore choose to set $u = 1$, and use

$$F_T(t) = 1 - \frac{B\left(\frac{y^F}{y^F+\delta}, v(t), \delta\right)}{B\left(v(t), \delta\right)}$$

(This notation is maybe a little confusing, since we are saying that $u = 1$ when we are considering the case "random scale parameter u". However, we have to recall that we introduced the parameter w on which the randomness was placed...)

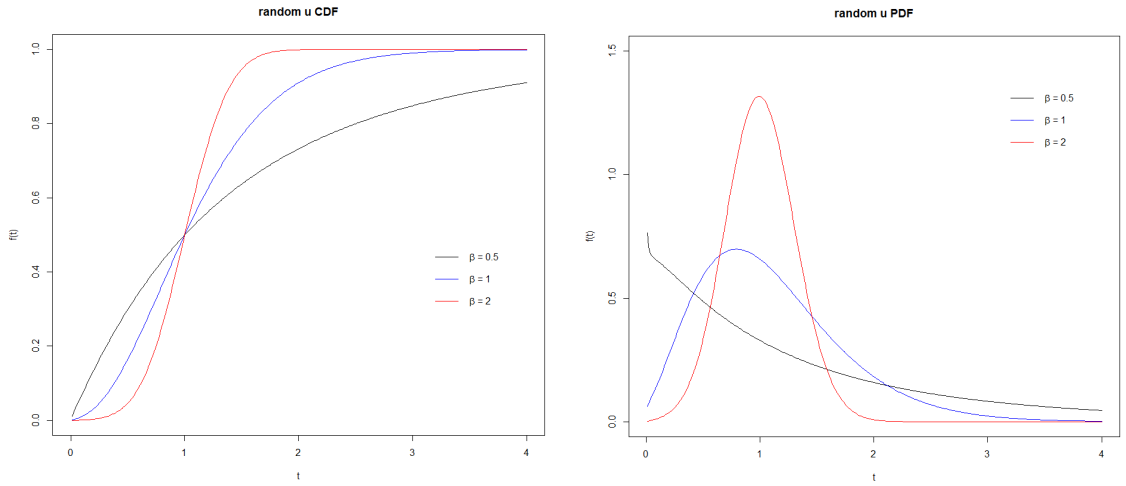

FIGURE 10.1: CDF (left) and PDF (right) for the first passage time with gamma distributed U for different values of β

To find the PDF of the first passage time, one may for instance use Wolfram Alpha [43] to get

$$\begin{aligned} f_T(t) = \frac{\partial}{\partial t}\left(1 - I_{\frac{y^F}{y^F+\delta}}(v(t),\delta)\right) &= -v'(t)\log\left(\frac{y^F}{y^F+\delta}\right) I_{\frac{y^F}{y^F+\delta}}(v(t),\delta) \\ &+ v'(t)\left(\frac{y^F}{y^F+\delta}\right)^{v(t)} \frac{\Gamma(v(t))^2}{B(v(t),\delta)} \cdot {}_3\tilde{F}_2\left(v(t), v(t), 1-\delta; v(t)+1, v(t)+1; \frac{y^F}{y^F+\delta}\right) \\ &+ v'(t)\left[\Psi(v(t)) - \Psi(v(t)+\delta)\right] I_{\frac{y^F}{y^F+\delta}}(v(t),\delta) \end{aligned}$$

Here, ${}_p\tilde{F}_q(a_1,...,a_p;b_1,...,b_q;z)$ denotes the *regularized* generalized hypergeometric function $= \frac{{}_pF_q}{\Gamma(b_1)\cdots\Gamma(b_q)}$.

A few examples of how the CDF and PDF for the first passage time may look is shown in figure 10.1. From here one can easily find the likelihood function for the model.

10.2.3 Covariates

It might also be of interest to study the effect of covariates on our model. There are several ways to incorporate covariates into the gamma process model, as there are more than one parameter that may be modelled by a covariate-dependent expression. Aalen and Gjessing [1] distinguish between two types of covariates: 1) those that only represent measures of how far the underlying process has advanced (e.g. threshold level) and 2) those that have causal influences on the development (e.g. drift parameter in a Wiener process).

One option, that has been studied by Lawless and Crowder in [21], is to let the scale parameter u depend on a vector of covariates $\mathbf{x}$, $u = u(\mathbf{x})$. Alternatively, Bagdonavicius and Nikulin [4] include covariates in their model via an accelerated life test model, replacing $v(t)$ by $v(te^{\mathbf{x}^T\boldsymbol{\rho}})$. This is however computationally more difficult. By letting u be a function of covariates, the covariates only affect the scaling of the degradation process, not the shape function. A natural choice might be $u(\mathbf{x}) = \exp(\boldsymbol{\rho}'\mathbf{x})$ where $\boldsymbol{\rho}$ is a vector of regression coefficients.

Putting the covariates on u is effectively the same as putting them on the critical threshold c or s. As we also mentioned in the discussion of random effects, u and c or s always appear together as the product $u \cdot c$ or $u \cdot s$ in the first passage time distribution. The scale parameter u is the same in both $f_X^*(x_i)$ and $f_Z^*(z_j)$, while the critical threshold is c in $f_X^*(x_i)$ and s in $f_Z^*(z_j)$. Therefore, the covariates have quite different meanings depending on which parameter they are assigned to.

For instance, we may first consider including covariates in the parameter c. If we let 0 be the starting point of the degradation process for all items, we may use covariates on c to vary the level where the failures occur. For example it might be natural to assume that a smoking patient with a heart disease may reach the critical level faster than one who doesn't smoke, and that a patient that is regularly working out will reach the level slower than an inactive patient.

Covariates on the level s on the other hand, provide information about the maintenance policy of the item. For example in medicine this level may describe the level where a disease is diagnosed and treatment is started. Then, patients who are examined more often are more likely to be diagnosed at an early stage and have a lower level s than those that only rarely are examined.

Choosing to let u depend on covariates has yet another implication. If the levels c and s are both fixed, as well as the probability q, the covariates in u say something about the overall speed of the process, i.e. both the time till it reaches s and till it reaches c. Covariates may also enter the model through the parameter q. This will have similar effect as putting them on s. One could for example say that a person who is examined more often has a greater probability q of detecting a signal that something is wrong than others.

10.2.4 Other options

There is a great number of other possibilities that may be explored. Some of them have already been mentioned in either this thesis or the project thesis.

Of course other distributions for S could be considered. An idea related to this, is to let the lower limit in the uniform distribution be different from 0. This would probably yield better results for the uniform model. Still, it is hard to imagine that this would be better than using for instance a lognormal distribution for most cases.

Another suggestion is to consider models with more than two competing risks. One could imagine a situation where there are k competing risks or events that could happen before the item potentially failed, not just PM. This would however not be a case of random signs censoring any more.

Another possibility, which was also mentioned in the project thesis, is to experiment with different shape functions for $v(t)$. An alternative is for instance $v(t) = e^{\alpha+\beta t}$.

One could also let the parameters α or β in the shape function $v(t)$ be random quantities. This is more computationally difficult and does not provide a similar intuitive interpretation as letting a threshold parameter be random.

Finally, another field it would be interesting to explore with the gamma process models is that of maintenance optimization. There are many examples of gamma process models used in maintenance optimization in [38]. In our model one could for instance optimize the level of preventive maintenance, S. If one simply lets failure be much more expensive than a PM, then of course the closer S is to c the better. A more advanced problem is to let the cost of PM increase with S so that an S very close to c not necessarily is the best solution.

Bibliography

[1] Aalen, O. O. and Gjessing, H. K. (2001). Understanding the shape of the hazard rate: A process point of view. *Statistical Science*, 16(1):1–22.

[2] Abramowitz, M. and Stegun, I. A. (1972). *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. National Bureau of Standards, Applied Mathematics Series – 55.

[3] Andersen, P. K. and Keiding, N. (2012). Interpretability and importance of functionals in competing risks and multistate models. *Statistics in medicine*, 31:1074–1088.

[4] Bagdonavicius, V. and Nikulin, M. (2001). Estimation in degradation models with explanatory variables. *Lifetime Data Analysis*, 7:85–103.

[5] Casella, G. and Berger, R. (2002). *Statistical Inference*. Brooks/Cole, Cengage Learning, 2nd edition.

[6] Clayton, D. G. (1978). A model for association in bivariate life tables and its application in epidemiological studies of familial tendency in chronic disease incidence. *Biometrika*, 65:141–151.

[7] Cooke, R. M. (1993). The total time on test statistic and age-dependent censoring. *Statistics & Probability Letters*, 18:307–312.

[8] Copeland, E. A., Biggs, J. C., Thompson, J. M., Crilley, P., Szer, J., Klein, J. P., Kapoor, N., Avalos, B. R., Cunningham, I., Atkinson, K., Downs, K., Harmon, G. S., Daly, M. B., Brodsky, I., and Bulova, S. I. amd Tutschka, P. J. (1991). Treatment for acute myelocytic leukemia with allogeneic bone marrow transplantation following preparation with bucy2. *Blood*, 78:838–843.

[9] Cox, D. R. (1959). The analysis of exponentially distributed lifetimes with two types of failure. *Journal of Royal Statistical Society Series B*, 21:411–421.

[10] David, H. A. and Moeschberger, M. L. (1978). *The theory of competing risks*. Charles Griffin & Co. LTD, 1st edition.

[11] Efron, B. and Tibshirani, R. J. (1993). *An introduction to the bootstrap*. Chapman & Hall.

[12] Fine, J. P., Jiang, H., and Chappell, R. (2001). On semi-competing risks data. *Biometrika*, 88:907–919.

[13] Fix, E. and Neyman, J. (1951). A simple stochastic model of recovery, relapse, death and loss of patients. *Human Biology*, 23:205–241.

[14] Givens, G. H. and Hoeting, J. A. (2013). *Computational Statistics*. John Wiley & Sons, Inc., 2nd edition.

[15] Hankin, R. K. S. (2014). *hypergeo: The Hypergeometric Function*. R package version 1.2–9.

[16] Horrocks, J. and Thompson, M. E. (2004). Modeling event times with multiple outcomes using the wiener process with drift. *Lifetime Data Analysis*, 10:29—-49.

[17] Hsieh, J.-J., Wang, W., and Ding, A. A. (2008). Regression analysis based on semicompeting risks data. *Journal of the Royal Statistical Society series B*, 70:3–20.

[18] Jiang, H., Chappell, R. J., and Fine, J. P. (2003). Estimating the distribution of nonterminal event time in the presence of mortality or informative dropout. *Controlled Clinical Trials*, 24:135–146.

[19] Klein, J. P. and Moeschberger, M. L. (1997). *Survival Analysis: Techniques for Censored and Truncated Data*. Springer Science+Business Media, 1st edition.

[20] Lagakos, S. W. and Williams, J. S. (1978). Models for censored survival analysis: A cone class of variable-sum models. *Biometrika*, 65:181–189.

[21] Lawless, J. and Crowder, M. (2004). Covariates and random effects in a gamma process model with application to degradation and failure. *Lifetime Data Analysis*, 10:213–227.

[22] Lawless, J. F. (2003). *Statistical Models and Methods for Lifetime Data*. John Wiley & Son, Inc., 2nd edition.

[23] Lindqvist, B. H. (2008). *Competing Risks*, pages 335–341. John Wiley & Sons, Ltd.

[24] Lindqvist, B. H. (2013). On random signs censoring and clinical trials. *Journal of the Indian Statistical Association*, 51(1).

[25] Lindqvist, B. H. and Skogsrud, G. (2008). Modeling of dependent competing risks by first passage times of wiener processes. *IIE Transactions*, 41(1):72–80.

[26] Lindqvist, B. H., Støve, B., and Langseth, H. (2004). Modelling of dependence between critical failure and preventive maintenance: The repair alert model. *Journal of Statistical Planning and Inference*, 136(5):1701–1717.

[27] Mendenhall, W. and Hader, R. J. (1958). Estimation of parameters of mixed exponentially distributed failure time distributions from censored life test data. *Biometrika*, 45:504–520.

[28] Park, C. and Padgett, W. (2005). Accelerated degradation models for failure based on geometric brownian motion and gamma processes. *Lifetime Data Analysis*, 11:511–527.

[29] Paroissin, C. and Salami, A. (2014). Failure time of non homogeneous gamma process. *Communications in Statistics - Theory and Methods*, 43(15):3148–3161.

[30] Peng, L. and Fine, J. P. (2006). Regression modeling of semicompeting risks data. *Biometrics*, 63:96–108.

[31] Peng, L., Jiang, H., Chappell, R. J., and Fine, J. P. (2008). *Statistical Advances in the Biomedical Sciences: Clinical Trials, Epidemiology, Survival Analysis, and Bioinformatics*, chapter 11. An Overview of the Semi–Competing Risks Problem. John Wiley & Sons, Inc.

[32] R Core Team (2013). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

[33] Rausand, M. and Høyland, A. (2004). *System Reliability Theory - Models, Statistical methods and Applications*. Wiley, 2nd edition.

[34] Ross, S. (2010). *Introduction to probability models*. Academic Press, 10th edition.

[35] Sildnes, B. (2014). Modeling of dependent competing risks by first passage times of gamma processes. TMA4500 Industrial mathematics specialization project. Department of Mathematical Sciences, Norwegian University of Science and Technology.

[36] Skogsrud, G. (2005). Statistical modeling of dependent competing risks by means of wiener processes and the inverse gaussian distribution. Master's thesis,

Norwegian University of Science and Technology, Department of Mathematical Sciences.

[37] Tsiatis, A. (1975). A nonidentifiability aspect of the problem of competing risks. *Proceedings of National Academy of Sciences USA*, 72:20–22.

[38] van Noortwijk, J. (2009). A survey of the application of gamma processes in maintenance. *Reliability Engineering and System Safety*, 94:2–21.

[39] Varadhan, R., Xue, Q. L., and Bandeen-Roche, K. (2014). Semicompeting risks in ageing research: methods, issues and needs. *Lifetime Data Analysis*, 20:538–562.

[40] Walpole, R., Myers, R., Myers, S., and Ye, K. (2012). *Probability and Statistics for Engineers and Scientists*. Pearson, 9th edition.

[41] Weisstein, E. W. (2015). Meijer g-function. `http://mathworld.wolfram.com/MeijerG-Function.html`. [Online: accessed 04-03-2015].

[42] Wolfram Alpha, L. L. C. (2015a). `http://www.wolframalpha.com/input/?i=int+d%2Fdt%28Gamma%28v%28t%29%2Cs%29%2FGamma%28v%28t%29%29%29ds`. [Online: accessed 04-03-2015].

[43] Wolfram Alpha, L. L. C. (2015b). `http://www.wolframalpha.com/input/?i=d%2Fdt+%281+-+Beta%28y%2F%28y%2Bdelta%29%2Cv%28t%29%2Cdelta%29%2FBeta%28v%28t%29%2Cdelta%29%29`. [Online: accessed 14-03-2015].

Appendix A

Basic theory

In this appendix, some important basic theory in statistics is included. This is theory that is assumed to be known to the reader. The appendix was written for the project thesis [35].

A.1 Probability distributions

In this section the probability distributions that we will use in the thesis are presented, along with some of their key properties. The theory is in large part selected from [40] unless stated otherwise.

A.1.1 The (continuous) uniform distribution

The probability density function of the uniform distribution is defined as

$$f(x) = \frac{1}{B-A} \quad \text{for} \quad A \leq x \leq B$$

The density function of the continuous uniform distribution is constant in the closed interval $[A, B]$. If $A = 0$ and $B = 1$ the density is called the standard uniform distribution

$$f(x) = \begin{cases} 1 & \text{if } 0 \leq x \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

A.1.2 The normal distribution

The normal distribution is a very central probability distribution in the field of statistics. It is sometimes also called the Gaussian distribution. The probability density function of the normal distribution with mean μ and variance σ^2 is given by

$$f(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(x-\mu)^2/(2\sigma^2)}, \quad -\infty < x < \infty \tag{A.1}$$

The parameters μ and σ^2 are also described as the location and scale parameter, respectively. The usual notation to say that a random variable X is normally

distributed with parameters μ and σ^2 is $X \sim N(\mu, \sigma^2)$. If $\mu = 0$ and $\sigma^2 = 1$ we get what is called the standard normal distribution

$$f(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}, \quad -\infty < x < \infty \tag{A.2}$$

A.1.3 The exponential distribution

The probability density function of the exponential distribution for a random variable X is given as

$$f(x; \lambda) = \begin{cases} \lambda e^{-\lambda x} & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{A.3}$$

λ is often called the rate parameter of the distribution.

A.1.4 The gamma distribution

In the following, we will use the same notation as in [38]. The probability density function of a gamma distributed random variable X with shape parameter $v > 0$ and scale parameter $u > 0$ is

$$Ga(x; v, u) = \frac{u^v}{\Gamma(v)} x^{v-1} \exp\{-ux\} I_{(0,\infty)}(x) \tag{A.4}$$

Here, $I_{(0,\infty)}(x) = 1$ if $x \in (0, \infty)$ and $I_{(0,\infty)}(x) = 0$ if $x \notin (0, \infty)$ and $\Gamma(a) = \int_{z=0}^{\infty} z^{a-1} e^{-z} dz$ is the gamma function for $a > 0$.

A.1.5 The lognormal distribution

If a random variable X is lognormally distributed with parameters μ and σ, then by definition $\ln X$ is normally distributed with mean μ and standard deviation σ. We can write

$$\ln X = \mu + \sigma Z$$

where Z follows the standard normal distribution. For the lognormal distribution the probability density function is given by

$$f(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi} x \sigma} e^{-(\ln x - \mu)^2/(2\sigma^2)}, \quad x > 0 \tag{A.5}$$

A.2 Maximum likelihood estimation

The theory in this section is selected from [5] with supplements from [33]. Informally speaking, the maximum likelihood estimate for a parameter θ is the point at which the observed sample is the most likely. This is the estimate that maximizes the likelihood function $L(\theta; \mathbf{t})$ for some fixed observations $\mathbf{t}$. If we have an i.i.d. sample $t_1, ..., t_n$ with probability density function $f(t; \theta_1, ..., \theta_k)$, the likelihood function is defined by

$$L(\theta; \mathbf{t}) = L(\theta_1, ..., \theta_k; t_1, ..., t_n) = \prod_{i=1}^{n} f(t_i; \theta_1, ..., \theta_k)$$

If the likelihood function is differentiable in θ_i, the possible maximum likelihood estimators are given by

$$\frac{\partial}{\partial \theta_i} L(\theta; \mathbf{t}) = 0, \quad i = 1, ..., k \tag{A.6}$$

We then have to check whether these possible candidates are global maxima.

It is often hard to do the differentiation in equation (A.6), and more convenient to work with the log-likelihood function:

$$l(\theta; \mathbf{t}) = \log L(\theta; \mathbf{t}) = \sum_{i=1}^{n} \log f(t_i; \theta_1, ..., \theta_k) \tag{A.7}$$

Since the extrema of $L(\theta; \mathbf{t})$ and $\log L(\theta; \mathbf{t})$ coincide, we will get same MLEs. These can be found either analytically or numerically by solving the system of equations resulting from $\frac{\partial}{\partial \theta_i} l(\theta; \mathbf{t}) = 0$. In this thesis we will use the `optim()` function in R to calculate the MLEs for our data.

A.2.1 Maximum likelihood estimation for censored data

If the dataset you wish estimate parameters from is censored, the likelihood function needs to be slightly modified. For the censored observations all we know is that the survival times are larger than the observed censoring times. Therefore we partition the data into two disjoint sets, one containing the failure times $t_1, ..., t_n$ and one containing the censoring times $\tau_1, ..., \tau_r$. For the censored observations we replace the PDF by the survival function in the likelihood function. Hence, the modified

likelihood function becomes:

$$\begin{aligned} L(\theta; \mathbf{t}) &= L(\theta_1, ..., \theta_k; t_1, ..., t_n, \tau_1, ..., \tau_r) \\ &= \prod_{i=1}^{n} f(t_i; \theta_1, ..., \theta_k) \prod_{j=1}^{r} S(\tau_j; \theta_1, ..., \theta_k) \end{aligned} \tag{A.8}$$

A.3 Stochastic processes

The following description is found in [34]. A stochastic process $\{X(t), t \in T\}$ is a collection of random variables, where t usually is time. The set T is called the index set of the process. When T is a countable set, the stochastic process is said to be a discrete-time process and when T is an interval of the real line, the stochastic process is said to be a continuous-time process. In reliability applications where the lifetime of an item is of interest, one will usually use a continuous stochastic process. $X(t)$ is often referred to as the state of the process at time t. Some basic concepts for describing stochastic processes are:

- **Independent increments** When a stochastic process is said to have independent increments, it means that the number of events that occur in disjoint time intervals are independent.

- **Stationary increments** The process can also have stationary increments, which means that the distribution of events that occur in any interval of time depends only on the length of the time interval and not on the interval's distance from the origin.

In this thesis we focus on a special case of a continuous-time stochastic process, namely the gamma process (see chapter 5).

A.4 The delta method

The following description of the delta method is from [5]. The delta method provides a way to find the distribution of a function of a random variable. The method is based on using Taylor series approximations of the mean and the variance of the function of the random variable. Suppose that we have a random variable X with mean $\theta \neq 0$ and a function $g(\cdot)$. A first-order approximation of $g(X)$ is then

$$g(X) = g(\theta) + g'(\mu)(X - \theta)$$

Thus, we can say that

$$\mathrm{E}[g(X)] \approx g(\theta)$$

Furthermore,

$$\begin{aligned}\mathrm{Var}[g(X)] &\approx \mathrm{E}([g(X) - g(\theta)]^2) \\ &\approx [g'(\theta)]^2 \mathrm{Var}(X)\end{aligned}$$

These estimates are used in the delta method which is described in the following way (p. 243 in [5]):

Let Y_n be a sequence of random variables that satisfies $\sqrt{n}(Y_n - \theta) \to N(0, \sigma^2)$ in distribution. For a given function g and a specific value of θ, suppose that $g'(\theta)$ exists and is not 0. Then

$$\sqrt{n}[g(Y_n) - g(\theta)] \to N(0, \sigma^2[g'(\theta)]^2) \quad \textit{in distribution}$$

Appendix B

Additional calculations

In this appendix we have included some additional calculations.

B.1 Calculating $\mathrm{E}[S|S < c]$

In the simulation studies of chapter 7 we compare the estimates of s from the basic gamma process model to $\mathrm{E}[S|S < c]$ in the random S models. For all of the random S models we have that

$$P(S > s|S < c) = \frac{P(s < S < c)}{P(S < c)} = \frac{\int_s^c f_S(s)ds}{F_S(c)} \tag{B.1}$$

Further,

$$P(s \leq S \leq S + ds|S < c) = \frac{d}{ds}(1 - P(S > s|S < c)) \tag{B.2}$$

and

$$\mathrm{E}[S|S < c] = \int_0^c sP(s \leq S \leq S + ds|S < c)ds \tag{B.3}$$

We insert for $f_S(s)$ and $F_S(c)$ in the four models:

B.1.1 Uniform S

With uniform S we have that $f_S(s)$ and $F_S(s)$ are given by equations (6.18) and (6.19). Hence, according to (B.1)

$$P(S > s|S < c) = \frac{\int_s^c \frac{1}{A}ds}{\frac{c}{A}} = \frac{c - s}{c}$$

This makes (B.2)

$$P(s \leq S \leq S + ds|S < c) = \frac{d}{ds}(1 - \frac{c - s}{c}) = \frac{1}{c}$$

and (B.3)

$$\mathrm{E}[S|S < c] = \int_0^c s\frac{1}{c}ds = \left[\frac{s^2}{2c}\right]_0^c = \frac{c}{2}$$

With the inserted parameter values from the simulation study in section 7.1.2 we get

$$\mathrm{E}[S|S<c] = \frac{5}{2} = 2.5$$

B.1.2 Exponential S

With exponential S we have that $f_S(s)$ and $F_S(s)$ are given by equations (6.20) and (6.21). Hence, according to (B.1)

$$P(S>s|S<c) = \frac{\int_s^c \lambda_S e^{-\lambda_S \cdot s} ds}{1-e^{-\lambda_S \cdot c}} = \frac{e^{-\lambda_S \cdot s} - e^{-\lambda_S \cdot c}}{1-e^{-\lambda_S \cdot c}}$$

This makes (B.2)

$$P(s \leq S \leq S+ds|S<c) = \frac{d}{ds}(1 - \frac{e^{-\lambda_S \cdot s} - e^{-\lambda_S \cdot c}}{1-e^{-\lambda_S \cdot c}}) = \frac{\lambda_S e^{-\lambda_S \cdot s}}{1-e^{-\lambda_S \cdot c}}$$

and (B.3)

$$\mathrm{E}[S|S<c] = \int_0^c s \frac{\lambda_S e^{-\lambda_S \cdot s}}{1-e^{-\lambda_S \cdot c}} ds = \frac{1}{\lambda_S} - \frac{c}{e^{\lambda_S \cdot c} - 1}$$

With the inserted parameter values from the simulation study in section 7.1.3 we get

$$\mathrm{E}[S|S<c] = \frac{1}{0.1} - \frac{7}{e^{0.1 \cdot 7} - 1} = 3.0950$$

B.1.3 Gamma distributed S

With gamma distributed S we have that $f_S(s)$ and $F_S(s)$ are given by equations (6.22) and (6.23). Hence, according to (B.1)

$$P(S>s|S<c) = \frac{\int_s^c \frac{\beta_S^{\alpha_S}}{\Gamma(\alpha_S)} s^{\alpha_S - 1} e^{-\beta_S \cdot s} ds}{1 - \frac{\Gamma(\alpha_S, \beta_S \cdot c)}{\Gamma(\alpha_S)}}$$

$$= \frac{\frac{\beta_S^{\alpha_S}}{\Gamma(\alpha_S)} \left(s^{\alpha_S}(\beta_S \cdot s)^{-\alpha_S}\Gamma(\alpha_S, -\beta_S \cdot s) - c^{\alpha_S}(\beta_S \cdot c)^{-\alpha_S}\Gamma(\alpha_S, -\beta_S \cdot c)\right)}{1 - \frac{\Gamma(\alpha_S, \beta_S \cdot c)}{\Gamma(\alpha_S)}}$$

This makes (B.2)

$$P(s \leq S \leq S+ds|S<c) = \frac{d}{ds}(1 - P(S>s|S<c)) = \frac{\beta_S^{\alpha_S} s^{\alpha_S - 1} e^{\beta_S \cdot s}}{\Gamma(\alpha_S) - \Gamma(\alpha_S, \beta_S \cdot c)}$$

and (B.3)

$$\mathrm{E}[S|S<c] = \frac{\beta_S^{\alpha_S}}{\Gamma(\alpha_S)-\Gamma(\alpha_S,\beta_S\cdot c)}\int_0^c s s^{\alpha_S-1}e^{\beta_S\cdot s}ds$$

$$\frac{1}{\Gamma(\alpha_S)-\Gamma(\alpha_S,\beta_S\cdot c)}\left(-\frac{(\beta_S c)^{\alpha_S}(\beta_S\cdot c)^{-\alpha_S}\Gamma(\alpha_S+1,\beta_S\cdot c)}{\beta_S}+\frac{\Gamma(\alpha_S+1)}{\beta_S}\right)$$

$$=\frac{\Gamma(\alpha_S+1)-\Gamma(\alpha_S+1,\beta_S\cdot c)}{\beta_S(\Gamma(\alpha_S)-\Gamma(\alpha_S,\beta_S\cdot c))}$$

With the inserted parameter values from the simulation study in section 7.1.4 we get

$$\mathrm{E}[S|S<c] = \frac{\Gamma(12.25+1)-\Gamma(12.25+1,1.75\cdot 7)}{1.75(\Gamma(12.25)-\Gamma(12.25,1.75\cdot 7))} = 5.5270$$

B.1.4 Lognormal S

With lognormal S we have that $f_S(s)$ and $F_S(s)$ are given by equations (6.24) and (6.25). Hence, according to (B.1)

$$P(S>s|S<c) = \frac{\int_s^c \frac{1}{s\sqrt{2\pi}\sigma_S}e^{\frac{(\ln s-\mu_S)^2}{2\sigma_S^2}}ds}{\Phi\left(\frac{\ln c-\mu_S}{\sigma_S}\right)}$$

$$=\frac{1}{\Phi\left(\frac{\ln c-\mu_S}{\sigma_S}\right)}\left[-\frac{1}{2}erf\left(\frac{\mu_S-\ln c}{\sqrt{2}\sigma_S}\right)+\frac{1}{2}erf\left(\frac{\mu_S-\ln s}{\sqrt{2}\sigma_S}\right)\right]$$

This makes (B.2)

$$P(s\le S\le S+ds|S<c) = \frac{d}{ds}(1-P(S>s|S<c)) = \frac{1}{\Phi\left(\frac{\ln c-\mu_S}{\sigma_S}\right)}\frac{1}{s\sqrt{2\pi}\sigma_S}e^{\frac{(\mu_S-\ln s)^2}{2\sigma_S^2}}$$

and (B.3)

$$\mathrm{E}[S|S<c] = \int_0^c s\frac{1}{\Phi\left(\frac{\ln c-\mu_S}{\sigma_S}\right)}\frac{1}{s\sqrt{2\pi}\sigma_S}e^{\frac{(\mu_S-\ln s)^2}{2\sigma_S^2}}ds$$

$$=-\left[\frac{e^{\mu_S+\frac{\sigma_S^2}{2}}erf\left(\frac{\mu_S+\sigma_S^2-\ln(s)}{\sqrt{2}\sigma_S}\right)}{erf\left(\frac{\ln(c)-\mu_S}{\sqrt{2}\sigma_S}\right)+1}\right]_0^c$$

$$=\frac{e^{\mu_S+\frac{\sigma_S^2}{2}}\left(1-erf\left(\frac{\mu_S+\sigma_S^2-\ln(c)}{\sqrt{2}\sigma_S}\right)\right)}{1-erf\left(\frac{\mu_S-\ln(c)}{\sqrt{2}\sigma_S}\right)}$$

With the inserted parameter values from the simulation study in section 7.1.5 we get

$$\mathrm{E}[S|S<c] = \frac{e^{2+\frac{0.25^2}{2}}\left(1-erf\left(\frac{2+0.25^2-\ln(7)}{\sqrt{2}0.25}\right)\right)}{1-erf\left(\frac{2-\ln(7)}{\sqrt{2}0.25}\right)} = 5.8964$$

Appendix C

Data sets

We have used two of the same datasets as in the project thesis, the VHF-data and the carcinoma data. In addition, a set containing bone marrow transplant data is included for the data analysis of semi-competing risks with censored observations. The data sets are further described in chapters 8 and 9 of the thesis.

C.1 VHF data

This data set is from [27].

TABLE C.1: Observations of times to failure X from the VHF data

16	224	16	80	128	168	144	176	176	568
392	576	128	56	112	160	384	600	40	416
408	384	256	246	184	440	64	104	168	408
304	16	72	8	88	160	48	168	80	512
208	194	136	224	32	504	40	120	320	48
256	216	168	184	144	224	488	304	40	160
488	120	208	32	112	288	336	256	40	296
60	208	440	104	528	384	264	360	80	96
360	232	40	112	120	32	56	280	104	168
56	72	64	40	480	152	48	56	328	192
168	168	114	280	128	416	392	160	144	208
96	536	400	80	40	112	160	104	224	336
616	224	40	32	192	126	392	288	248	120
328	464	448	616	169	112	448	296	328	56
80	72	56	608	144	408	16	560	144	612
80	16	424	264	256	528	56	256	112	544
552	72	184	240	128	40	600	96	24	184
272	152	328	480	96	296	592	400	8	280
72	168	40	152	488	480	40	576	392	552
112	288	168	352	160	272	320	80	296	248
184	264	96	224	592	176	256	344	360	184
152	208	160	176	72	584	144	176	-	-

TABLE C.2: Observations of times to censoring Z from the VHF data

368	136	512	136	472	96	144	112	104	104
344	246	72	80	312	24	128	304	16	320
560	168	120	616	24	176	16	24	32	232
32	112	56	184	40	256	160	456	48	24
200	72	168	288	112	80	584	368	272	208
144	208	114	480	114	392	120	48	104	272
64	112	96	64	360	136	168	176	256	112
104	272	320	8	440	224	280	8	56	216
120	256	104	104	8	304	240	88	248	472
304	88	200	392	168	72	40	88	176	216
152	184	400	424	88	152	184	-	-	-

C.2 Carcinoma data

This dataset is from [20].

TABLE C.3: Observations of times to failure X from the carcinoma data

0.43	3.86	11.14	29.14	61.68
2.86	6.14	13.0	29.71	66.57
3.14	6.86	14.43	40.57	68.71
3.14	9.0	15.71	48.57	68.99
3.43	9.43	18.43	49.43	72.86
3.43	10.71	18.57	53.86	72.86
3.71	10.86	20.71	-	-

TABLE C.4: Observations of times to censoring Z from the carcinoma data. The numbers in parenthesis are the times to failure for the censored observations X_Z

0.14 (3.0)	1.86 (12.14)	6.0 (38.0)	16.57 (45.0)	26.00 (53.86)
0.14 (12.43)	3.0 (7.86)	6.14 (9.29)	17.29 (24.14)	27.57 (49.71)
0.29 (1.14)	3.0 (13.86)	8.71 (20.43)	18.71 (29.43)	32.14 (63.86)
0.43 (17.14)	3.29 (10.57)	10.57 (25.0)	21.29 (26.71)	33.14 (99.0)
0.57 (4.43)	3.29 (34.43)	11.86 (17.29)	23.86 (29.0)	47.29 (48.71)
0.57 (5.43)	6.0 (7.86)	15.57(21.57)	-	-

C.3 Bonemarrow transplant data

This data is taken from [19]. In the thesis, we have only used the data from columns 2–6 .

g - disease group
- 1 - ALL
- 2 - AML low-risk
- 3 - AML high-risk

T_1 - Time (in days) to death or on time study time

T_2 - Disease-free survival time (time to relapse, death or end of study)

δ_1 - Death indicator
- 1 - Dead
- 0 - Alive

δ_2 - Relapse indicator
- 1 - Relapsed
- 0 - Disease-free

δ_3 - Disease-free survival indicator
- 1 - Dead or relapsed
- 0 - Alive disease-free

T_A - Time (in days) to acute graft-versus-host disease

δ_A - Acute graft-versus-host disease indicator
- 1 - Developed acute graft-versus-host disease
- 0 - Never developed acute graft-versus-host disease

T_C - Time (in days) to chronic graft-versus-host disease

δ_C - Chronic graft-versus-host disease indicator
- 1 - Developed chronic graft-versus-host disease
- 0 - Never developed chronic graft-versus-host disease

T_P - Time (in days) to return of platelets to normal levels

δ_P - Platelet recovery indicator
- 1 - Platelets returned to normal levels
- 0 - Platelets never returned to normal levels

TABLE C.5: Data on 137 bone marrow transplant patients

g	T_1	T_2	δ_1	δ_2	δ_3	T_A	δ_A	T_C	δ_C	T_P	δ_P
1	2081	2081	0	0	0	67	1	121	1	13	1
1	1602	1602	0	0	0	1602	0	139	1	18	1
1	1496	1496	0	0	0	1496	0	307	1	12	1
1	1462	1462	0	0	0	70	1	95	1	13	1
1	1433	1433	0	0	0	1433	0	236	1	12	1
1	1377	1377	0	0	0	1377	0	123	1	12	1
1	1330	1330	0	0	0	1330	0	96	1	17	1
1	996	996	0	0	0	72	1	121	1	12	1
1	226	226	0	0	0	226	0	226	0	10	1
1	1199	1199	0	0	0	1199	0	91	1	29	1
1	1111	1111	0	0	0	1111	0	1111	0	22	1
1	530	530	0	0	0	38	1	84	1	34	1
1	1182	1182	0	0	0	1182	0	112	1	22	1
1	1167	1167	0	0	0	39	1	487	1	1167	0
1	418	418	1	0	1	418	0	220	1	21	1
1	417	383	1	1	1	417	0	417	0	16	1
1	276	276	1	0	1	276	0	81	1	21	1
1	156	104	1	1	1	28	1	156	0	20	1
1	781	609	1	1	1	781	0	781	0	26	1
1	172	172	1	0	1	22	1	172	0	37	1
1	487	487	1	0	1	487	0	76	1	22	1
1	716	662	1	1	1	716	0	716	0	17	1
1	194	194	1	0	1	194	0	94	1	25	1
1	371	230	1	1	1	371	0	184	1	9	1
1	526	526	1	0	1	526	0	121	1	11	1
1	122	122	1	0	1	88	1	122	0	13	1
1	1279	129	1	1	1	1279	0	1279	0	22	1
1	110	74	1	1	1	110	0	110	0	49	1
1	243	122	1	1	1	243	0	243	0	23	1
1	86	86	1	0	1	86	0	86	0	86	0
1	466	466	1	0	1	466	0	119	1	100	1
1	262	192	1	1	1	10	1	84	1	59	1
1	162	109	1	1	1	162	0	162	0	40	1
1	262	55	1	1	1	262	0	262	0	24	1
1	1	1	1	0	1	1	0	1	0	1	0
1	107	107	1	0	1	107	0	107	0	107	0

Table C.5: (continued)

g	T_1	T_2	δ_1	δ_2	δ_3	T_A	δ_A	T_C	δ_C	T_P	δ_P
1	269	110	1	1	1	269	0	120	1	27	1
1	350	332	1	0	1	350	0	350	0	33	1
2	2569	2569	0	0	0	2569	0	2569	0	21	1
2	2506	2506	0	0	0	2506	0	2506	0	17	1
2	2409	2409	0	0	0	2409	0	2409	0	16	1
2	2218	2218	0	0	0	2218	0	2218	0	11	1
2	1857	1857	0	0	0	1857	0	260	1	15	1
2	1829	1829	0	0	0	1829	0	1829	0	19	1
2	1562	1562	0	0	0	1562	0	1562	0	18	1
2	1470	1470	0	0	0	1470	0	180	1	14	1
2	1363	1363	0	0	0	1363	0	200	1	12	1
2	1030	1030	0	0	0	1030	0	210	1	14	1
2	860	860	0	0	0	860	0	860	0	15	1
2	1258	1258	0	0	0	1258	0	120	1	66	1
2	2246	2246	0	0	0	52	1	380	1	15	1
2	1870	1870	0	0	0	1870	0	230	1	16	1
2	1799	1799	0	0	0	1799	0	140	1	12	1
2	1709	1709	0	0	0	20	1	348	1	19	1
2	1674	1674	0	0	0	1674	0	1674	0	24	1
2	1568	1568	0	0	0	1568	0	1568	0	14	1
2	1527	1527	0	0	0	1527	0	1527	0	13	1
2	1324	1324	0	0	0	25	1	1324	0	15	1
2	957	957	0	0	0	957	0	957	0	69	1
2	932	932	0	0	0	29	1	932	0	7	1
2	847	847	0	0	0	847	0	847	0	16	1
2	848	848	0	0	0	848	0	155	1	16	1
2	1850	1850	0	0	0	1850	0	1850	0	9	1
2	1843	1843	0	0	0	1843	0	1843	0	19	1
2	1535	1535	0	0	0	1535	0	1535	0	21	1
2	1447	1447	0	0	0	1447	0	220	1	24	1
2	1384	1384	0	0	0	1384	0	200	1	19	1
2	414	414	1	0	1	414	0	414	0	27	1
2	2204	2204	1	0	1	2204	0	2204	0	12	1
2	1063	1063	1	0	1	1063	0	240	1	16	1
2	481	481	1	0	1	30	1	120	1	24	1
2	105	105	1	0	1	21	1	105	0	15	1

Table C.5: (continued)

g	T_1	T_2	δ_1	δ_2	δ_3	T_A	δ_A	T_C	δ_C	T_P	δ_P
2	641	641	1	0	1	641	0	641	0	11	1
2	390	390	1	0	1	390	0	390	0	11	1
2	288	288	1	0	1	18	1	100	1	288	0
2	522	421	1	1	1	25	1	140	1	20	1
2	79	79	1	0	1	16	1	79	0	79	0
2	1156	748	1	1	1	1153	0	180	1	18	1
2	583	486	1	1	1	583	0	583	0	11	1
2	48	48	1	0	1	48	0	48	0	14	1
2	431	272	1	1	1	431	0	431	0	12	1
2	1074	1074	1	0	1	1074	0	120	1	19	1
2	393	381	1	1	1	393	0	100	1	16	1
2	10	10	1	0	1	10	0	10	0	10	0
2	53	53	1	0	1	53	0	53	0	53	0
2	80	80	1	0	1	10	1	80	0	80	0
2	35	35	1	0	1	35	0	35	0	35	0
2	1499	248	0	1	1	1499	0	1499	0	9	1
2	704	704	1	0	1	36	1	155	1	18	1
2	653	211	1	1	1	653	0	653	0	23	1
2	222	219	1	1	1	222	0	123	1	52	1
2	1356	606	0	1	1	1356	0	1356	0	14	1
3	2640	2640	0	0	0	2640	0	2640	0	22	1
3	2430	2430	0	0	0	2430	0	2430	0	14	1
3	2252	2252	0	0	0	2252	0	150	1	17	1
3	2140	2140	0	0	0	2140	0	220	1	18	1
3	2133	2133	0	0	0	2133	0	250	1	17	1
3	1238	1238	0	0	0	1238	0	250	1	18	1
3	1631	1631	0	0	0	1631	0	150	1	40	1
3	2024	2024	0	0	0	2024	0	180	1	16	1
3	1345	1345	0	0	0	32	1	360	1	14	1
3	1136	1136	0	0	0	1236	0	140	1	15	1
3	845	845	0	0	0	845	0	845	0	20	1
3	491	422	1	1	1	491	0	180	1	491	0
3	162	162	1	0	1	162	0	162	0	13	1
3	1298	84	1	1	1	1298	0	1298	0	1298	0
3	121	100	1	1	1	28	1	121	0	65	1
3	2	2	1	0	1	2	0	2	0	2	0

Table C.5: (continued)

g	T_1	T_2	δ_1	δ_2	δ_3	T_A	δ_A	T_C	δ_C	T_P	δ_P
3	62	47	1	1	1	62	0	62	0	11	1
3	265	242	1	1	1	265	0	210	1	65	1
3	547	456	1	1	1	547	0	130	1	24	1
3	341	268	1	1	1	21	1	100	1	17	1
3	318	318	1	0	1	318	0	140	1	12	1
3	195	32	1	1	1	195	0	195	0	16	1
3	469	467	1	1	1	469	0	90	1	20	1
3	93	47	1	1	1	93	0	93	0	28	1
3	515	390	1	1	1	515	0	515	0	31	1
3	183	183	1	0	1	183	0	130	1	21	1
3	105	105	1	0	1	105	0	105	0	105	0
3	128	115	1	1	1	128	0	128	0	12	1
3	164	164	1	0	1	164	0	164	0	164	0
3	129	93	1	1	1	129	0	129	0	51	1
3	122	120	1	1	1	122	0	122	0	12	1
3	80	80	1	0	1	21	1	80	0	0	1
3	677	677	1	0	1	677	0	150	1	8	1
3	73	64	1	1	1	73	0	73	0	38	1
3	168	168	1	0	1	168	0	200	1	48	1
3	74	74	1	0	1	29	1	74	0	24	1
3	16	16	1	0	1	16	0	16	0	16	0
3	248	157	1	1	1	248	0	100	1	52	1
3	732	625	1	1	1	732	0	732	0	18	1
3	105	48	1	1	1	105	0	105	0	30	1
3	392	273	1	1	1	392	0	122	1	24	1
3	63	63	1	0	1	38	1	63	0	16	1
3	97	76	1	1	1	97	0	97	0	97	0
3	153	113	1	1	1	153	0	153	0	59	1
3	363	363	1	0	1	363	0	363	0	19	1

Appendix D

R functions

In this appendix we present the functions that were used to generate the results from the simulation study in chapter 7 and the data analyses in chapters 8 and 9. A lot of the code is just expanded from the project thesis and written exactly as in [36] so that the results of the gamma process models could be compared directly with the Wiener process models of Skogsrud and Lindqvist.

D.1 Simulation

In this section, the code for simulating data from the gamma process is shown. The function `simdata`() simulates data from the first passage time distribution of the gamma process. This function is used by the programs `simRandomS`() and `simSemiCens`() which respectively simulate competing risks data and semi-competing risks data from the gamma process models defined in chapter 6. In addition, `simSemiCens`() uses the function `simdata2`() to draw from the first passage time distribution to the level c given $z = t_1$ and starting from level s.

D.1.1 Simulation from first passage time distribution

```
simdata = function(a,b,d,N){

  # Purpose: simulate data from the distribution of the first passage time
  #          in a gamma process (f(t)) with shape function v(t) = a*t^b,
  #          scale parameter u = 1 and critical level d
  # Input:
  #  - a: parameter value (alpha in the thesis)
  #  - b: parameter value (beta in the thesis)
  #  - d: critical level
  #  - N: number of samples
  # Output:
  #  - T: vector of N samples from f(t)

  # first define a grid of possible values for t
  # (NB: has to be adjusted to match the parameter values!)
  t = seq(0,4,0.01)

  # generate N t-values from f(t) by the inverse transformation method
  tsamples = c()
  for(n in 1:N){
```

```
    unif = runif(1,0,1)
    ti = 0
    tj = 0
    for(i in 1:length(t)){
      F = pgamma(d, a*t[i]^b,lower = FALSE)
      if(F > unif){
        tj = t[i]
        ti = t[i-1]
        uj = F
        ui = pgamma(d, a*t[i-1]^b,lower = FALSE)
        break
      }
    }
    tsamples[n] = (uj - unif)/(uj - ui)*ti + (unif - ui)/(uj - ui)*tj
  }
  T = tsamples
}


simdata2 = function(a,b,c,S,N,t1){

  # Purpose: simulate data from the distribution of the first passage
  #          time in a gamma process (f(t)) with shape function v(t) =
  #          a*t^b, scale parameter u = 1 and critical level c starting
  #          at level S and time z = t1 (f(t1) = S)
  # Input:
  #  - a  : parameter value (alpha in the thesis)
  #  - b  : parameter value (beta in the thesis)
  #  - c  : critical level
  #  - S  : starting point
  #  - t1 : starting time
  #  - N  : number of samples
  # Output:
  #  - T  : vector of N samples from f(t)
  # Define a grid of possible values for t
  # (NB: has to be adjusted to match the parameter values!)
  t = seq(t1+0.01,4,0.01)

  # generate N t-values from f(t) by the inverse transformation method
  tsamples = c()
  for(n in 1:N){
    unif = runif(1,0,1)
    ti = 0
    tj = 0
    for(i in 2:length(t)){
      F = pgamma(c-S, a*t[i]^b- a*t1^b,lower = FALSE)
      if(F > unif){
        tj = t[i]
        ti = t[i-1]
```

```
        uj = F
        ui = pgamma(c-S, a*t[i-1]^b-a*t1^b,lower = FALSE)
        break
      }
    }
    tsamples[n] = (uj - unif)/(uj - ui)*ti + (unif - ui)/(uj - ui)*tj
  }
  T = tsamples
}
```

D.1.2 Random S models - competing risks

```
simRandomS = function(c, s,a, b,q, mu_s, sigma_s, A, alpha_s,beta_s,
                      lambda_s, type, N){

  # Purpose: simulate N samples from a gamma process model with censoring
  #          (ordinary competing risks data)
  # Input:
  #  - c: critical level in the gamma process
  #  - a: parameter value (alpha in the thesis)
  #  - b: parameter value (beta in the thesis)
  #  - N: number of samples
  #  - type: if type = "unif":   simulate from uniform model where S is
  #                              uniformly distributed on (0,A)
  #          if type = "expon":  simulate from the exponential model
  #                              where S is exponentially distributed
  #                              with parameter lambda_s
  #          if type = "gamma":  simulate from the gamma model where S
  #                              is gamma distributed with shape para-
  #                              meter alpha_s and scale parameter
  #                              beta_s
  #          if type ="lognorm": simulate from the lognormal model where
  #                              S is lognormally distributed with para-
  #                              meters mu_s and sigma_s
  #          if type = "const":  simulate from the basic model where S
  #                              has value s and q is the probability
  #                              of observing Z.
  # Output:
  #  - data: data consisting of a column T of times and a column C of
  #          indication variables
  #  - x   : vector of observations from X (C=1)
  #  - z   : vector of observations from Z (C=2)
  #  - tau : vector of censored observations (C=3)

  x = c()
  z = c()
  tau = c()
  j = 1
  k = 1
```

```
l = 1
C = c()
T = c()

for(i in 1:N){
 if(type == "unif"){
    S = runif(1, min =0, max=A)
    if(S < c){
      q = 1
    }else {
      q = 0
    }
  } else if(type == "gamma"){
    S = rgamma(1, alpha_s, beta_s)
    if(S < c){
      q = 1
    }else {
      q = 0
    }
  } else if(type == "expon"){
    S = rexp(1, lambda_s)
    if(S < c){
      q = 1
    }else {
      q = 0
    }
  } else if(type == "lognorm"){
    S = rlnorm(1, mu_s, sigma_s)
    if(S < c){
      q = 1
    }else {
      q = 0
    }
  } else if(type == "const"){
    S = s
    q = q
  } else{
    print("wrong input type")
    break
  }
  u = runif(1,0,1)
  Tau = rgamma(1,1,0.1)
  # NB! these parameters need to be adjusted depending on the model!

  if(u < q){
    time = simdata(a,b,S,1)
    if(time < Tau){
      z[j] = time
```

```
        T[i] = z[j]
        C[i] = 2
        j = j+1
      }else{
        tau[l] = Tau
        T[i] = tau[l]
        C[i] = 3
        l = l+1
      }
    }
    else{
      time = simdata(a,b,c,1)
      if(time < Tau){
        x[k] = time
        T[i] = x[k]
        C[i] = 1
        k = k+1
      }else{
        tau[l] = Tau
        T[i] = tau[l]
        C[i] = 3
        l = l+1
      }
    }
  }
  data = data.frame(T=T,C=C)
  sorted = sort(T, index = TRUE)
  data.sort = data[sorted$ix,]
  list(x = x, z = z, tau = tau, data = data.sort)
}
```

D.1.3 Random S models - semi-competing risks

```
simSemiCens = function(c, a, b, A, lambda_s, alpha_s, beta_s,mu_s,
                       sigma_s, type,  N){

  # Purpose: simulate N samples from a gamma process model with censoring
  #          (semi-competing risks data)
  # Input:
  #  - c: critical level in the gamma process
  #  - a: parameter value (alpha in the thesis)
  #  - b: parameter value (beta in the thesis)
  #  - N: number of samples
  #  - type: if type = "unif":   simulate from uniform model where S is
  #                              uniformly distributed on (0,A)
  #          if type = "expon":  simulate from the exponential model
  #                              where S is exponentially distributed
  #                              with parameter lambda_s
  #          if type = "gamma":  simulate from the gamma model where S is
```

```
#                               gamma distributed with shape parameter
#                               alpha_s and scale parameter beta_s
#          if type ="lognorm": simulate from the lognormal model where
#                               S is lognormally distributed with para-
#                               meters mu_s and sigma_s
# Output:
#  - x    : vector of observations X
#  - z    : vector of observations Z
#  - x_z  : vector of observations X_Z following Z
#  - z_o  : vector of observations Z_o
#  - tau_o: vector of censored observations following Z_o
#  - tau  : vector of censored observations

x = c()
z = c()
x_z = c()
z_o = c()
tau_o = c()
tau = c()

j = 1
k = 1
l = 1
r =1

for(i in 1:N){
  if(type == "unif"){
    S = runif(1, min =0, max=A)
    if(S < c){
      q = 1
    }else {
      q = 0
    }
  } else if(type == "expon"){
    S = rexp(1, lambda_s)
    if(S < c){
      q = 1
    }else {
      q = 0
    }
  } else if(type == "gamma"){
    S = rgamma(1, alpha_s, beta_s)
    if(S < c){
      q = 1
    }else {
      q = 0
    }
  } else if(type == "lognorm"){
```

```
      S = rlnorm(1, mu_s, sigma_s)
      if(S < c){
        q = 1
      }else {
        q = 0
      }
    } else{
      print("wrong input type")
      break
    }
    u = runif(1,0,1)
    Tau = rgamma(1,1,0.1)
    # NB! Needs to be adjusted depending on the distribution

    if(u < q){
      t1 = simdata(a,b,S,1)
      if(t1 < Tau){
        t2 = simdata2(a,b,c,S,1,t1)
        if(t2 < Tau){
          z[j] = t1
          x_z[j] = t2
          j = j +1
        }else{
          z_o[r] = t1
          tau_o[r] = Tau
          r = r+1
        }
      } else{
        tau[l] = Tau
        l = l+1
      }
    }else{
      time = simdata(a,b,c,1)
      if(time < Tau){
        x[k] = time
        k = k+1
      } else{
        tau[l] = Tau
        l = l+1
      }
    }
  }
  list(x=x, z=z ,x_z=x_z, z_o=z_o, tau_o=tau_o, tau=tau)
}
```

D.2 Estimation

The parameter estimation is done by the `optim`() function in R. In the competing risks case this is called by the function `condSurv`(), which also is used to make non-parametric estimates of the conditional sub-survival functions. To make these estimates, the function `subdistrZ`() is also needed. In the semi-competing risks case, `optim`() is called by the function `estSemi`() to do the parameter estimation. To make estimates and plots of the crude and net quantities, we use the script `semiQuant`.

In order to use `optim`() one needs to have the expressions for the log-likelihood functions of the different models. These are calculated by the functions `lklhBasic`(), `lklhUnif`(), `lklhExp`(), `lklhGam`() and `lklhLognorm`() in the competing risks case and by `lklhUnifSemi`(), `lklhExpSemi`(), `lklhGamSemi`(), `lklhLnormSemi`() and `lklhNormSemi`() in the semi-competing risks case.

Furthermore, bootstrap estimates for the VHF-data and the carcinoma data are found using the script in D.2.5.

D.2.1 Competing risks case

```
condSurv = function(data, p0=NULL, upper=NULL, lower=NULL, type="plot"){

  # Purpose: plot the non-parametric estimates of the conditional sub-
  #          survival functions of X and Z by using the method described
  #          in Lawless(2003). In addition it can estimate parameters in
  #          the gamma process models: basic, uniform, exponential,
  #          gamma and lognormal
  # Input:
  # data - data-frame on the form [t,C] where t is a column of sorted
  #        failure times and C is a column of failure modes; C = 1 means
  #        failure (X), C = 2 means PM (Z) and C = 3 means censoring (tau)
  # p0   - vector of starting values for estimation of parameters by one
  #        of the gamma process models. Default is p0 = NULL, which is
  #        used when there is no estimation, only plots of the non-
  #        parametric estimates of the conditional sub-survival functions
  #        of X and Z
  # lower -vector of lower parameter values for estimation. Default is
  #        NULL, used in the basic model or when there is no estimation,
  #        only plots of the non-parametric estimates of the conditional
  #        sub-survival functions of X and Z
  # upper -vector of upper parameter values for estimation. Default is
  #        NULL, used in the basic model or when there is no estimation,
  #        only plots of the non-parametric estimates of the conditional
  #        sub-survival functions of X and Z
  # type - string that states which of the gamma process models the data
  #        should be fitted to. Possibilities: "const"(basic model),
  #        "unif", "expon", "gamma", "lognorm". Default is "plot" which
  #        means no estimation, only plots of the non-parametric
```

```
#        estimates of the conditional sub-survival functions of X and Z
# Output:
# Default is only plots of the non-parametric estimates of the condi-
# tional sub-survival functions of X(thin line) and Z(thick line). If
# estimation by one of the gamma process models is performed, the
# output is a list consisting of:
# par   - estimated parameters
# std   - standard deviation of the estimates, based on the Hessian
# logL  - value of log-likelihood function for the estimated parameters
# cor   - matrix of correlations between the parameters
# lower - lower bound in a 95% standard positive confidence interval
# upper - upper bound in a 95% standard positive confidence interval
# In addition, parametric estimates of the conditional sub-survival
# functions of X (thin dotted line) and Z (thick dotted line) are
# plotted in the same plot as the non-parametric estimates.

N = length(data[,1])
C = data[,2]
t = data[,1]

# make columns deltaix and deltaiz:
# (to estimate the non-parametric conditional sub-distr. functions)
n = c() # number at risk
deltaix = c()
deltaiz = c()
d = c()
ix = c()

n[1] = N
ix[1] = 1
if(C[1] == 1){
  deltaix[1] = 1
  deltaiz[1] = 0
} else if(C[1]==2){
    deltaix[1] = 0
    deltaiz[1] = 1
} else{
    deltaix[1] = 0
    deltaiz[1] = 0
}

j = 2
for(i in 2:N){
  n[i] = N -i +1
  if(t[i] != t[i-1] && C[i] != 3){
    ix[j] = i
    j = j+1
  } else{
```

```
    j = j
  }
  if(C[i] == 1){
    deltaix[i] = 1
    deltaiz[i] = 0
  } else if(C[i] == 2){
    deltaix[i] = 0
    deltaiz[i] = 1
  } else{
    deltaix[i] = 0
    deltaiz[i] = 0
  }
}
# find the column d which shows number of items failing at time t
j = 1
for(i in 1:(N+1)){
  if(i == ix[j]){
    d[i] = sum(deltaix[ix[j]:ix[j+1]]) + sum(deltaiz[ix[j]:ix[j+1]])-1
    if(j < (length(ix)-1)){
      j = j+1
    } else{
      j = j
    }
  } else{
    d[i]= 0
  }
}
ix.n = length(ix)
index = ix[ix.n]
d[index] = 1
d.ny = d[1:N]

p = (n - d.ny)/n
S = cumprod(p)

nelson.x = deltaix/n
nelson.z = deltaiz/n

px = c()
pz = c()

if(deltaix[1] == 1){
  px[1] = nelson.x[1]
  pz[1] = 0
} else{
  px[1] = 0
  pz[1] = nelson.z[1]
}
```

```
for(i in 2:N){
  if(deltaix[i] == 1){
    px[i] = S[i-1]*nelson.x[i]
    pz[i] = 0
  } else{
    px[i] = 0
    pz[i] = S[i-1]*nelson.z[i]
  }
}

Fx = cumsum(px)
Fz = cumsum(pz)
Sx = 1 - Fx
Sz = 1 - Fz

n = 0
m = 0
x = c()
z = c()
tau = c()

tellerx = 1
tellerz = 1
tellertau = 1

# make the vectors x and z from the indexing in the data
for(i in 1:N){
  if(C[i] == 1){
    x[tellerx] = t[i]
    tellerx = tellerx +1
  } else if (C[i] == 2){
      z[tellerz] = t[i]
      tellerz = tellerz + 1
  } else{
      tau[tellertau] = t[i]
      tellertau = tellertau + 1
  }
}
if(is.null(tau)==T){ tau = 0}

n = length(z)
m = length(x)
r = length(tau)

if(type =="const"){
  opt = optim(par =p0, fn=lklhBasic, x=x,z=z,tau=tau,method="BFGS",
              hessian = T)
```

```
  p = opt$par
  SZ.hat = subdistrZ(p,t,type="const")
}else if(type == "unif"){
  opt = optim(par = p0, fn=lklhUnif, x = x, z=z, tau = tau, method =
  "L-BFGS-B", lower = lower, upper = upper, hessian = T)
  p = opt$par
  SZ.hat = subdistrZ(p, t, type = "unif")
}else if(type == "gamma"){
  opt = optim(par = p0 , fn=lklhGam, x = x, z=z, tau = tau,method =
  "L-BFGS-B", lower = lower, upper = upper, hessian = T)
  p = opt$par
  SZ.hat = subdistrZ(p, t, type = "gamma")
}else if(type == "expon"){
  opt = optim(par = p0, fn=lklhExp, x = x, z=z, tau = tau,method =
  "L-BFGS-B", lower = lower, upper = upper, hessian = T)
  p = opt$par
  SZ.hat = subdistrZ(p, t, type = "expon")
}else if(type == "lognorm"){
  opt = optim(par = p0, fn=lklhLognorm, x = x, z=z, tau = tau,
  method = "L-BFGS-B", lower = lower,upper =upper, hessian=T)
  p = opt$par
  SZ.hat = subdistrZ(p, t, type = "lognorm")
}else if(type == "plot"){
}else{
    stop("wrong type input")
}

# non-parametric estimate
q = Fz[N]/(Fx[N] + Fz[N])

Fx.cond = Fx/(1-q)
Fz.cond = Fz/q
Sx.cond = 1-Fx.cond
Sz.cond = 1-Fz.cond

if(type =="const" || type == "unif" || type == "gamma"
|| type == "expon" || type == "lognorm"){
  SX.hat = pgamma(p[3],p[1]*t^p[2])
}

plot(t,Sx.cond,type="s", ylab = "Sj(t)", ylim = c(0,1))
lines(t, Sz.cond, type="s", lwd =2)

# Results from estimation
if(type != "plot"){
  lines(t,SX.hat, type = "l", lty = 2)
  lines(t,SZ.hat, type = "l", lty = 2, lwd = 2)
```

```
    std = sqrt(solve(opt$hessian))
    lower = p*exp(-1.96*diag(std)/p)
    upper = p*exp(1.96*diag(std)/p)
    list(par = opt$par, std = diag(std), logL = -opt$value, cor =
    cov2cor(solve(opt$hessian)), lower = lower, upper = upper)
  }
}
```

D.2.2 Functions used by the estimation function `condSurv()` (competing risks)

```
subdistrZ = function(p, t, type) {

  # Purpose: estimate the conditional sub-survival function for Z in the
  #          gamma process models
  # Input:
  # p    - vector of values for the parameters in the desired model
  # t    - vector of times
  # type - string that states which of the gamma process models SZ.hat
  #        should be estimated from. The possibilities are: "const"
  #        (basic), "unif", "expon", "gamma" and "lognorm"
  # Output:
  # SZ.hat - estimated conditional sub-survival function for Z

  int = c()

  if(type == "unif"){
    const = p[4]/p[3]
    funk = function(s, p, t){
      pgamma(s, p[1]*t^p[2])*1/p[4]
    }
  }else if(type == "gamma"){
    const = 1/pgamma(p[3],p[4],p[5])
    funk = function(s, p,t){
      pgamma(s, p[1]*t^p[2])*dgamma(s,p[4],p[5])
    }
  }else if(type == "expon"){
    const = 1/pexp(p[3],p[4])
    funk = function(s, p,t){
      pgamma(s, p[1]*t^p[2])*dexp(s,p[4])
    }
  }else if(type == "lognorm"){
    const = 1/plnorm(p[3],p[4],p[5])
    funk = function(s, p,t){
      pgamma(s, p[1]*t^p[2])*dlnorm(s,p[4],p[5])
    }
  }else if(type == "const"){
    s = p[4]
    SZ.hat = pgamma(s,p[1]*t^p[2])
```

```
  }
  if(type != "const"){
    for(i in 1:length(t)){
      int[i] = integrate(funk, lower = 0.01, upper = p[3], p = p,
               t = t[i])$value
    }
    SZ.hat = int*const
  }
  SZ.hat
}


lklhBasic = function(param,x,z,tau){

  # Purpose: find an expression for the log-likelihood function in the
  #          basic model for competing risks data, with censoring
  # Input:
  # - param: vector of starting values for the parameters a,b,c,s and q
  # - x    : vector of observations of times to failure
  # - z    : vector of observations of times to PM
  # - tau  : vector of censored observations
  # Output:
  # -  -lnL: minus the log-likelihood function, which will be minimized
  #          by optim() in condSurv()

  m = length(x)
  n = length(z)
  r = length(tau)

  a = param[1]
  b = param[2]
  c = param[3]
  s = param[4]
  q = param[5]

  xdensity = c()
  zdensity = c()
  taudensity = c()

  require(hypergeo)
  # Contribution from X
  for(i in 1:m){
    xp = x[i]
    up = c(a*xp^b,a*xp^b)
    lo = c(a*xp^b+1,a*xp^b +1)
    nder = b*a*xp^(b-1)
    psi = digamma(a*xp^b)
    fun = genhypergeo(up,lo,-c,tol=0, maxiter=2000, check_mod=TRUE,
          polynomial=FALSE, debug=FALSE, series=TRUE)
```

```
    xdensity[i] = nder*(psi-log(c))*pgamma(c,a*xp^b) +
                  (nder/((a*xp^b)^2*gamma(a*xp^b)))*((c)^(a*xp^b))*fun
  }
  # Contribution from Z
  for(j in 1:n){
    zp = z[j]
    up = c(a*zp^b,a*zp^b)
    lo = c(a*zp^b+1,a*zp^b +1)
    nder = b*a*zp^(b-1)
    psi = digamma(a*zp^b)
    fun = genhypergeo(up,lo,-s,tol=0, maxiter=2000, check_mod=TRUE,
          polynomial=FALSE, debug=FALSE, series=TRUE)
    zdensity[j] = nder*(psi-log(s))*pgamma(s,a*zp^b) +
                  (nder/((a*zp^b)^2*gamma(a*zp^b)))*((s)^(a*zp^b))*fun
  }
  # Contribution from tau
  for(k in 1:r){
    tau_p = tau[k]
    taudensity[k] = (1-q)*pgamma(c,a*tau_p^b) + q*pgamma(s,a*tau_p^b)
  }
  -(m*log(1-q)+sum(log(xdensity))+n*log(q)+sum(log(zdensity))+
    sum(log(taudensity)))
}


lklhUnif = function(param,x,z,tau){

  # Purpose: find an expression for the log-likelihood function in the
  #          uniform model for competing risks data with censoring
  # Input:
  # - param: vector of starting values for the parameters a,b,c and A
  # - x    : vector of observations of times to failure
  # - z    : vector of observations of times to PM
  # - tau  : vector of censored observations
  # Output:
  # -  -lnL: minus the log-likelihood function, which will be minimized
  #          by optim() in condSurv()

  a = param[1]
  b = param[2]
  c = param[3]
  A = param[4]

  m = length(x)
  n = length(z)
  r = length(tau)

  xdensity = c()
  zdensity = c()
```

```
  taudensity = c()

  require(hypergeo)
  for(i in 1:m){
  # Contribution from X
    xp = x[i]
    up = c(a*xp^b,a*xp^b)
    lo = c(a*xp^b+1,a*xp^b +1)
    nder = b*a*xp^(b-1)
    psi = digamma(a*xp^b)
    fun = genhypergeo(up,lo,-c,tol=0, maxiter=2000, check_mod=TRUE,
          polynomial=FALSE, debug=FALSE, series=TRUE)
    xdensity[i] = (nder*(psi-log(c))*pgamma(c,a*xp^b) +
                  (nder/((a*xp^b)^2*gamma(a*xp^b)))*((c)^(a*xp^b))*fun)
                  *(1-c/A)
  }
  for(j in 1:n){
  # Contribution from Z
    zp = z[j]
    up = c(a*zp^b,a*zp^b)
    lo = c(a*zp^b+1,a*zp^b +1)
    nder = b*a*zp^(b-1)
    psi = digamma(a*zp^b)
    zdensity[j] = integrate(uniffunc, lower = 0.01, upper = c, a = a, b =
                  b, z = zp,up = up, lo = lo, nder = nder, psi = psi, A =
                  A)$value
  }
  for(k in 1:r){
   # Contribution from tau
    tau_p = tau[k]
    integral = integrate(tauUnif, lower = 0.01, upper = c, a = a, b = b,
               tau = tau_p, A = A)$value
    taudensity[k] = (1-c/A)*pgamma(c,a*tau_p^b) + integral
  }
  -(sum(log(xdensity))+sum(log(zdensity))+sum(log(taudensity)))
}
uniffunc = function(s,a, b, z, up, lo, nder, psi,A){
 (nder*(psi-log(s))*pgamma(s,a*z^b) +
 (nder/((a*z^b)^2*gamma(a*z^b)))*((s)^(a*z^b))*genhypergeo(up,lo,-s,tol=0,
  maxiter=2000, check_mod=TRUE,polynomial=FALSE,debug=FALSE,series=TRUE))
  *1/A
}
tauUnif = function(s, a,b,tau,A){
  pgamma(s, a*tau^b)*1/A
}


lklhExp = function(param,x,z,tau){
```

```
# Purpose: find an expression for the log-likelihood function in the
#          exponential model for competing risks data with censoring
# Input:
# - param: vector of starting values for the parameters a,b,c and
#          lambda_s
# - x    : vector of observations of times to failure
# - z    : vector of observations of times to PM
# - tau  : vector of censored observations
# Output:
# -  -lnL: minus the log-likelihood function, which will be minimized
#          by optim() in condSurv()

a = param[1]
b = param[2]
c = param[3]
lambda_s= param[4]

m = length(x)
n = length(z)
r = length(tau)

xdensity = c()
zdensity = c()
taudensity = c()

require(hypergeo)
for(i in 1:m){
# Contribution from X
  xp = x[i]
  up = c(a*xp^b,a*xp^b)
  lo = c(a*xp^b+1,a*xp^b +1)
  nder = b*a*xp^(b-1)
  psi = digamma(a*xp^b)
  fun = genhypergeo(up,lo,-c,tol=0, maxiter=2000, check_mod=TRUE,
        polynomial=FALSE, debug=FALSE, series=TRUE)
  xdensity[i] = (nder*(psi-log(c))*pgamma(c,a*xp^b) +
                (nder/((a*xp^b)^2*gamma(a*xp^b)))*((c)^(a*xp^b))*fun)
                *(1-pexp(c,lambda_s))
}
for(j in 1:n){
# Contribution from Z
  zp = z[j]
  up = c(a*zp^b,a*zp^b)
  lo = c(a*zp^b+1,a*zp^b +1)
  nder = b*a*zp^(b-1)
  psi = digamma(a*zp^b)
  zdensity[j] = integrate(expfunc, lower = 0.01, upper = c, a = a,b = b,
                z = zp,up = up, lo = lo, nder = nder, psi = psi,lambda_s
```

```
                  = lambda_s)$value
  }
  for(k in 1:r){
   # Contribution from tau
    tau_p = tau[k]
    integral = integrate(tauExp, lower = 0.01, upper = c, a = a, b = b,
               tau = tau_p, lambda_s = lambda_s)$value
    taudensity[k] = (1-pexp(c,lambda_s))*pgamma(c,a*tau_p^b) + integral
  }
  -(sum(log(xdensity))+sum(log(zdensity)) +sum(log(taudensity)))
}
expfunc = function(s,a, b, z, up, lo, nder, psi,lambda_s){
    (nder*(psi-log(s))*pgamma(s,a*z^b) +
    (nder/((a*z^b)^2*gamma(a*z^b)))*((s)^(a*z^b))*genhypergeo(up,lo,-s,
    tol=0,maxiter=2000, check_mod=TRUE,polynomial=FALSE, debug=FALSE,
    series=TRUE))*(dexp(s,lambda_s))
}
tauExp = function(s, a,b,tau,lambda_s){
   pgamma(s, a*tau^b)*dexp(s,lambda_s)
}


lklhGam = function(param,x,z,tau){

  # Purpose: find an expression for the log-likelihood function in the
  #          gamma model for competing risks data, with censoring
  # Input:
  # - param: vector of starting values for the parameters a,b,c, alpha_s
  #          and beta_s
  # - x    : vector of observations of times to failure
  # - z    : vector of observations of times to PM
  # - tau  : vector of censored observations
  # Output:
  # -  -lnL: minus the log-likelihood function, which will be minimized
  #          by optim() in condSurv()

  a = param[1]
  b = param[2]
  c = param[3]
  alpha_s= param[4]
  beta_s = param[5]

  m = length(x)
  n = length(z)
  r = length(tau)

  xdensity = c()
  zdensity = c()
  taudensity = c()
```

```
  require(hypergeo)
  for(i in 1:m){
  # Contribution from X
    xp = x[i]
    up = c(a*xp^b,a*xp^b)
    lo = c(a*xp^b+1,a*xp^b +1)
    nder = b*a*xp^(b-1)
    psi = digamma(a*xp^b)
    fun = genhypergeo(up,lo,-c,tol=0, maxiter=2000, check_mod=TRUE,
          polynomial= FALSE, debug=FALSE, series=TRUE)
    xdensity[i] = (nder*(psi-log(c))*pgamma(c,a*xp^b) +
                  (nder/((a*xp^b)^2*gamma(a*xp^b)))*((c)^(a*xp^b))*fun)
                  *(1-pgamma(c,alpha_s,beta_s))
  }
  for(j in 1:n){
  # Contribution from Z
    zp = z[j]
    up = c(a*zp^b,a*zp^b)
    lo = c(a*zp^b+1,a*zp^b +1)
    nder = b*a*zp^(b-1)
    psi = digamma(a*zp^b)
    zdensity[j] = integrate(gamfunc,lower = 0.01, upper = c, a = a, b = b,
                  z = zp, up = up, lo = lo, nder = nder, psi = psi,
                  alpha_s = alpha_s, beta_s = beta_s)$value
  }
  for(k in 1:r){
   # Contribution from tau
    tau_p = tau[k]
    integral = integrate(tauGam, lower = 0.01, upper = c, a = a, b = b,
               tau =tau_p, alpha_s = alpha_s, beta_s = beta_s)$value
    taudensity[k] = (1-pgamma(c,alpha_s,beta_s))*pgamma(c,a*tau_p^b)
                     + integral
  }
  -(sum(log(xdensity))+sum(log(zdensity)) +sum(log(taudensity)))
}
gamfunc = function(s,a, b, z, up, lo, nder, psi,alpha_s, beta_s){
  (nder*(psi-log(s))*pgamma(s,a*z^b) +
  (nder/((a*z^b)^2*gamma(a*z^b)))*((s)^(a*z^b))*genhypergeo(up,lo,-s,
  tol=0, maxiter=2000, check_mod=TRUE,polynomial=FALSE, debug=FALSE,
  series=TRUE))*(dgamma(s,alpha_s,beta_s))
}
tauGam = function(s, a,b,tau,alpha_s, beta_s){
  pgamma(s, a*tau^b)*dgamma(s,alpha_s,beta_s)
}


lklhLognorm = function(param,x,z,tau){
```

```
# Purpose: find an expression for the log-likelihood function in the
#          lognormal model for competing risks data, with censoring
# Input:
# - param: vector of starting values for the parameters a,b,c, mu_s
#          and sigma_s
# - x    : vector of observations of times to failure
# - z    : vector of observations of times to PM
# - tau  : vector of censored observations
# Output:
# -  -lnL: minus the log-likelihood function, which will be minimized
#          by optim() in condSurv()

a = param[1]
b = param[2]
c = param[3]
mu_s= param[4]
sigma_s = param[5]

m = length(x)
n = length(z)
r = length(tau)

xdensity = c()
zdensity = c()
taudensity = c()

require(hypergeo)
for(i in 1:m){
# Contribution from X
  xp = x[i]
  up = c(a*xp^b,a*xp^b)
  lo = c(a*xp^b+1,a*xp^b +1)
  nder = b*a*xp^(b-1)
  psi = digamma(a*xp^b)
  fun = genhypergeo(up,lo,-c,tol=0, maxiter=2000, check_mod=TRUE,
        polynomial=FALSE, debug=FALSE, series=TRUE)
  xdensity[i] = (nder*(psi-log(c))*pgamma(c,a*xp^b) +
                (nder/((a*xp^b)^2*gamma(a*xp^b)))*((c)^(a*xp^b))*fun)
                *(1-plnorm(c,mu_s,sigma_s))
}
for(j in 1:n){
# Contribution from Z
  zp = z[j]
  up = c(a*zp^b,a*zp^b)
  lo = c(a*zp^b+1,a*zp^b +1)
  nder = b*a*zp^(b-1)
  psi = digamma(a*zp^b)
  zdensity[j] = integrate(lnormfunc, lower = 0.01, upper = c, a = a,
```

```
                  b = b,z = zp, up = up, lo = lo, nder = nder, psi = psi,
                  mu_s = mu_s, sigma_s = sigma_s)$value
  }
  for(k in 1:r){
   # Contribution from tau
    tau_p = tau[k]
    integral = integrate(tauLogN, lower = 0.01, upper = c, a = a, b = b,
               tau = tau_p, mu_s = mu_s, sigma_s = sigma_s)$value
    taudensity[k] = (1-plnorm(c,mu_s,sigma_s))*pgamma(c,a*tau_p^b)
                    + integral
  }
  -(sum(log(xdensity))+sum(log(zdensity)) +sum(log(taudensity)))
}
lnormfunc = function(s,a, b, z, up, lo, nder, psi,mu_s, sigma_s){
  (nder*(psi-log(s))*pgamma(s,a*z^b) +
  (nder/((a*z^b)^2*gamma(a*z^b)))*((s)^(a*z^b))*genhypergeo(up,lo,-s,
  tol=0,maxiter=2000, check_mod=TRUE,polynomial=FALSE, debug=FALSE,
  series=TRUE))*(dlnorm(s,mu_s,sigma_s))
}
tauLogN = function(s, a,b,tau,mu_s, sigma_s){
  pgamma(s, a*tau^b)*dlnorm(s,mu_s,sigma_s)
}
```

D.2.3 Semi-competing risks case

```
estSemi = function(x,z,x_z,z_o=0,tau_o=0,tau=0, p0, upper, lower, type) {

  # Purpose: estimate parameters in any of the gamma process models:
  #          uniform, exponential, gamma, lognormal and normal
  #          for semi-competing risks data
  #
  # Input:
  # data  - event times x, z, x_z, z_o, tau_o, tau
  #         (only x, z, and x_z if no censoring)
  # p0    - vector of starting values for estimation of the parameters in
  #         one of the gamma process models
  # lower - vector of lower parameter estimates
  # upper - vector of upper parameter estimates
  # type  - string that states which of the gamma process models the data
  #         should be fitted to. Possibilities: "unif", "expon", "gamma",
  #         "lognorm", "norm".
  # Output:
  # The output is a list consisting of:
  # par   - estimated parameters
  # std   - standard deviation of the estimates, based on the Hessian
  # logL  - value of log likelihood function for the estimated parameters
  # cor   - matrix of correlations between the parameters
  # lower - lower bound in a 95% standard positive confidence interval
  # upper - upper bound in a 95% standard positive confidence interval
```

```
  if(type == "unif"){
    opt = optim(par = p0, fn=lklhUnifSemi, x = x, z = z, xz = x_z, z_o =
                z_o, tau_o = tau_o, tau = tau, method = "L-BFGS-B",
                lower = lower, upper = upper, hessian = T)
  }else if(type == "gamma"){
    opt = optim(par = p0, fn=lklhGamSemi, x = x, z = z, xz = x_z, z_o =
                z_o, tau_o = tau_o, tau = tau, method = "L-BFGS-B",
                lower = lower, upper = upper, hessian = T)
  }else if(type == "expon"){
    opt = optim(par = p0, fn=lklhExpSemi, x = x, z = z, xz = x_z, z_o =
                z_o, tau_o = tau_o, tau = tau, method = "L-BFGS-B",
                lower = lower, upper = upper, hessian = T)
  }else if(type == "lognorm"){
    opt = optim(par = p0, fn=lklhLnormSemi, x = x, z = z, xz = x_z, z_o =
                z_o, tau_o = tau_o, tau = tau, method = "L-BFGS-B",
                lower = lower, upper = upper, hessian = T)
  }else if(type == "norm"){
    opt = optim(par = p0, fn=lklhNormSemi, x = x, z = z, xz = x_z, z_o =
                z_o, tau_o = tau_o, tau = tau, method = "L-BFGS-B",
                lower = lower, upper = upper, hessian = T)
  } else{
    stop("wrong type input")
  }
  p = opt$par
  std = sqrt(solve(opt$hessian))
  lower = p*exp(-1.96*diag(std)/p)
  upper = p*exp(1.96*diag(std)/p)
  list(par = opt$par, std = diag(std), logL = -opt$value, cor =
       cov2cor(solve(opt$hessian)), lower = lower, upper = upper)
}


# semiQuant

# Script to estimate crude + net quantities for semi-competing risks data

# Crude quantities:
# Parametric and non-parametric estimates of F^*_Z(t) and Lambda^*_Z(t)
# Net quantities:
# The marginal survivor functions of Z and X and the marginal hazard rates
# of Z and X

# Non-parametric estimates require the following data:
# - T:      vector of min(Z,X,tau), sorted from lowest to highest
# - delta:  vector of indicator variables corresponding to the entries
#           in T, delta=1 if the non-terminal event has happened,
#           delta = 0 if not
# - cens:   vector of indicator variables corresponding to the entries in
```

```
#           T, cens = 1 if both X and Z were censored, cens=0 if not
# Parametric estimates require:
# - t       : vector of t-values where the functions should be estimated
# - p_unif  : vector of parameter estimates in the uniform model
# - p_exp   : vector of parameter estimates in the exponential model
# - p_gam   : vector of parameter estimates in the gamma model
# - p_lnorm : vector of parameter estimates in the lognormal model

# Non-parametric curves, expressions from section 4.4
# Kaplan Meier estimate of S_T(t)
N = length(T)
n = c() # number at risk
deltaix = c()
d = c()
ix = c()

n[1] = N
ix[1] = 1
if(cens[1]== 0){
  deltaix[1] = 1
} else{
  deltaix[1] = 0
}

j = 2
for(i in 2:N){
  n[i] = N -i +1
  if(T[i] != T[i-1] && cens[i] ==0){
    ix[j] = i
    j = j+1
  } else{
    j = j
  }
  if(cens[i]== 0){
    deltaix[i] = 1
  } else{
    deltaix[i] = 0
  }
}
j = 1

for(i in 1:(N+1)){
  if(i == ix[j]){
    d[i] = sum(deltaix[ix[j]:ix[j+1]]) -1
    if(j < (length(ix)-1)){
      j = j+1
    } else{
      j = j
```

```
    }
  } else{
    d[i]= 0
  }
}
ix.n = length(ix)
index = ix[ix.n]
d[index] = 1
d.ny = d[1:N]

p = (n - d.ny)/n
S = cumprod(p)

# make Nelson-Aalen type estimate of Lambda^*_Z(t)
N_A = c()
for(t in 1:length(T)){
  teller = c()
  for(i in 1:length(T)){
    if((T[i] < T[t] || T[i] == T[t]) && delta[i] ==1){
      teller[i] = 1
    } else{
      teller[i] = 0
    }
  }
  bidrag = teller/n
  N_A[t] = sum(bidrag)
}

# make estimate of F^*_Z(t)
FZ.star = c()
for(j in 1:length(T)){
  sum_tot = 0
  for(i in 1:length(T)){
    if((T[i] < T[j] || T[i] == T[j])){
      sum_tot = sum_tot + S[i]*teller[i]/n[i]
    }
  }
  FZ.star[j] = sum_tot
}

# parametric estimates of crude quantities
F1_unif = c()
F1_exp = c()
F1_gam = c()
F1_lnorm = c()

funcUnif = function(s, t, p){
  (1-pgamma(s, p[1]*t^p[2]))*1/p[4]
```

```
}
funcExp = function(s, t, p){
  (1-pgamma(s, p[1]*t^p[2]))*dexp(s, p[4])
}
funcGam = function(s, t, p){
  (1-pgamma(s, p[1]*t^p[2]))*dgamma(s, p[4], p[5])
}
funcLnorm = function(s,t, p){
  (1-pgamma(s, p[1]*t^p[2]))*dlnorm(s, p[4], p[5])
}

for(i in 1:length(t)){
  tp = t[i]
  F1_unif[i] = integrate(funcUnif, lower= 0.01, upper = p_unif[3], p=
      p_unif, t= tp)$value
  F1_exp[i] = integrate(funcExp, lower= 0.01, upper = p_exp[3], p=
              p_exp, t= tp)$value
  F1_gam[i] = integrate(funcGam, lower= 0.01, upper = p_gam[3], p=
              p_gam, t= tp)$value
  F1_lnorm[i] = integrate(funcLnorm, lower= 0.01, upper = p_lnorm[3],
                p=p_lnorm, t= tp)$value
}

f1_unif = c()
f1_exp = c()
f1_gam = c()
f1_lnorm = c()

funcUnif2 = function(s, t, p, nder, psi, up, lo){
  (nder*(psi-log(s))*pgamma(s,p[1]*t^p[2]) +
  (nder/((p[1]*t^p[2])^2*gamma(p[1]*t^p[2])))*((s)^(p[1]*t^p[2]))
  *genhypergeo(up,lo,-s,tol=0, maxiter=2000, check_mod=TRUE,polynomial=
  FALSE, debug=FALSE, series=TRUE))*1/p[4]
}
funcExp2 = function(s, t, p, nder, psi, up, lo){
  (nder*(psi-log(s))*pgamma(s,p[1]*t^p[2]) +
  (nder/((p[1]*t^p[2])^2*gamma(p[1]*t^p[2])))*((s)^(p[1]*t^p[2]))
  *genhypergeo(up,lo,-s,tol=0, maxiter=2000, check_mod=TRUE,polynomial=
  FALSE,debug=FALSE, series=TRUE))*dexp(s, p[4])
}
funcGam2 = function(s, t, p, nder, psi, up, lo){
  (nder*(psi-log(s))*pgamma(s,p[1]*t^p[2]) +
  (nder/((p[1]*t^p[2])^2*gamma(p[1]*t^p[2])))*((s)^(p[1]*t^p[2]))
  *genhypergeo(up,lo,-s,tol=0, maxiter=2000, check_mod=TRUE,polynomial=
  FALSE,debug=FALSE, series=TRUE))*dgamma(s, p[4],p[5])
}
funcLnorm2 = function(s, t, p, nder, psi, up, lo){
  (nder*(psi-log(s))*pgamma(s,p[1]*t^p[2]) +
```

```
  (nder/((p[1]*t^p[2])^2*gamma(p[1]*t^p[2])))*((s)^(p[1]*t^p[2]))
  *genhypergeo(up,lo,-s,tol=0, maxiter=2000, check_mod=TRUE,polynomial=
  FALSE,debug=FALSE, series=TRUE))*dlnorm(s, p[4],p[5])
}

for(i in 1:length(t)){
  tp = t[i]
  up_unif = c(p_unif[1]*tp^p_unif[2],p_unif[1]*tp^p_unif[2])
  lo_unif = c(p_unif[1]*tp^p_unif[2]+1,p_unif[1]*tp^p_unif[2] +1)
  nder_unif = p_unif[2]*p_unif[1]*tp^(p_unif[2]-1)
  psi_unif = digamma(p_unif[1]*tp^p_unif[2])

  up_exp = c(p_exp[1]*tp^p_exp[2],p_exp[1]*tp^p_exp[2])
  lo_exp = c(p_exp[1]*tp^p_exp[2]+1,p_exp[1]*tp^p_exp[2] +1)
  nder_exp = p_exp[2]*p_exp[1]*tp^(p_exp[2]-1)
  psi_exp = digamma(p_exp[1]*tp^p_exp[2])

  up_gam = c(p_gam[1]*tp^p_gam[2],p_gam[1]*tp^p_gam[2])
  lo_gam = c(p_gam[1]*tp^p_gam[2]+1,p_gam[1]*tp^p_gam[2] +1)
  nder_gam = p_gam[2]*p_gam[1]*tp^(p_gam[2]-1)
  psi_gam = digamma(p_gam[1]*tp^p_gam[2])

  up_lnorm = c(p_lnorm[1]*tp^p_lnorm[2],p_lnorm[1]*tp^p_lnorm[2])
  lo_lnorm = c(p_lnorm[1]*tp^p_lnorm[2]+1,p_lnorm[1]*tp^p_lnorm[2] +1)
  nder_lnorm = p_lnorm[2]*p_lnorm[1]*tp^(p_lnorm[2]-1)
  psi_lnorm = digamma(p_lnorm[1]*tp^p_lnorm[2])

  f1_unif[i] = integrate(funcUnif2, lower= 0.01, upper = p_unif[3], p =
               p_unif, t= tp, nder = nder_unif, psi = psi_unif, up =
               up_unif, lo = lo_unif)$value
  f1_exp[i] = integrate(funcExp2, lower= 0.01, upper = p_exp[3], p =
              p_exp, t= tp, nder = nder_exp, psi = psi_exp, up = up_exp,
              lo = lo_exp)$value
  f1_gam[i] = integrate(funcGam2, lower= 0.01, upper = p_gam[3], p =
              p_gam, t= tp, nder = nder_gam, psi = psi_gam, up = up_gam,
              lo = lo_gam)$value
  f1_lnorm[i] = integrate(funcLnorm2, lower= 0.01, upper = p_lnorm[3], p
                = p_lnorm, t= tp, nder = nder_lnorm, psi = psi_lnorm, up
                = up_lnorm, lo = lo_lnorm)$value
}
s_ny_unif = c()
s_ny_exp = c()
s_ny_gam = c()
s_ny_lnorm = c()

tauUnif = function(s,t,p){
  pgamma(s, p[1]*t^p[2])*1/p[4]
}
```

```
tauExp = function(s, t,p){
  pgamma(s, p[1]*t^p[2])*dexp(s,p[4])
}
tauGam = function(s, t, p){
  pgamma(s, p[1]*t^p[2])*dgamma(s,p[4],p[5])
}
tauLnorm = function(s, t,p){
  pgamma(s, p[1]*t^p[2])*dlnorm(s,p[4],p[5])
}

for(k in 1:length(t)){
  tp = t[k]
  integral_unif = integrate(tauUnif, lower = 0.01, upper = p_unif[3], p =
                  p_unif, t = tp)$value
  integral_exp = integrate(tauExp, lower = 0.01, upper = p_exp[3], p =
                 p_exp, t = tp)$value
  integral_gam = integrate(tauGam, lower = 0.01, upper = p_gam[3], p =
                 p_gam, t = tp)$value
  integral_lnorm = integrate(tauLnorm, lower = 0.01, upper = p_lnorm[3], p
                   = p_lnorm, t = tp)$value

  s_ny_unif[k] = (1-p_unif[3]/p_unif[4])*pgamma(p_unif[3],p_unif[1]
                  *tp^p_unif[2])+ integral_unif
  s_ny_exp[k] = (1-pexp(p_exp[3],p_exp[4]))*pgamma(p_exp[3],p_exp[1]
                *tp^p_exp[2]) + integral_exp
  s_ny_gam[k] = (1-pgamma(p_gam[3],p_gam[4],p_gam[5]))
  *pgamma(p_gam[3],p_gam[1]*tp^p_gam[2]) + integral_gam
  s_ny_lnorm[k] = (1 plnorm(p_lnorm[3],p_lnorm[4],p_lnorm[5]))
  *pgamma(p_lnorm[3],p_lnorm[1]*tp^p_lnorm[2])+ integral_lnorm
}

lambda1_unif = f1_unif /s_ny_unif
lambda1_exp = f1_exp /s_ny_exp
lambda1_gam = f1_gam /s_ny_gam
lambda1_lnorm = f1_lnorm /s_ny_lnorm

require(caTools)
Lambda_unif = c()
Lambda_exp = c()
Lambda_gam = c()
Lambda_lnorm = c()

for(i in 1:length(t)){
  t_vec = t[1:i]
  lambda_vec_unif = lambda1_unif[1:i]
  lambda_vec_exp = lambda1_exp[1:i]
  lambda_vec_gam = lambda1_gam[1:i]
  lambda_vec_lnorm = lambda1_lnorm[1:i]
```

```
  Lambda_unif[i] = trapz(t_vec,lambda_vec_unif)
  Lambda_exp[i] = trapz(t_vec,lambda_vec_exp)
  Lambda_gam[i] = trapz(t_vec,lambda_vec_gam)
  Lambda_lnorm[i] = trapz(t_vec,lambda_vec_lnorm)
}

plot(T,FZ.star,type="s", ylim =c(0,1), ylab =expression(F[Z]^"*"),
     xlab = "t")
lines(t,F1_unif,lty=2 ,col ="orange")
lines(t,F1_exp,lty=2, col ="red")
lines(t,F1_gam,lty=2, col = "blue")
lines(t,F1_lnorm,lty=2, col = "forestgreen")
legend(60,1, c(expression(paste("unif ",hat(F)[Z]^"*"*(t))),
expression(paste("exp ",hat(F)[Z]^"*"*(t))),
expression(paste("gamma ",hat(F)[Z]^"*"*(t))),
expression(paste("lognorm ",hat(F)[Z]^"*"*(t))),
expression(paste("non-par ",hat(F)[Z]^"*"*(t)))), lty=c(2,2,2,2,1),
lwd=c(1,1,1,1,1),col=c("orange", "red", "blue", "forestgreen","black"),
bty ="n",cex=0.9)

plot(T,N_A,type="s", ylim =c(0,1.5), ylab =expression(Lambda[Z]^"*"),
     xlab = "t")
lines(t,Lambda_unif,lty = 2, col ="orange")
lines(t,Lambda_exp,lty = 2, col ="red")
lines(t,Lambda_gam,lty = 2, col = "blue")
lines(t,Lambda_lnorm,lty = 2, col = "forestgreen")
legend(0,1.5, c(expression(paste("unif ",hat(Lambda)[Z]^"*"*(t))),
expression(paste("exp ",hat(Lambda)[Z]^"*"*(t))),
expression(paste("gamma ",hat(Lambda)[Z]^"*"*(t))),
expression(paste("lognorm ",hat(Lambda)[Z]^"*"*(t))),
expression(paste("non-par ",hat(Lambda)[Z]^"*"*(t)))), lty=c(2,2,2,2,1),
lwd=c(1,1,1,1,1),col=c("orange", "red", "blue", "forestgreen","black"),
bty ="n",cex=0.9)

# parametric estimates if net quantities
SZ_unif = c()
SZ_exp = c()
SZ_gam = c()
SZ_lnorm = c()

funk_unif = function(s, p, t){
  pgamma(s, p[1]*t^p[2])*dunif(s,0,p[4])
}
funk_exp = function(s, p, t){
  pgamma(s, p[1]*t^p[2])*dexp(s,p[4])
}
funk_gam = function(s, p, t){
  pgamma(s, p[1]*t^p[2])*dgamma(s,p[4],p[5])
```

```
}
funk_lnorm = function(s, p, t){
  pgamma(s, p[1]*t^p[2])*dlnorm(s,p[4],p[5])
}

for(i in 1:length(t)){
  SZ_unif[i] = integrate(funk_unif, lower = 0, upper = Inf, p = p_unif,
  t = t[i])$value
  SZ_exp[i] = integrate(funk_exp, lower = 0, upper = Inf, p = p_exp,
   t = t[i])$value
  SZ_gam[i] = integrate(funk_gam, lower = 0, upper = 100, p = p_gam,
  t = t[i])$value
  SZ_lnorm[i] = integrate(funk_lnorm, lower = 0, upper = Inf, p = p_lnorm,
   t = t[i])$value
}
# OBS: integratiion does not work for all datasets or parameter values.
# If not, try with a large number instead of Inf as upper limit

SX_unif = pgamma(p_unif[3], p_unif[1]*t^p_unif[2])
SX_exp = pgamma(p_exp[3], p_exp[1]*t^p_exp[2])
SX_gam = pgamma(p_gam[3], p_gam[1]*t^p_gam[2])
SX_lnorm = pgamma(p_lnorm[3], p_lnorm[1]*t^p_lnorm[2])

pdf_x_unif = c()
pdf_x_exp = c()
pdf_x_gam = c()
pdf_x_lnorm = c()

pdf_z_unif = c()
pdf_z_exp = c()
pdf_z_gam = c()
pdf_z_lnorm = c()

funk2_unif = function(s,p, t, up, lo, nder, psi){
  (nder*(psi-log(s))*pgamma(s,p[1]*t^p[2]) +
  (nder/((p[1]*t^p[2])^2*gamma(p[1]*t^p[2])))*((s)^(p[1]*t^p[2]))
  *genhypergeo(up,lo,-s,tol=0, maxiter=2000, check_mod=TRUE,polynomial=
  FALSE, debug=FALSE, series=TRUE))*dunif(s,0,p[4])
}
funk2_exp = function(s,p, t, up, lo, nder, psi){
  (nder*(psi-log(s))*pgamma(s,p[1]*t^p[2]) +
  (nder/((p[1]*t^p[2])^2*gamma(p[1]*t^p[2])))*((s)^(p[1]*t^p[2]))
  *genhypergeo(up,lo,-s,tol=0, maxiter=2000, check_mod=TRUE,polynomial=
  FALSE, debug=FALSE, series=TRUE))*(dexp(s,p[4]))
}
funk2_gam = function(s,p, t, up, lo, nder, psi){
  (nder*(psi-log(s))*pgamma(s,p[1]*t^p[2]) +
  (nder/((p[1]*t^p[2])^2*gamma(p[1]*t^p[2])))*((s)^(p[1]*t^p[2]))
```

```
  *genhypergeo(up,lo,-s,tol=0, maxiter=2000, check_mod=TRUE,polynomial=
  FALSE, debug=FALSE, series=TRUE))*(dgamma(s,p[4],p[5]))
}
funk2_lnorm = function(s,p, t, up, lo, nder, psi){
  (nder*(psi-log(s))*pgamma(s,p[1]*t^p[2]) +
  (nder/((p[1]*t^p[2])^2*gamma(p[1]*t^p[2])))*((s)^(p[1]*t^p[2]))
  *genhypergeo(up,lo,-s,tol=0, maxiter=2000, check_mod=TRUE,polynomial=
  FALSE,debug=FALSE, series=TRUE))*(dlnorm(s,p[4],p[5]))
}

for(i in 1:length(t)){
  tp = t[i]

  up_unif = c(p_unif[1]*tp^p_unif[2],p_unif[1]*tp^p_unif[2])
  lo_unif = c(p_unif[1]*tp^p_unif[2]+1,p_unif[1]*tp^p_unif[2] +1)
  nder_unif = p_unif[2]*p_unif[1]*tp^(p_unif[2]-1)
  psi_unif = digamma(p_unif[1]*tp^p_unif[2])
  fun_unif = genhypergeo(up_unif,lo_unif,-p_unif[3],tol=0, maxiter=2000,
  check_mod=TRUE,polynomial=FALSE, debug=FALSE, series=TRUE)
  pdf_x_unif[i] = (nder_unif*(psi_unif-log(p_unif[3]))
                  *pgamma(p_unif[3],p_unif[1]*tp^p_unif[2]) +
                  (nder_unif/((p_unif[1]*tp^p_unif[2])^2*gamma(p_unif[1]
                  *tp^p_unif[2])))*((p_unif[3])^(p_unif[1]*tp^p_unif[2]))
                  *fun_unif)
  pdf_z_unif[i] = integrate(funk2_unif, lower = 0, upper = Inf p = p_unif,
                  t = tp, up = up_unif, lo = lo_unif, nder = nder_unif,
                  psi = psi_unif,subdivisions=2000)$value

  up_exp = c(p_exp[1]*tp^p_exp[2],p_exp[1]*tp^p_exp[2])
  lo_exp = c(p_exp[1]*tp^p_exp[2]+1,p_exp[1]*tp^p_exp[2] +1)
  nder_exp = p_exp[2]*p_exp[1]*tp^(p_exp[2]-1)
  psi_exp = digamma(p_exp[1]*tp^p_exp[2])
  fun_exp = genhypergeo(up_exp,lo_exp,-p_exp[3],tol=0, maxiter=2000,
            check_mod=TRUE,polynomial=FALSE, debug=FALSE, series=TRUE)
  pdf_x_exp[i] = (nder_exp*(psi_exp-log(p_exp[3]))
                 *pgamma(p_exp[3],p_exp[1]*tp^p_exp[2]) +
                 (nder_exp/((p_exp[1]*tp^p_exp[2])^2*gamma(p_exp[1]
                 *tp^p_exp[2])))*((p_exp[3])^(p_exp[1]*tp^p_exp[2]))
                 *fun_exp)
  pdf_z_exp[i] = integrate(funk2_exp, lower = 0, upper = Inf, p = p_exp,
                 t = tp, up = up_exp, lo = lo_exp, nder = nder_exp, psi =
                 psi_exp,subdivisions = 10000)$value

  up_gam = c(p_gam[1]*tp^p_gam[2],p_gam[1]*tp^p_gam[2])
  lo_gam = c(p_gam[1]*tp^p_gam[2]+1,p_gam[1]*tp^p_gam[2] +1)
  nder_gam = p_gam[2]*p_gam[1]*tp^(p_gam[2]-1)
  psi_gam = digamma(p_gam[1]*tp^p_gam[2])
  fun_gam = genhypergeo(up_gam,lo_gam,-p_gam[3],tol=0, maxiter=2000,
```

```
            check_mod=TRUE,polynomial=FALSE, debug=FALSE, series=TRUE)
  pdf_x_gam[i] = (nder_gam*(psi_gam-log(p_gam[3]))
                 *pgamma(p_gam[3],p_gam[1]*tp^p_gam[2]) +
                 (nder_gam/((p_gam[1]*tp^p_gam[2])^2
                 *gamma(p_gam[1]*tp^p_gam[2])))*((p_gam[3])
                 ^(p_gam[1]*tp^p_gam[2]))*fun_gam)
  pdf_z_gam[i] = integrate(funk2_gam, lower = 0, upper = Inf, p = p_gam,
                 t = tp, up = up_gam, lo = lo_gam, nder = nder_gam, psi
                 = psi_gam,subdivisions=8000)$value

  up_lnorm = c(p_lnorm[1]*tp^p_lnorm[2],p_lnorm[1]*tp^p_lnorm[2])
  lo_lnorm = c(p_lnorm[1]*tp^p_lnorm[2]+1,p_lnorm[1]*tp^p_lnorm[2] +1)
  nder_lnorm = p_lnorm[2]*p_lnorm[1]*tp^(p_lnorm[2]-1)
  psi_lnorm = digamma(p_lnorm[1]*tp^p_lnorm[2])
  fun_lnorm = genhypergeo(up_lnorm,lo_lnorm,-p_lnorm[3],tol=0, maxiter =
              2000, check_mod=TRUE,polynomial=FALSE, debug=FALSE, series
              =TRUE)
  pdf_x_lnorm[i] = (nder_lnorm*(psi_lnorm-log(p_lnorm[3]))
                   *pgamma(p_lnorm[3],p_lnorm[1]*tp^p_lnorm[2]) +
                   (nder_lnorm/((p_lnorm[1]*tp^p_lnorm[2])^2
                   *gamma(p_lnorm[1]*tp^p_lnorm[2])))*((p_lnorm[3])
                   ^(p_lnorm[1]*tp^p_lnorm[2]))*fun_lnorm)
  pdf_z_lnorm[i] = integrate(funk2_lnorm, lower = 0, upper = Inf, p =
                   p_lnorm, t = tp, up = up_lnorm, lo = lo_lnorm, nder =
                   nder_lnorm, psi =psi_lnorm, subdivisions=10000)$value
}
# OBS: integration does not work for all datasets or parameter values.
# If not, try with a large number instead of Inf as upper limit

haz_z_unif = pdf_z_unif/SZ_unif
haz_z_exp = pdf_z_exp/SZ_exp
haz_z_gam = pdf_z_gam/SZ_gam
haz_z_lnorm = pdf_z_lnorm/SZ_lnorm

haz_x_unif = pdf_x_unif/SX_unif
haz_x_exp = pdf_x_exp/SX_exp
haz_x_gam = pdf_x_gam/SX_gam
haz_x_lnorm = pdf_x_lnorm/SX_lnorm
```

D.2.4 Functions used by the estimation function `estSemi()` (semi-competing risks)

```
lklhUnifSemi = function(param,x,z,xz,z_o,tau_o, tau){

  # Purpose: find an expression for the log-likelihood function in the
  #          uniform model for semi-competing risks data, with or
  #          without censoring
  # Input:
```

```
# - param: vector of starting values for the parameters a,b,c and A
# - x    : vector of observations of times to terminal event
# - z    : vector of observations of times to non-terminal event
# - xz   : vector of observations of times to terminal event (belonging
#          to the z's)
# - z_o  : vector of observations of times to non-terminal event
# - tau_o: vector of censored observations (belonging to the z_o's)
# - tau  : vector of censored observations
# (z_o, tau_o and tau are all = 0 if there is no censoring)
# Output:
# -  -lnL: minus the log-likelihood function, which will be minimized
#          by optim() in estSemi()

# to make the code easier to read we denote the parameters by:
a = param[1]
b = param[2]
c = param[3]
A = param[4]

m = length(x)
n = length(z)
w = length(z_o)
r = length(tau)

density_x = c()
density_zx = c()
density_tau = c()
density_zo = c()

require(hypergeo)
for(i in 1:m){
  # case 1: observe only x
  xp = x[i]
  up = c(a*xp^b,a*xp^b)
  lo = c(a*xp^b+1,a*xp^b +1)
  nder = b*a*xp^(b-1)
  psi = digamma(a*xp^b)
  fun = genhypergeo(up,lo,-c,tol=0, maxiter=2000, check_mod=TRUE,
        polynomial= FALSE, debug=FALSE, series=TRUE)
  density_x[i] = (nder*(psi-log(c))*pgamma(c,a*xp^b) +
                 (nder/((a*xp^b)^2*gamma(a*xp^b)))*((c)^(a*xp^b))*fun)
                 *(1-c/A)
}
for(j in 1:n){
  # case 2: observe z and x
  zp = z[j]
  x_zp = xz[j]
```

```
  up_z = c(a*zp^b,a*zp^b)
  lo_z = c(a*zp^b+1,a*zp^b +1)
  nder_z = b*a*zp^(b-1)
  psi_z = digamma(a*zp^b)

  up_xz = c(a*x_zp^b - a*zp^b,a*x_zp^b- a*zp^b)
  lo_xz = c((a*x_zp^b - a*zp^b)+1,(a*x_zp^b-a*zp^b) +1)
  nder_xz = b*a*x_zp^(b-1)
  psi_xz = digamma(a*x_zp^b - a*zp^b)

  density_zx[j] = integrate(unifZXfunc, lower = 0.01, upper = c, a = a,
                  b = b ,c=c, z = zp,  up_z = up_z, lo_z = lo_z, nder_z
                  = nder_z, psi_z = psi_z, A = A, x_z = x_zp,up_xz =
                  up_xz, lo_xz = lo_xz, nder_xz = nder_xz, psi_xz =
                  psi_xz)$value
}
if(length(z_o) ==1 && z_o ==0){
  density_zo = 1
}else{
for(l in 1:w){
  # case 3: observe z_o and censoring time tau_o
  z_op = z_o[l]
  tau_op = tau_o[l]

  up = c(a*z_op^b,a*z_op^b)
  lo = c(a*z_op^b+1,a*z_op^b +1)
  nder = b*a*z_op^(b-1)
  psi = digamma(a*z_op^b)

  density_zo[l] = integrate(unifZOfunc, lower= 0.01, upper = c, a = a,
                  b = b, c = c, z= z_op, tau = tau_op, up = up, lo =
                  lo, nder = nder, psi = psi, A = A)$value
}
}
if(length(tau) ==1 && tau ==0){
  density_tau = 1
}else{
for(k in 1:r){
  # case 4: observe only tau
  tau_p = tau[k]
  integral = integrate(unifTAUfunc, lower = 0.01, upper = c, a = a, b =
             b, tau = tau_p, A=A)$value
  density_tau[k] = (1-c/A)*pgamma(c,a*tau_p^b) + integral
}
}

-(sum(log(density_x))+sum(log(density_zx))+ sum(log(density_zo)) +
  sum(log(density_tau)))
```

```
}
unifTAUfunc = function(s, a,b,tau,A){
  pgamma(s, a*tau^b)*1/A
}
unifZOfunc = function(s, a, b, c, z, tau, up, lo, nder, psi, A){
  (nder*(psi-log(s))*pgamma(s,a*z^b) +
  (nder/((a*z^b)^2*gamma(a*z^b)))*((s)^(a*z^b))*genhypergeo(up,lo,-s,tol=0,
  maxiter=2000, check_mod=TRUE,polynomial=FALSE,debug=FALSE, series=TRUE))
  *pgamma(c-s, a*tau^b - a*z^b)*1/A
}
unifZXfunc = function(s,a, b,c, z, up_z, lo_z, nder_z, psi_z,A, x_z,up_xz,
                      lo_xz, nder_xz, psi_xz){
  (nder_z*(psi_z-log(s))*pgamma(s,a*z^b) +
  (nder_z/((a*z^b)^2*gamma(a*z^b)))*((s)^(a*z^b))*genhypergeo(up_z,lo_z,
  -s,tol=0, maxiter=2000, check_mod=TRUE,polynomial=FALSE, debug=FALSE,
  series=TRUE))*(nder_xz*(psi_xz-log(c-s))*pgamma(c-s,a*x_z^b- a*z^b) +
  (nder_xz/((a*x_z^b-a*z^b)^2*gamma(a*x_z^b-a*z^b)))
  *((c-s)^(a*x_z^b-a*z^b))*genhypergeo(up_xz,lo_xz,-(c-s),tol=0,
  maxiter=2000, check_mod=TRUE,polynomial=FALSE, debug=FALSE,
  series=TRUE))*1/A
}


lklhExpSemi = function(param,x,z,xz,z_o,tau_o, tau){

  # Purpose: find an expression for the log-likelihood function in the
  #          exponential model for semi-competing risks data, with or
  #          without censoring
  # Input:
  # - param: vector of starting values for the parameters a,b,c and
  #          lambda_S
  # - x    : vector of observations of times to terminal event
  # - z    : vector of observations of times to non-terminal event
  # - xz   : vector of observations of times to terminal event (belonging
  #          to the z's)
  # - z_o  : vector of observations of times to non-terminal event
  # - tau_o: vector of censored observations (belonging to the z_o's)
  # - tau  : vector of censored observations
  # (z_o, tau_o and tau are 0 if there is no censoring)
  # Output:
  # -  -lnL: minus the log-likelihood function, which will be minimized
  #          by optim() in estSemi()

  # to make the code easier to read we denote the parameters by:
  a = param[1]
  b = param[2]
  c = param[3]
  lambda_s = param[4]
```

```
m = length(x)
n = length(z)
w = length(z_o)
r = length(tau)

density_x = c()
density_zx = c()
density_tau = c()
density_zo = c()

require(hypergeo)
for(i in 1:m){
  # case 1: observe only x
  xp = x[i]
  up = c(a*xp^b,a*xp^b)
  lo = c(a*xp^b+1,a*xp^b +1)
  nder = b*a*xp^(b-1)
  psi = digamma(a*xp^b)
  fun = genhypergeo(up,lo,-c,tol=0, maxiter=2000, check_mod=TRUE,
        polynomial=FALSE, debug=FALSE, series=TRUE)
  density_x[i] = (nder*(psi-log(c))*pgamma(c,a*xp^b) +
                 (nder/((a*xp^b)^2*gamma(a*xp^b)))*((c)^(a*xp^b))*fun)
                 *(1-pexp(c,lambda_s))
}
for(j in 1:n){
  # case 2: observe z and x
  zp = z[j]
  x_zp = xz[j]

  up_z = c(a*zp^b,a*zp^b)
  lo_z = c(a*zp^b+1,a*zp^b +1)
  nder_z = b*a*zp^(b-1)
  psi_z = digamma(a*zp^b)

  up_xz = c(a*x_zp^b - a*zp^b,a*x_zp^b- a*zp^b)
  lo_xz = c((a*x_zp^b - a*zp^b)+1,(a*x_zp^b-a*zp^b) +1)
  nder_xz = b*a*x_zp^(b-1)
  psi_xz = digamma(a*x_zp^b - a*zp^b)

  density_zx[j] = integrate(expZXfunc, lower = 0.01, upper = c, a = a,
                  b = b, c=c, z = zp,  up_z = up_z, lo_z = lo_z, nder_z
                  = nder_z, psi_z = psi_z, lambda_s = lambda_s, x_z =
                  x_zp,up_xz = up_xz, lo_xz = lo_xz, nder_xz = nder_xz,
                  psi_xz = psi_xz)$value
}
if(length(z_o) ==1 && z_o ==0){
  density_zo = 1
}else{
```

```
  for(l in 1:w){
    # case 3: observe z and censoring time tau
    z_op = z_o[l]
    tau_op = tau_o[l]

    up = c(a*z_op^b,a*z_op^b)
    lo = c(a*z_op^b+1,a*z_op^b +1)
    nder = b*a*z_op^(b-1)
    psi = digamma(a*z_op^b)

    density_zo[l] = integrate(expZOfunc, lower= 0.01, upper = c, a = a,
                    b = b, c = c, z= z_op, tau = tau_op, up = up, lo = lo,
                    nder = nder, psi = psi, lambda_s =lambda_s)$value
  }
  }
  if(length(tau) ==1 && tau ==0){
    density_tau = 1
  }else{
  for(k in 1:r){
    # case 4: observe only tau
    tau_p = tau[k]
    integral = integrate(expTAUfunc, lower = 0.01, upper = c, a = a, b = b,
               tau = tau_p, lambda_s=lambda_s)$value
    density_tau[k] = (1-pexp(c,lambda_s))*pgamma(c,a*tau_p^b) + integral
  }
  }

  -(sum(log(density_x))+sum(log(density_zx))+ sum(log(density_zo)) +
   sum(log(density_tau)))
}
expTAUfunc = function(s, a,b,tau,lambda_s){
  pgamma(s, a*tau^b)*dexp(s,lambda_s)
}
expZOfunc = function(s, a, b, c, z, tau, up, lo, nder, psi, lambda_s){
  (nder*(psi-log(s))*pgamma(s,a*z^b) + (nder/((a*z^b)^2*gamma(a*z^b)))*
  ((s)^(a*z^b))*genhypergeo(up,lo,-s,tol=0, maxiter=2000, check_mod=TRUE,
  polynomial=FALSE,debug=FALSE, series=TRUE))*pgamma(c-s, a*tau^b - a*z^b)
  *dexp(s,lambda_s)
}
expZXfunc = function(s,a, b,c, z, up_z, lo_z, nder_z, psi_z,lambda_s, x_z,
            up_xz, lo_xz, nder_xz, psi_xz){
  (nder_z*(psi_z-log(s))*pgamma(s,a*z^b)+(nder_z/((a*z^b)^2*gamma(a*z^b)))
  *((s)^(a*z^b))*genhypergeo(up_z,lo_z,-s,tol=0,
  maxiter=2000,check_mod=TRUE,polynomial=FALSE,debug=FALSE,series=TRUE))*
  (nder_xz*(psi_xz-log(c-s))*pgamma(c-s,a*x_z^b- a*z^b) +
  (nder_xz/((a*x_z^b-a*z^b)^2*gamma(a*x_z^b-a*z^b)))
  *((c-s)^(a*x_z^b-a*z^b))*genhypergeo(up_xz,lo_xz,-(c-s),tol=0, maxiter=
  2000, check_mod=TRUE,polynomial=FALSE, debug=FALSE, series=TRUE))
```

```
  *dexp(s,lambda_s)
}


lklhGamSemi = function(param,x,z,xz,z_o,tau_o, tau){

  # Purpose: find an expression for the log-likelihood function in the
  #          gamma model for semi-competing risks data, with or
  #          without censoring
  # Input:
  # - param: vector of starting values for the parameters a,b,c,alpha_S
  #          and beta_S
  # - x    : vector of observations of times to terminal event
  # - z    : vector of observations of times to non-terminal event
  # - xz   : vector of observations of times to terminal event (belonging
  #          to the z's)
  # - z_o  : vector of observations of times to non-terminal event
  # - tau_o: vector of censored observations (belonging to the z_o's)
  # - tau  : vector of censored observations
  # (z_o, tau_o and tau are 0 if there is no censoring)
  # Output:
  # -  -lnL: minus the log-likelihood function, which will be minimized
  #          by optim() in estSemi()

  # to make the code easier to read we denote the parameters by:
  a = param[1]
  b = param[2]
  c = param[3]
  alpha_s = param[4]
  beta_s = param[5]

  m = length(x)
  n = length(z)
  w = length(z_o)
  r = length(tau)

  density_x = c()
  density_zx = c()
  density_tau = c()
  density_zo = c()

  require(hypergeo)
  for(i in 1:m){
    #case 1: observe only x
    xp = x[i]
    up = c(a*xp^b,a*xp^b)
    lo = c(a*xp^b+1,a*xp^b +1)
    nder = b*a*xp^(b-1)
    psi = digamma(a*xp^b)
```

```
  fun = genhypergeo(up,lo,-c,tol=0, maxiter=2000, check_mod=TRUE,
        polynomial= FALSE, debug=FALSE, series=TRUE)
  density_x[i] = (nder*(psi-log(c))*pgamma(c,a*xp^b) +
                 (nder/((a*xp^b)^2*gamma(a*xp^b)))*((c)^(a*xp^b))*fun)
                 *(1-pgamma(c,alpha_s,beta_s))
}
for(j in 1:n){
  # case 2: observe z and x
  zp = z[j]
  x_zp = xz[j]

  up_z = c(a*zp^b,a*zp^b)
  lo_z = c(a*zp^b+1,a*zp^b +1)
  nder_z = b*a*zp^(b-1)
  psi_z = digamma(a*zp^b)

  up_xz = c(a*x_zp^b - a*zp^b,a*x_zp^b- a*zp^b)
  lo_xz = c((a*x_zp^b - a*zp^b)+1,(a*x_zp^b-a*zp^b) +1)
  nder_xz = b*a*x_zp^(b-1)
  psi_xz = digamma(a*x_zp^b - a*zp^b)

  density_zx[j] = integrate(gamZXfunc, lower = 0.01, upper = c, a = a,
                  b = b, c = c, z = zp,  up_z = up_z, lo_z = lo_z,
                  nder_z = nder_z, psi_z = psi_z, alpha_s = alpha_s,
                  beta_s = beta_s, x_z = x_zp, up_xz = up_xz, lo_xz =
                  lo_xz, nder_xz = nder_xz, psi_xz = psi_xz)$value
}
if(length(z_o) ==1 && z_o ==0){
  density_zo = 1
}else{
for(l in 1:w){
  # case 3: observe z and censoring time tau
  z_op = z_o[l]
  tau_op = tau_o[l]

  up = c(a*z_op^b,a*z_op^b)
  lo = c(a*z_op^b+1,a*z_op^b +1)
  nder = b*a*z_op^(b-1)
  psi = digamma(a*z_op^b)

  density_zo[l] = integrate(gamZOfunc, lower= 0.01, upper = c, a = a,
                  b = b, c = c, z= z_op, tau = tau_op, up = up, lo =
                  lo, nder = nder, psi = psi, alpha_ =alpha_s, beta_s
                  = beta_s)$value
}
}
if(length(tau) ==1 && tau ==0){
  density_tau = 1
```

```
  }else{
  for(k in 1:r){
    # case 4: observe only tau
    tau_p = tau[k]
    integral = integrate(gamTAUfunc, lower = 0.01, upper = c, a = a,b = b,
               tau = tau_p, alpha_s=alpha_s, beta_s = beta_s)$value
    density_tau[k] = (1-pgamma(c,alpha_s,beta_s))*pgamma(c,a*tau_p^b)
                   + integral
  }
  }

  -(sum(log(density_x))+sum(log(density_zx))+ sum(log(density_zo))
  + sum(log(density_tau)))
}

gamTAUfunc = function(s, a,b,tau,alpha_s,beta_s){
  pgamma(s, a*tau^b)*dgamma(s,alpha_s,beta_s)
}
gamZOfunc = function(s, a, b, c, z, tau,up,lo,nder,psi,alpha_s,beta_s){
  (nder*(psi-log(s))*pgamma(s,a*z^b) + (nder/((a*z^b)^2*gamma(a*z^b)))*
  ((s)^(a*z^b))*genhypergeo(up,lo,-s,tol=0, maxiter=2000, check_mod=TRUE,
  polynomial=FALSE,debug=FALSE,series=TRUE))*pgamma(c-s, a*tau^b - a*z^b)
  *dgamma(s,alpha_s, beta_s)
}
gamZXfunc = function(s,a, b,c, z, up_z, lo_z, nder_z, psi_z, alpha_s,
                     beta_s, x_z, up_xz,  lo_xz, nder_xz, psi_xz){
  (nder_z*(psi_z-log(s))*pgamma(s,a*z^b) + (nder_z/((a*z^b)^2
  *gamma(a*z^b)))*((s)^(a*z^b))*genhypergeo(up_z,lo_z,-s,tol=0, maxiter
  =2000, check_mod=TRUE,polynomial=FALSE, debug=FALSE, series=TRUE))
  *(nder_xz*(psi_xz-log(c-s))*pgamma(c-s,a*x_z^b- a*z^b) +
  (nder_xz/((a*x_z^b-a*z^b)^2*gamma(a*x_z^b-a*z^b)))
  *((c-s)^(a*x_z^b-a*z^b))*genhypergeo(up_xz,lo_xz,-(c-s),tol=0,maxiter
  =2000,check_mod=TRUE,polynomial=FALSE, debug=FALSE, series=TRUE))
   *dgamma(s,alpha_s,beta_s)
}


lklhLnormSemi = function(param,x,z,xz,z_o,tau_o, tau){

  # Purpose: find an expression for the log-likelihood function in the
  #          lognormal model for semi-competing risks data, with or
  #          without censoring
  # Input:
  # - param: vector of starting values for the parameters a,b,c,mu_S
  #          and sigma_S
  # - x    : vector of observations of times to terminal event
  # - z    : vector of observations of times to non-terminal event
  # - xz   : vector of observations of times to terminal event (belonging
  #          to the z's)
```

```
# - z_o  : vector of observations of times to non-terminal event
# - tau_o: vector of censored observations (belonging to the z_o's)
# - tau  : vector of censored observations
# (z_o, tau_o and tau are 0 if there is no censoring)
# Output:
# -  -lnL: minus the log-likelihood function, which will be minimized
#          by optim() in estSemi()

# to make the code easier to read we denote the parameters by:
a = param[1]
b = param[2]
c = param[3]
mu_s = param[4]
sigma_s = param[5]

m = length(x)
n = length(z)
w = length(z_o)
r = length(tau)

density_x = c()
density_zx = c()
density_tau = c()
density_zo = c()

require(hypergeo)
for(i in 1:m){
  # case 1: observe only x
  xp = x[i]
  up = c(a*xp^b,a*xp^b)
  lo = c(a*xp^b+1,a*xp^b +1)
  nder = b*a*xp^(b-1)
  psi = digamma(a*xp^b)
  fun = genhypergeo(up,lo,-c,tol=0, maxiter=2000, check_mod=TRUE,
        polynomial= FALSE, debug=FALSE, series=TRUE)
  density_x[i] = (nder*(psi-log(c))*pgamma(c,a*xp^b) +
                 (nder/((a*xp^b)^2*gamma(a*xp^b)))*((c)^(a*xp^b))*fun)
                 *(1-plnorm(c,mu_s,sigma_s))
}
for(j in 1:n){
  # case 2: observe z and x
  zp = z[j]
  x_zp = xz[j]

  up_z = c(a*zp^b,a*zp^b)
  lo_z = c(a*zp^b+1,a*zp^b +1)
  nder_z = b*a*zp^(b-1)
  psi_z = digamma(a*zp^b)
```

```
    up_xz = c(a*x_zp^b - a*zp^b,a*x_zp^b- a*zp^b)
    lo_xz = c((a*x_zp^b - a*zp^b)+1,(a*x_zp^b-a*zp^b) +1)
    nder_xz = b*a*x_zp^(b-1)
    psi_xz = digamma(a*x_zp^b - a*zp^b)

    density_zx[j] = integrate(lnormZXfunc,lower = 0.01, upper = c, a = a,
                    b = b,c=c, z = zp,  up_z = up_z, lo_z = lo_z, nder_z
                    = nder_z, psi_z = psi_z,mu_s=mu_s, sigma_s = sigma_s,
                    x_z = x_zp, up_xz = up_xz, lo_xz = lo_xz, nder_xz =
                    nder_xz, psi_xz = psi_xz)$value
  }
  if(length(z_o) ==1 && z_o ==0){
    density_zo = 1
  }else{
  for(l in 1:w){
    # case 3: observe z and censoring time tau
    z_op = z_o[l]
    tau_op = tau_o[l]

    up = c(a*z_op^b,a*z_op^b)
    lo = c(a*z_op^b+1,a*z_op^b +1)
    nder = b*a*z_op^(b-1)
    psi = digamma(a*z_op^b)

    density_zo[l] = integrate(lnormZOfunc, lower= 0.01, upper = c, a = a,
                    b = b, c = c, z= z_op, tau = tau_op, up = up, lo = lo,
                    nder = nder, psi = psi, mu_s = mu_s, sigma_ s=
                    sigma_s)$value
  }
  }
  if(length(tau) ==1 && tau ==0){
    density_tau = 1
  }else{
  for(k in 1:r){
    # case 4: observe only tau
    tau_p = tau[k]
    integral = integrate(lnormTAUfunc, lower = 0.01, upper = c, a = a,
               b = b, tau = tau_p, mu_s=mu_s, sigma_s = sigma_s)$value
    density_tau[k] = (1-plnorm(c,mu_s,sigma_s))*pgamma(c,a*tau_p^b)
                     + integral
  }
  }

  -(sum(log(density_x))+sum(log(density_zx))+ sum(log(density_zo))
  + sum(log(density_tau)))
}
lnormTAUfunc = function(s, a,b,tau,mu_s,sigma_s){
```

```
  pgamma(s, a*tau^b)*dlnorm(s,mu_s,sigma_s)
}
lnormZOfunc = function(s, a, b, c, z, tau, up, lo,nder,psi,mu_s,sigma_s){
  (nder*(psi-log(s))*pgamma(s,a*z^b) + (nder/((a*z^b)^2*gamma(a*z^b)))
  *((s)^(a*z^b))*genhypergeo(up,lo,-s,tol=0, maxiter=2000, check_mod=TRUE,
  polynomial=FALSE,debug=FALSE,series=TRUE))*pgamma(c-s, a*tau^b - a*z^b)
  *dlnorm(s,mu_s, sigma_s)
}
lnormZXfunc = function(s,a, b,c, z, up_z, lo_z, nder_z, psi_z, mu_s,
                       sigma_s, x_z, up_xz,  lo_xz, nder_xz, psi_xz){
  (nder_z*(psi_z-log(s))*pgamma(s,a*z^b) + (nder_z/((a*z^b)^2
  *gamma(a*z^b)))*((s)^(a*z^b))*genhypergeo(up_z,lo_z,-s,tol=0, maxiter
  =2000, check_mod=TRUE,polynomial=FALSE, debug=FALSE, series=TRUE))
  *(nder_xz*(psi_xz-log(c-s))*pgamma(c-s,a*x_z^b- a*z^b) +
  (nder_xz/((a*x_z^b-a*z^b)^2*gamma(a*x_z^b-a*z^b)))*((c-s)^(a*x_z^b-
  a*z^b))*genhypergeo(up_xz,lo_xz,-(c-s),tol=0, maxiter=2000,
  check_mod=TRUE,polynomial=FALSE, debug=FALSE, series=TRUE))
  *dlnorm(s,mu_s,sigma_s)
}


lklhNormSemi = function(param,x,z,xz,z_o,tau_o,tau){

  # Purpose: find an expression for the log-likelihood function in the
  #          normal model for semi-competing risks data, with or
  #          without censoring
  # Input:
  # - param: vector of starting values for the parameters a,b,c,mu_S
  #          and sigma_S
  # - x    : vector of observations of times to terminal event
  # - z    : vector of observations of times to non-terminal event
  # - xz   : vector of observations of times to terminal event (belonging
  #          to the z's)
  # - z_o  : vector of observations of times to non-terminal event
  # - tau_o: vector of censored observations (belonging to the z_o's)
  # - tau  : vector of censored observations
  # (z_o, tau_o and tau are 0 if there is no censoring)
  # Output:
  # -  -lnL: minus the log-likelihood function, which will be minimized
  #          by optim() in estSemi()

  # to make the code easier to read we denote the parameters by:
  a = param[1]
  b = param[2]
  c = param[3]
  mu_s = param[4]
  sigma_s = param[5]

  m = length(x)
```

```
n = length(z)
w = length(z_o)
r = length(tau)

density_x = c()
density_zx = c()
density_tau = c()
density_zo = c()

require(hypergeo)
for(i in 1:m){
  # case 1: observe only x
  xp = x[i]
  up = c(a*xp^b,a*xp^b)
  lo = c(a*xp^b+1,a*xp^b +1)
  nder = b*a*xp^(b-1)
  psi = digamma(a*xp^b)
  fun = genhypergeo(up,lo,-c,tol=0, maxiter=2000, check_mod=TRUE,
        polynomial= FALSE, debug=FALSE, series=TRUE)
  density_x[i] = (nder*(psi-log(c))*pgamma(c,a*xp^b) +
                 (nder/((a*xp^b)^2*gamma(a*xp^b)))*((c)^(a*xp^b))*fun)
                 *(1-pnorm(c,mu_s,sigma_s))
}
for(j in 1:n){
  # case 2: observe z and x
  zp = z[j]
  x_zp = xz[j]

  up_z = c(a*zp^b,a*zp^b)
  lo_z = c(a*zp^b+1,a*zp^b +1)
  nder_z = b*a*zp^(b-1)
  psi_z = digamma(a*zp^b)

  up_xz = c(a*x_zp^b - a*zp^b,a*x_zp^b- a*zp^b)
  lo_xz = c((a*x_zp^b - a*zp^b)+1,(a*x_zp^b-a*zp^b) +1)
  nder_xz = b*a*x_zp^(b-1)
  psi_xz = digamma(a*x_zp^b - a*zp^b)

  density_zx[j] = integrate(normZXfunc,lower = 0.01, upper = c, a = a,
                  b = b,c=c, z = zp,  up_z = up_z, lo_z = lo_z, nder_z
                  = nder_z, psi_z = psi_z,mu_s=mu_s, sigma_s = sigma_s,
                  x_z = x_zp, up_xz = up_xz, lo_xz = lo_xz, nder_xz =
                  nder_xz, psi_xz = psi_xz)$value
}
if(length(z_o) ==1 && z_o ==0){
  density_zo = 1
}else{
for(l in 1:w){
```

```
    # case 3: observe z and censoring time tau
    z_op = z_o[l]
    tau_op = tau_o[l]

    up = c(a*z_op^b,a*z_op^b)
    lo = c(a*z_op^b+1,a*z_op^b +1)
    nder = b*a*z_op^(b-1)
    psi = digamma(a*z_op^b)

    density_zo[l] = integrate(normZOfunc, lower= 0.01, upper = c, a = a,
                    b = b, c = c, z= z_op, tau = tau_op, up = up, lo = lo,
                    nder = nder, psi = psi, mu_s = mu_s, sigma_ s=
                    sigma_s)$value
  }
  }
  if(length(tau) ==1 && tau ==0){
    density_tau = 1
  }else{
  for(k in 1:r){
    # case 4: observe only tau
    tau_p = tau[k]
    integral = integrate(normTAUfunc, lower = 0.01, upper = c, a = a,
               b = b, tau = tau_p, mu_s=mu_s, sigma_s = sigma_s)$value
    density_tau[k] = (1-pnorm(c,mu_s,sigma_s))*pgamma(c,a*tau_p^b)
                     + integral
  }
  }

  -(sum(log(density_x))+sum(log(density_zx))+ sum(log(density_zo))
  + sum(log(density_tau)))
}
normTAUfunc = function(s, a,b,tau,mu_s,sigma_s){
  pgamma(s, a*tau^b)*dnorm(s,mu_s,sigma_s)
}
normZOfunc = function(s, a, b, c, z, tau, up, lo,nder,psi,mu_s,sigma_s){
  (nder*(psi-log(s))*pgamma(s,a*z^b) + (nder/((a*z^b)^2*gamma(a*z^b)))
  *((s)^(a*z^b))*genhypergeo(up,lo,-s,tol=0, maxiter=2000, check_mod=TRUE,
  polynomial=FALSE,debug=FALSE,series=TRUE))*pgamma(c-s, a*tau^b - a*z^b)
  *dnorm(s,mu_s, sigma_s)
}
normZXfunc = function(s,a, b,c, z, up_z, lo_z, nder_z, psi_z, mu_s,
                      sigma_s, x_z, up_xz,  lo_xz, nder_xz, psi_xz){
  (nder_z*(psi_z-log(s))*pgamma(s,a*z^b) + (nder_z/((a*z^b)^2
  *gamma(a*z^b)))*((s)^(a*z^b))*genhypergeo(up_z,lo_z,-s,tol=0, maxiter
  =2000, check_mod=TRUE,polynomial=FALSE, debug=FALSE, series=TRUE))
  *(nder_xz*(psi_xz-log(c-s))*pgamma(c-s,a*x_z^b- a*z^b) +
  (nder_xz/((a*x_z^b-a*z^b)^2*gamma(a*x_z^b-a*z^b)))*((c-s)^(a*x_z^b-
  a*z^b))*genhypergeo(up_xz,lo_xz,-(c-s),tol=0, maxiter=2000,
```

```
  check_mod=TRUE,polynomial=FALSE, debug=FALSE, series=TRUE))
  *dnorm(s,mu_s,sigma_s)
}
```

D.2.5 Bootstrapping

```
## non-parametric bootstrapping for the VHF data
# requires the sorted VHF data, here denoted data.sort_vhf
# column 1 contains event times t
# column 2 contains cause variable C, where C =1 means failure,
# C = 2 means PM and C = 3 means censoring

# number of bootstrap samples
B = 500
bootestimates = c()
b = 0
while(b  < B){
  # sample index values
    ind = sample(seq(1,369),369,replace =TRUE)
  # make bootstrap data sample
    nydata = data.sort_vhf[ind,]
  N = length(nydata[,1])
  C = nydata[,2]
  t = nydata[,1]

  x = c()
  z = c()
  tau = c()

  tellerx = 1
  tellerz = 1
  tellertau = 1

  # make the vectors x and z from the indexing in the data
  for(i in 1:N){
    if(C[i] == 1){
      x[tellerx] = t[i]
      tellerx = tellerx +1
    } else if (C[i] == 2){
      z[tellerz] = t[i]
      tellerz = tellerz + 1
    } else{
      tau[tellertau] = t[i]
      tellertau = tellertau + 1
    }
  }
  if(is.null(tau)==T){ tau = 0}

  #vhf start-values (gamma model)
```

```
  p0 = c(4.65,0.24,16.7,122,6.97)

  #ERROR HANDLING
  out <- tryCatch({
    # gamma model
    opt = optim(par = p0 , fn=lklhGam, x = x, z=z, tau = tau, method =
    "L-BFGS-B", lower = c(4, 0.15, 15, 116, 5.5),
    upper = c(5.8,0.3,19,125, 8.0),control =list(parscale =
    c(1,0.1,1,10,1)),hessian=T)
  },
  error=function(e) {
    message("could not estimate")
    return(NULL)
  },
  finally = {
    message("trial completed")
  }
  )
  if(is.null(out)==F){
    message("estimation successful")
    p = out$par
    bootestimates = rbind(bootestimates,p)
    b = b+1
  }
}

## non-parametric bootstrapping of carcinoma data

# requires the carcinoma data as a table where each
# individual's data is registered in a row.

# first column = time to the non-terminal event Z,
# second column = time to the terminal event X_Z.
# if only the terminal event is observed:
# first column = second column = X
# third column = delta, indicator variable which = 1 if the non-terminal
# event is observed and = 0 else

# number of bootstrap samples
B = 500
bootestimates = c()
b = 0
while(b  < B){
  #sample index values without replacement
  ind = sample(seq(1,61),61,replace =TRUE)
  # make bootstrap data sample
  nydata = data[ind,]
  N = length(nydata[,1])
```

```
x = c()
z = c()
xz = c()

tellerx = 1
tellerz = 1

# make the vectors x and z from the indexing in the data
for(i in 1:N){
  if(nydata[i,3] == 0){
    x[tellerx] = nydata[i,1]
    tellerx = tellerx +1
  } else if (nydata[i,3] == 1){
    z[tellerz] = nydata[i,1]
    xz[tellerz] = nydata[i,2]
    tellerz = tellerz + 1
  }
}

m = length(x)
n = length(z)

# carcinoma start-values
# for the gamma model use
# p0 = c(2.18,0.36,5.66,5.93,0.95)
# for the lognormal model use
p0 =c(1.84,0.38,5.00,1.66,0.48)

#ERROR HANDLING
out <- tryCatch({
 # gamma model
 # opt = optim(par = p0, fn=lklhGamSemi, x = x, z=z, xz = xz, z_o = 0,
 tau_o = 0, tau = 0,method = "L-BFGS-B", lower=c(0.1,0.2,0.1,0.1,0.1),
 upper= c(10,0.8,15,15,2.9), hessian=T)
 # lognormal model
  opt = optim(par = p0, fn=lklhLnormSemi, x = x, z=z, xz = xz,z_o = 0,
  tau_o=0, tau=0, method = "L-BFGS-B", lower=c(0.1,0.2,0.1,0.001,0.1),
  upper= c(8,0.9,16,3,1),hessian=T)
},
error=function(e) {
  message("could not estimate")
  return(NULL)
},
finally = {
  message("trial completed")
}
)
```

```
 if(is.null(out)==F){
   message("estimation successful")
   p = out$par
   bootestimates = rbind(bootestimates,p)
   b = b+1
 }
}


boot.BCa = function(x ,th0, th, stat, conf = 0.95){

  # Purpose: calculate BCa intervals
  # Input:
  # x - table of data, e.g. carcinoma data or vhf data
  # th0 - observed parameter values from data
  # th - matrix of bootstrap replicates
  #      if gamma model:
  #      col1: alpha, col2: beta, col3: c, col4: alpha_s col5: beta_s
  #      if lognormal model:
  #      col1: alpha, col2: beta, col3: c, col4: mu_s col5: sigma_s
  # stat - function that calculates the parameter estimates
  # Output:
  # list of original parameter estimates and the estimated
  # BCa confidence intervals

  n = nrow(x) # number of observations
  B = 500 # number of bootstrap replicates

  alpha = (1 + c(-conf, conf))/2
  zalpha = qnorm(alpha)

  #bias correction factor for each parameter
  b1 = qnorm(sum(th[,1] < th0[1])/B)
  b2 = qnorm(sum(th[,2] < th0[2])/B)
  b3 = qnorm(sum(th[,3] < th0[3])/B)
  b4 = qnorm(sum(th[,4] < th0[4])/B)
  b5 = qnorm(sum(th[,5] < th0[5])/B)

  #acceleration factor for each parameter
  sub_est = c()
  for(i in 1:n){
    sub_est = rbind(sub_est,stat(x[-i,]))
  }
  L1 = mean(sub_est[,1]) - sub_est[,1]
  L2 = mean(sub_est[,2]) - sub_est[,2]
  L3 = mean(sub_est[,3]) - sub_est[,3]
  L4 = mean(sub_est[,4]) - sub_est[,4]
  L5 = mean(sub_est[,5]) - sub_est[,5]
  a1 = sum(L1^3)/(6*sum(L1^2)^1.5)
```

```
  a2 = sum(L2^3)/(6*sum(L2^2)^1.5)
  a3 = sum(L3^3)/(6*sum(L3^2)^1.5)
  a4 = sum(L4^3)/(6*sum(L4^2)^1.5)
  a5 = sum(L5^3)/(6*sum(L5^2)^1.5)

  #BCa conf limits
  beta1 = pnorm(b1 + (b1 + zalpha)/(1 - a1*(b1 + zalpha)))
  beta2 = pnorm(b2 + (b2 + zalpha)/(1 - a2*(b2 + zalpha)))
  beta3 = pnorm(b3 + (b3 + zalpha)/(1 - a3*(b3 + zalpha)))
  beta4 = pnorm(b4 + (b4 + zalpha)/(1 - a4*(b4 + zalpha)))
  beta5 = pnorm(b5 + (b5 + zalpha)/(1 - a5*(b5 + zalpha)))
  limits1 = quantile(th[,1], beta1, type = 6)
  limits2 = quantile(th[,2], beta2, type = 6)
  limits3 = quantile(th[,3], beta3, type = 6)
  limits4 = quantile(th[,4], beta4, type = 6)
  limits5 = quantile(th[,5], beta5, type = 6)
  return(list("est1" = th0[1], "BCa1" = limits1,
              "est2" = th0[2], "BCa2" = limits2,
              "est3" = th0[3], "BCa3" = limits3,
              "est4" = th0[4], "BCa4" = limits4,
              "est5" = th0[5], "BCa5" = limits5))

}
# two alternative definitions of the function stat():
# one for ordinary competing risks (VHF-data):
stat = function(x){

  # Purpose: computes the statistic theta, in our case alpha, beta,
  #          c, and the parameters in the distribution of S for
  #          ordinary competing risks data
  # Input:
  # x - reduced sample
  # Output:
  # p - vector of parameter estimates

  nydata = x
  N = length(nydata[,1])
  C = nydata[,2]
  t = nydata[,1]

  x = c()
  z = c()
  tau = c()

  tellerx = 1
  tellerz = 1
  tellertau = 1
```

```
  # make the vectors x, z and tau from the indexing in the data
  for(i in 1:N){
    if(C[i] == 1){
      x[tellerx] = t[i]
      tellerx = tellerx +1
    } else if (C[i] == 2){
      z[tellerz] = t[i]
      tellerz = tellerz + 1
    } else{
      tau[tellertau] = t[i]
      tellertau = tellertau + 1
    }
  }
  if(is.null(tau)==T){ tau = 0}

  #vhf start-values (gamma model)
  p0 = c(4.65,0.24,16.7,122,6.97)

  # error handling
  estOK = 0
  while(estOK < 1){
    out <- tryCatch({
     # VHF gamma model
      opt = optim(par = p0 , fn=lklhGam, x = x, z=z, tau = tau,method =
      "L-BFGS-B", lower = c(4, 0.15, 15, 116, 5.5),
      upper = c(5.8,0.3,19,125,8.0), control =list(parscale =
      c(1,0.1,1,10,1)),hessian=T)
    },
    error=function(e) {
      message("estimation failed")
      return(NULL)
    },
    finally = {
      message("trial completed")
    }
    )
    if(is.null(out)==F){
      message("estimation successful")
      p = out$par
      estOK = 1
    }
  }
  return(p)
}
# and one for semi-competing risks data without censoring (carcinoma):
stat = function(x){

  # Purpose: computes the statistic theta, in our case alpha, beta,
```

```
  #          c, and the parameters in the distribution of S for
  #          semi-competing risks data
  # Input:
  # x - reduced sample
  # Output:
  # p - vector of parameter estimates

  nydata = x
  N = length(nydata[,1])

  x = c()
  z = c()
  xz= c()

  tellerx = 1
  tellerz = 1

  # make the vectors x,z and xz from the indexing in the data
  for(i in 1:N){
    if(nydata[i,3] == 0){
      x[tellerx] = nydata[i,1]
      tellerx = tellerx +1
    } else if (nydata[i,3] == 1){
      z[tellerz] = nydata[i,1]
      xz[tellerz] = nydata[i,2]
      tellerz = tellerz + 1
    }
  }

  m = length(x)
  n = length(z)

  # carcinoma start-values
  # gamma model
  # p0 = c(2.18,0.36,5.66,5.93,0.95)
  # lognormal model
  p0 =c(1.84,0.38,5.00,1.66,0.48)

  estOK = 0
  while(estOK < 1){
    #ERROR HANDLING
    out <- tryCatch({
    # gamma model
    # opt = optim(par = p0, fn=lklhGamSemi, x = x, z=z, xz = xz,z_o= 0,
    tau_o = 0, tau = 0, method ="L-BFGS-B", lower=c(1,0.2,1,1,0.1),
    upper= c(3,0.9,6,6,2),hessian=T)
# lognormal model
      opt = optim(par = p0,fn=lklhLnormSemi, x=x, z=z, xz = xz,z_o= 0,
```

```
      tau_o=0,tau=0 method="L-BFGS-B", lower = c(0.1,0.2,0.1,0.1,0.1),
      upper = c(8,0.8,16,3,0.8),hessian=T)
    },
    error=function(e) {
      message("could not estimate")
      return(NULL)
    },
    finally = {
      message("trial completed")
    }
    )
    if(is.null(out)==F){
      message("estimation successful")
      p = out$par
      estOK = 1
    }
  }
  return(p)
}
```

Appendix E

Output from R

In this appendix the outputs from R of the estimation done in chapters 7, 8 and 9 are displayed.

E.1 Simulation study - competing risks data

E.1.1 Uniform model

```
# simulate data from the uniform model
> data = simRandomS(c=5, a=5, b=1, A=10, type="unif",  N= 1000)
# estimate back parameters in the uniform model
> est = condSurv(data$data, p0 = c(5,1,5,10), upper = c(10,2,14,21),
  lower = c(0.1, 0.1, 0.1, 0.1), type = "unif")
> est
$par
             a          b          c          A
[1]  5.4523584  0.9922765  5.3410719 10.8534050

$std
             a          b          c          A
[1] 0.66080023 0.04452704 0.70786746 1.49632582

$logL
[1] -1064.135

$cor
               a          b          c          A
   a   1.0000000 -0.8330705  0.9908657  0.9613122
   b  -0.8330705  1.0000000 -0.8172470 -0.7944439
   c   0.9908657 -0.8172470  1.0000000  0.9696615
   A   0.9613122 -0.7944439  0.9696615  1.0000000

$lower
            a         b         c         A
[1] 4.2995289 0.9087313 4.1192124 8.2834507

$upper
            a         b         c         A
[1]  6.914295  1.083502  6.925365 14.220692
```

```
# estimate parameters in the basic model
> est = condSurv(data$data, p0 = c(5,1,5,2.5,0.5), type = "const")
> est
$par
           a         b         c         s         q
[1] 3.1295303 1.0569807 2.9402547 1.2397494 0.4916526

$std
            a          b          c          s          q
[1] 0.59378262 0.08979711 0.60534262 0.36949203 0.01656229

$logL
[1] -1087.523

$cor
                 a            b           c            s            q
   a   1.000000000 -0.946311551  0.99114616  0.989286165 -0.008901023
   b  -0.946311551  1.000000000 -0.93067702 -0.951793358  0.006570545
   c   0.991146161 -0.930677019  1.00000000  0.979165674 -0.010723955
   s   0.989286165 -0.951793358  0.97916567  1.000000000 -0.002721713
   q  -0.008901023  0.006570545 -0.01072395 -0.002721713  1.000000000

$lower
           a         b         c         s         q
[1] 2.1576110 0.8948512 1.9639730 0.6912563 0.4602390

$upper
           a         b         c         s         q
[1] 4.5392612 1.2484848 4.4018416 2.2234568 0.5252103

# estimate parameters in the lognormal model
> est = condSurv(data$data, p0 = c(11.13,0.70,11.09,2.43,0.80),
  upper = c(14,1.2,14,3,2), lower = c(6, 0.2, 6, 1,0.1), type = "lognorm")
> est
$par
           a         b         c      mu_s   sigma_s
[1] 11.128125  0.696360 11.091949  2.428168  0.803065

$std
           a         b           c       mu_s    sigma_s
[1] 2.31583549 0.07263501 2.33263146 0.21247678 0.09551545

$logL
[1] -1059.603

$cor
                 a          b          c       mu_s    sigma_s
      a   1.0000000 -0.9535293  0.9979581  0.9853627 -0.8178452
```

```
      b  -0.9535293  1.0000000 -0.9467626 -0.9322216  0.8832165
      c   0.9979581 -0.9467626  1.0000000  0.9874022 -0.8016723
   mu_s   0.9853627 -0.9322216  0.9874022  1.0000000 -0.7602878
sigma_s  -0.8178452  0.8832165 -0.8016723 -0.7602878  1.0000000

$lower
            a         b         c      mu_s   sigma_s
[1] 7.4007912 0.5676049 7.3450929 2.0454699 0.6360747

$upper
             a          b          c       mu_s    sigma_s
[1] 16.7326935  0.8543217 16.7501391  2.8824683  1.0138958
```

E.1.2 Exponential model

```
# simulate data from the exponential model
> data = simRandomS(c=7, a=5, b=1, lambda_s= 1/10, type="expon", N= 1000)
# estimate back parameters in the exponential model
> est = condSurv(data$data, p0 = c(5,1,7,1/10), upper = c(10,2,14,2),
  lower = c(0.1, 0.1, 0.1, 0.1), type = "expon")
> est
$par
             a          b          c   lambda_s
[1] 4.88950331 0.99432575 6.70684906 0.09927012

$std
             a          b          c   lambda_s
[1] 0.57471754 0.04007560 0.77282409 0.01286032

$logL
[1] -1275.883

$cor
                   a          b          c   lambda_s
       a   1.0000000 -0.8271299  0.9865128 -0.9241595
       b  -0.8271299  1.0000000 -0.7689095  0.7379054
       c   0.9865128 -0.7689095  1.0000000 -0.9300622
lambda_s  -0.9241595  0.7379054 -0.9300622  1.0000000

$lower
             a          b          c   lambda_s
[1] 3.88339664 0.91879998 5.35098278 0.07700952

$upper
            a         b         c  lambda_s
[1] 6.1562711 1.0760598 8.4062734 0.1279654

# estimate parameters in the basic model
```

```
> est = condSurv(data$data, c(5,1,7,3.095,0.5), type ="const")
> est
$par
           a         b         c         s         q
[1] 1.9437458 1.1758608 2.6866071 0.8086558 0.4837551

$std
            a          b          c          s          q
[1] 0.31773190 0.07302182 0.42138276 0.20884065 0.01640193

$logL
[1] -1314.095

$cor
                 a            b            c             s             q
   a   1.000000000 -0.919613573  0.979214577  0.9810371646 -0.0078037306
   b  -0.919613573  1.000000000 -0.861597818 -0.9098558256  0.0059129113
   c   0.979214577 -0.861597818  1.000000000  0.9583941200 -0.0093735074
   s   0.981037165 -0.909855826  0.958394120  1.0000000000 -0.0008053892
   q  -0.007803731  0.005912911 -0.009373507 -0.0008053892  1.0000000000

$lower
           a         b         c         s         q
[1] 1.4109004 1.0411054 1.9755790 0.4874514 0.4526522

$upper
           a         b         c         s         q
[1] 2.6778275 1.3280582 3.6535403 1.3415168 0.5169951
```

E.1.3 Gamma model

```
# simulate data from the gamma model
> data = simRandomS(c=7, a=5, b=1, alpha_s = 49/4, beta_s = 7/4, type=
         "gamma",  N= 1000)
# estimate back parameters in the gamma model
> est = condSurv(data$data, p0= c(5,1,7,49/4,7/4),upper = c(10,2,14,30,5),
  lower = c(0.1, 0.1, 0.1, 0.1, 0.1), type = "gamma")
> est
$par
           a         b         c   alpha_s    beta_s
[1]  3.841355  1.053434  5.458698 10.762508  1.916018

$std
           a         b         c   alpha_s    beta_s
[1] 1.0977941 0.1229922 1.3639332 3.5498189 0.5212722

$logL
[1] -1377.681
```

```
$cor
                  a           b          c    alpha_s      beta_s
      a   1.00000000 -0.97948828  0.9950966  0.6714026 -0.07583677
      b  -0.97948828  1.00000000 -0.9617708 -0.7207250 -0.01662397
      c   0.99509664 -0.96177082  1.0000000  0.6186256 -0.14635505
alpha_s   0.67140261 -0.72072498  0.6186256  1.0000000  0.68581919
 beta_s  -0.07583677 -0.01662397 -0.1463551  0.6858192  1.00000000

$lower
            a         b         c   alpha_s    beta_s
[1] 2.1939211 0.8379626 3.3450328 5.6383868 1.1241316

$upper
            a         b         c   alpha_s    beta_s
[1]  6.725861  1.324311  8.907950 20.543389  3.265742

# estimate parameters in the basic model
> est = condSurv(data$data, p0 = c(5,1,7,5.53,0.54), type ="const")
> est
$par
            a         b         c         s         q
[1] 3.5104217 1.0581961 4.9842110 3.8440807 0.5026315

$std
             a          b          c          s          q
[1] 0.95373452 0.11583314 1.18229470 1.03224219 0.01656741

$logL
[1] -1378.151

$cor
                 a             b             c             s             q
  a   1.0000000000 -0.9815935010  0.9950038321  9.961205e-01 -1.688666e-04
  b  -0.9815935010  1.0000000000 -0.9675099177 -9.722442e-01  1.901939e-04
  c   0.9950038321 -0.9675099177  1.0000000000  9.925339e-01 -2.778626e-04
  s   0.9961205257 -0.9722441573  0.9925339245  1.000000e+00  2.017399e-05
  q  -0.0001688666  0.0001901939 -0.0002778626  2.017399e-05  1.000000e+00

$lower
            a         b         c         s         q
[1] 2.6511119 0.8510979 3.1310323 2.2710817 0.4711237

$upper
            a         b         c         s         q
[1] 5.9786254 1.3156556 7.9343785 6.5066185 0.5362594
```

E.1.4 Lognormal model

```
# simulate data from the lognormal model
> data = simRandomS(c=7, a=5, b=1, mu_s = 2, sigma_s = 0.25, type =
        "lognorm",  N= 1000)
# estimate back parameters in the lognormal model
> est = condSurv(data$data,p0 =c(5,1,7,2,0.25),upper = c(8,1.5,10,3,0.5),
  lower =c(2, 0.5, 4, 1, 0.1)type = "lognorm")
> est
$par
            a         b         c      mu_s   sigma_s
[1] 4.0680990 1.0850722 5.8791031 1.7815489 0.2433203

$std
             a          b          c       mu_s    sigma_s
[1] 1.08516998 0.11685029 1.36351485 0.23133096 0.04848929

$logL
[1] -1321.492

$cor
                  a          b          c       mu_s    sigma_s
      a   1.0000000 -0.9787560  0.9948124  0.9931416 -0.5205515
      b  -0.9787560  1.0000000 -0.9605370 -0.9581883  0.5697002
      c   0.9948124 -0.9605370  1.0000000  0.9989595 -0.4598813
   mu_s   0.9931416 -0.9581883  0.9989595  1.0000000 -0.4471489
sigma_s  -0.5205515  0.5697002 -0.4598813 -0.4471489  1.0000000

$lower
            a         b         c      mu_s   sigma_s
[1] 2.4117282 0.8786016 3.7315732 1.3812386 0.1646442

$upper
            a         b         c      mu_s   sigma_s
[1] 6.8620622 1.3400634 9.2625418 2.2978770 0.3595923

# estimate parameters in the basic model
> est = condSurv(data$data, c(5,1,7,5.9,0. 41), type ="const")
> est
$par
            a         b         c         s         q
[1] 3.8198416 1.0916439 5.5070490 4.5919598 0.4834896

$std
             a          b          c          s          q
[1] 0.99455512 0.11496758 1.24401946 1.12946991 0.01663226

$logL
[1] -1322.205
```

```
$cor
                a             b            c            s             q
  a   1.0000000000 -0.9804360788  0.994967310  0.995419276 -0.0002700728
  b  -0.9804360788  1.0000000000 -0.965549136 -0.969437219 -0.0001650704
  c   0.9949673105 -0.9655491358  1.000000000  0.991965660 -0.0018542618
  s   0.9954192758 -0.9694372190  0.991965660  1.000000000  0.0019528383
  q  -0.0002700728 -0.0001650704 -0.001854262  0.001952838  1.0000000000

$lower
           a         b         c         s         q
[1] 2.2930722 0.8880434 3.5369793 2.8354818 0.4519651

$upper
           a        b        c        s        q
[1] 6.363162 1.341924 8.574432 7.436512 0.517213
```

E.2 Simulation study - semi-competing risks data

E.2.1 Uniform model

```
# simulate data from the uniform model
> data = simSemiCens(c=5,a=5,b=1,A=10,type="unif",N=1000)
# estimate back parameters in the uniform model
> est = estSemi(x=data$x, z=data$z, x_z=data$x_z, z_o=data$z_o, tau_o=
        data$tau_o, tau=data$tau, p0=c(5,1,5,10), upper= c(7,1.2,7,12),
        lower=c(3,0.6,3,8), type="unif")
> est
$par
           a        b        c        A
[1] 4.359639 1.044099 4.147613 8.302907

$std
             a          b          c          A
[1] 0.42793742 0.03812465 0.45065301 0.94949841

$logL
[1] -1226.953

$cor
               a          b          c          A
   a   1.0000000 -0.8307446  0.9884136  0.9470393
   b  -0.8307446  1.0000000 -0.8007250 -0.7687389
   c   0.9884136 -0.8007250  1.0000000  0.9575824
   A   0.9470393 -0.7687389  0.9575824  1.0000000

$lower
           a         b         c         A
```

```
[1] 3.5966318 0.9719865 3.3520495 6.6357077

$upper
            a         b         c         A
[1]  5.284514  1.121563  5.131993 10.388985
```

E.2.2 Exponential model

```
# simulate data from the exponential model
> data = simSemiCens(c=7,a=5,b=1,lambda_s=1/10,type="expon",N=1000)
# estimate back parameters in the exponential model
> est = estSemi(x=data$x, z=data$z, x_z=data$x_z, z_o=data$z_o, tau_o=
        data$tau_o, tau = data$tau, p0 = c(5,1,7,1/10), upper =
        c(6,1.2,8,0.15), lower=c(4,0.6,6,0.05), type="expon")
> est
$par
             a          b          c   lambda_s
[1] 5.20652289 0.96767871 7.17269943 0.09751067

$std
             a          b          c   lambda_s
[1] 0.58286753 0.03952583 0.75465342 0.01185323

$logL
[1] -1554.162

$cor
                    a          b          c   lambda_s
       a   1.0000000 -0.8883496  0.9892632 -0.9188265
       b  -0.8883496  1.0000000 -0.8381623  0.7937611
       c   0.9892632 -0.8381623  1.0000000 -0.9246157
lambda_s  -0.9188265  0.7937611 -0.9246157  1.0000000

$lower
             a          b          c   lambda_s
[1] 4.18075254 0.89322804 5.83612312 0.07683863

$upper
            a         b         c  lambda_s
[1] 6.4839716 1.0483349 8.8153756 0.1237441
```

E.2.3 Gamma model

```
# simulate data from the gamma model
> data = simSemiCens(c=7,a=5,b=1,alpha_s=49/4, beta_s =7/4 ,type="gamma",
         N=1000)
# estimate back parameters in the gamma model
> est = estSemi(x=data$x, z=data$z, x_z=data$x_z, z_o=data$z_o, tau_o=
```

```
        data$tau_o, tau=data$tau, p0=c(5,1,7,49/4,7/4), upper=
        c(7,1.2,9,14,2), lower=c(3,0.6,5,10,1), type="gamma")
> est
$par
           a         b         c   alpha_s    beta_s
[1]  5.977629  0.963138  8.384926 13.054532  1.548342

$std
             a          b          c    alpha_s     beta_s
[1] 0.93699944 0.06519331 1.15889965 1.86004242 0.20261157

$logL
[1] -1292.934

$cor
                  a          b          c    alpha_s     beta_s
      a   1.0000000 -0.9522623  0.9939360  0.6262215 -0.3606337
      b  -0.9522623  1.0000000 -0.9239829 -0.7029166  0.2014730
      c   0.9939360 -0.9239829  1.0000000  0.5960365 -0.4004466
alpha_s   0.6262215 -0.7029166  0.5960365  1.0000000  0.4939645
 beta_s  -0.3606337  0.2014730 -0.4004466  0.4939645  1.0000000

$lower
           a         b         c   alpha_s    beta_s
[1] 4.3964265 0.8434726 6.3951477 9.8736505 1.1980611

$upper
           a         b         c   alpha_s    beta_s
[1]  8.127522  1.099781 10.993802 17.260162  2.001035
```

E.2.4 Lognormal model

```
# simulate data from the lognormal model
data = simSemiCens(c=7, a=5, b=1, mu_s = 2, sigma_s=0.25,type="lognorm",
       N = 1000)
# estimate back parameters in the lognormal model
> est = estSemi(x=data$x, z=data$z, x_z=data$x_z, z_o=data$z_o, tau_o=
        data$tau_o, tau=data$tau, p0=c(5,1,7,2,0.25), upper=
        c(7,1.2,9,4,0.4), lower=c(3,0.6,5,0.1,0.1), type="lognorm")
> est
$par
           a         b         c      mu_s   sigma_s
[1] 4.6975142 1.0108667 6.7393923 1.9440112 0.2563841

$std
             a          b          c       mu_s    sigma_s
[1] 0.58611658 0.05222002 0.75064433 0.11133174 0.01948182
```

```
$logL
[1] -1277.885

$cor
                  a          b          c       mu_s    sigma_s
      a   1.0000000 -0.9233959  0.9887799  0.9822202 -0.3662678
      b  -0.9233959  1.0000000 -0.8757601 -0.8659999  0.4780673
      c   0.9887799 -0.8757601  1.0000000  0.9944276 -0.3265499
   mu_s   0.9822202 -0.8659999  0.9944276  1.0000000 -0.2772342
sigma_s  -0.3662678  0.4780673 -0.3265499 -0.2772342  1.0000000

$lower
            a         b         c      mu_s   sigma_s
[1] 3.6784117 0.9135265 5.4176482 1.7376021 0.2209071

$upper
            a         b         c      mu_s   sigma_s
[1] 5.9989586 1.1185789 8.3836024 2.1749395 0.2975585
```

E.3 Data analysis - VHF data

For this dataset we obtained better results using the `parscale` option in the `optim`() function, which is not built in in the general `condSurv`() function. Therefore the call to `optim`() including `parscale` is shown instead of the call to `condSurv`(). To plot the conditional sub-survival curves for the VHF-data, run `condSurv`() with these calls to `optim`() instead of the ones given in section D.2.1.

E.3.1 Uniform model

```
> opt = optim(par = c(0.01, 0.98, 0.36, 1.2), fn=lklhUnif, x = vhf_x, z =
        vhf_z, tau = vhf_tau, method="L-BFGS-B", lower=c(0.001,0.6,0.3,1),
        upper =c(0.05,1,1,3),control =list(parscale = c(0.01,0.1,1,1)),
        hessian=T)
> p = opt$par
> hessian = opt$hessian
> std = sqrt(solve(hessian))
# make confidence intervals
> lower = p*exp(-1.96*diag(std)/p)
> upper = p*exp(1.96*diag(std)/p)
> list(par = p, std = diag(std), logL = -opt$value, cor =
  cov2cor(solve(hessian)),lower=lower, upper = upper)
$par
              a           b           c           A
[1] 0.002722714 0.981088504 0.355242005 1.150123081

$std
               a            b            c            A
[1] 0.0003483071 0.0231150675 0.0822655982 0.2855749231
```

```
$logL
[1] -2389.049

$cor
              a          b          c          A
   a  1.0000000 -0.5964609  0.3723681  0.3516705
   b -0.5964609  1.0000000  0.4599028  0.4371730
   c  0.3723681  0.4599028  1.0000000  0.9453478
   A  0.3516705  0.4371730  0.9453478  1.0000000

$lower
              a           b           c           A
[1] 0.002118892 0.936813132 0.225633023 0.706948153

$upper
              a           b           c           A
[1] 0.003498607 1.027456405 0.559301475 1.871117557
```

E.3.2 Exponential model

```
> opt = optim(par = c(0.005, 0.98, 0.37, 0.99), fn=lklhExp, x = vhf_x, z=
        vhf_z,tau=vhf_tau,method ="L-BFGS-B",lower=c(0.001,0.6,0.25,0.4),
        upper =c(0.05,1.1,1,1.5),control=list(parscale =
        c(0.01,0.1,0.1,0.1)),hessian=T)
> p = opt$par
> hessian = opt$hessian
> std = sqrt(solve(hessian))
# make confidence intervals
> lower = p*exp(-1.96*diag(std)/p)
> upper = p*exp(1.96*diag(std)/p)
> list(par = p, std = diag(std), logL = -opt$value, cor =
  cov2cor(solve(hessian)),lower=lower, upper = upper)
$par
              a           b           c    lambda_s
[1] 0.002707739 0.983180717 0.365285487 0.998944840

$std
               a            b            c     lambda_s
[1] 0.0003430045 0.0229936331 0.0822246500 0.2473211100

$logL
[1] -2389.753

$cor
                    a          b          c   lambda_s
       a   1.0000000 -0.6071946  0.3603825 -0.3329657
       b  -0.6071946  1.0000000  0.4575940 -0.4235559
       c   0.3603825  0.4575940  1.0000000 -0.9193817
```

```
lambda_S  -0.3329657 -0.4235559 -0.9193817  1.0000000

$lower
             a          b          c   lambda_s
[1] 0.00211241 0.93913051 0.23497736 0.61488678

$upper
              a           b           c    lambda_s
[1] 0.003470845 1.029297117 0.567856776 1.622885417
```

E.3.3 Gamma model

```
> opt = optim(par = c(4.65,0.24,16.7,122,6.97), fn=lklhGam, x = vhf_x, z=
        vhf_z, tau = vhf_tau ,method = "L-BFGS-B", lower =
        c(4, 0.15, 15, 116, 5.5),upper = c(5.8,0.3,19,125, 8.0),control =
        list(parscale = c(1,0.1,1,10,1)),hessian=T)
> p = opt$par
> hessian = opt$hessian
> std = sqrt(solve(hessian))
# make confidence intervals
> lower = p*exp(-1.96*diag(std)/p)
> upper = p*exp(1.96*diag(std)/p)
> list(par = p, std = diag(std), logL = -opt$value, cor =
  cov2cor(solve(hessian)),lower=lower, upper = upper)
$par
              a           b           c     alpha_s      beta_s
[1]   4.6497239   0.2397269  16.7150800 121.9852362   6.9725385

$std
               a            b            c      alpha_s       beta_s
[1] 2.441352e-02 3.276978e-03 8.129194e-02 1.220926e+02 7.159346e+00

$logL
[1] -2377.063

$cor
                   a          b            c     alpha_s       beta_s
       a   1.0000000 -0.5482942  0.172615257 -0.17335662 -0.174087725
       b  -0.5482942  1.0000000  0.271445958  0.45925493  0.457740950
       c   0.1726153  0.2714460  1.000000000  0.01069039  0.005923669
 alpha_s  -0.1733566  0.4592549  0.010690392  1.00000000  0.999968524
  beta_s  -0.1740877  0.4577410  0.005923669  0.99996852  1.000000000

$lower
             a          b          c    alpha_s     beta_s
[1]  4.6021188  0.2333893 16.5565048 17.1530264  0.9318971
```

```
$upper
              a           b           c     alpha_s      beta_s
[1]   4.6978214   0.2462366  16.8751741 867.5085978  52.1691632
```

E.3.4 Lognormal model

```
> opt = optim(par = c(4.69,0.24,16.7,2.86,0.09),fn=lklhLognorm, x = vhf_x,
        z=vhf_z, tau = vhf_tau,method = "L-BFGS-B", lower =
        c(4,0.15,15,2,0.05),upper = c(6,0.3,20,3.5, 0.15),control =
        list(parscale = c(1,0.1,1,1,0.1)),hessian=T)
> p = opt$par
> hessian = opt$hessian
> std = sqrt(solve(hessian))
# make confidence intervals
> lower = p*exp(-1.96*diag(std)/p)
> upper = p*exp(1.96*diag(std)/p)
> list(par = p, std = diag(std), logL = -opt$value, cor =
  cov2cor(solve(hessian)),lower=lower, upper = upper)
$par
              a           b           c        mu_s     sigma_s
[1]  4.68309217  0.23851057 16.72049151  2.85920420  0.09036403

$std
              a           b           c        mu_s     sigma_s
[1] 0.018456329 0.002782282 0.010374313 0.024293375 0.046665438

$logL
[1] -2377.047

$cor
                   a           b           c        mu_s     sigma_s
      a   1.00000000 -0.33659890 -0.04916612  0.02344246  0.02651393
      b  -0.33659890  1.00000000  0.02887839 -0.44233645 -0.47016611
      c  -0.04916612  0.02887839  1.00000000  0.04145266  0.01676736
   mu_s   0.02344246 -0.44233645  0.04145266  1.00000000  0.96433663
sigma_s   0.02651393 -0.47016611  0.01676736  0.96433663  1.00000000

$lower
              a           b           c        mu_s     sigma_s
[1]  4.64705712  0.23311916 16.70017022  2.81198347  0.03284077

$upper
             a          b          c       mu_s    sigma_s
[1]  4.7194067  0.2440267 16.7408375  2.9072179  0.2486439
```

E.4 Data analysis - Carcinoma

E.4.1 Uniform model

```
> est = estSemi(x = car_x, z = car_z, x_z = car_xz, p0 =
        c(0.17,0.69,0.80,1.75), upper= c(0.2,0.9,1,2), lower=
        c(0.01,0.2,0.1,0.2), type ="unif")
> est
$par
            a         b         c         A
[1] 0.1675556 0.6877733 0.8027270 1.7487681

$std
            a         b         c         A
[1] 0.1370179 0.1044886 0.6615942 1.4616310

$logL
[1] -382.4325

$cor
               a          b          c          A
   a   1.0000000 -0.9347591  0.9549628  0.9416651
   b  -0.9347591  1.0000000 -0.8039725 -0.7927773
   c   0.9549628 -0.8039725  1.0000000  0.9860752
   A   0.9416651 -0.7927773  0.9860752  1.0000000

$lower
             a          b          c          A
[1] 0.03373489 0.51065295 0.15959118 0.33984409

$upper
            a         b         c         A
[1] 0.8322207 0.9263279 4.0376331 8.9988025
```

E.4.2 Exponential model

```
> est = estSemi(x = car_x, z = car_z, x_z = car_xz, p0 =
        c(0.14,0.71,0.68,0.88), upper= c(0.2,0.9,1,1), lower=
        c(0.01,0.2,0.1,0.05), type ="expon")
> est
$par
            a         b         c  lambda_s
[1] 0.1383447 0.7143961 0.6806455 0.8824024

$std
             a          b          c   lambda_s
[1] 0.09619097 0.09202761 0.50281925 0.67967161

$logL
```

```
[1] -382.9794

$cor
                  a          b          c   lambda_s
       a   1.0000000 -0.9081362  0.9336541 -0.9072017
       b  -0.9081362  1.0000000 -0.7212055  0.7040949
       c   0.9336541 -0.7212055  1.0000000 -0.9686767
lambda_s  -0.9072017  0.7040949 -0.9686767  1.0000000

$lower
             a          b          c   lambda_s
[1] 0.03540882 0.55499157 0.15999036 0.19499141

$upper
            a         b         c  lambda_s
[1] 0.5405219 0.9195849 2.8956639 3.9931704
```

E.4.3 Gamma model

```
> est = estSemi(x = car_x, z = car_z, x_z = car_xz, p0 =
        c(2.18,0.36,5.66,5.93,0.95), upper= c(3,0.9,6,6,2), lower=
        c(1,0.2,1,1,0.1), type ="gamma")
> est
$par
            a         b         c   alpha_s    beta_s
[1] 2.1788318 0.3619981 5.6606065 5.9300986 0.9499155

$std
            a         b         c   alpha_s    beta_s
[1] 3.5832024 0.1850263 6.7956321 4.5707542 0.7050852

$logL
[1] -380.814

$cor
                  a          b          c    alpha_s     beta_s
      a   1.0000000 -0.9924300  0.9967761  0.8189348 -0.6963824
      b  -0.9924300  1.0000000 -0.9810184 -0.8375437  0.6499930
      c   0.9967761 -0.9810184  1.0000000  0.8028775 -0.7196085
alpha_s   0.8189348 -0.8375437  0.8028775  1.0000000 -0.1673029
 beta_s  -0.6963824  0.6499930 -0.7196085 -0.1673029  1.0000000

$lower
             a          b          c    alpha_s     beta_s
[1] 0.08676663 0.13293150 0.53822717 1.30908397 0.22174800

$upper
             a          b          c    alpha_s     beta_s
[1] 54.7135263  0.9857905 59.5333491 26.8631118  4.0692113
```

E.4.4 Lognormal model

```
> est = estSemi(x = car_x, z = car_z, x_z = car_xz, p0 =
        c(1.84,0.38,5,1.66,0.48), upper=c(2,0.9,6,3,0.5), lower=
        c(0.1,0.2,0.1,0.1,0.1), type ="lognorm")
> est
$par
            a         b         c      mu_s   sigma_s
[1] 1.8396373 0.3809158 5.0004989 1.6561903 0.4785995

$std
            a         b         c      mu_s   sigma_s
[1] 3.0683398 0.1923464 6.0726554 1.2040146 0.2054011

$logL
[1] -380.7409

$cor
                  a          b          c       mu_s    sigma_s
      a   1.0000000 -0.9924649  0.9964698  0.9938461 -0.7964774
      b  -0.9924649  1.0000000 -0.9804023 -0.9772638  0.8162346
      c   0.9964698 -0.9804023  1.0000000  0.9977106 -0.7791034
   mu_s   0.9938461 -0.9772638  0.9977106  1.0000000 -0.7593330
sigma_s  -0.7964774  0.8162346 -0.7791034 -0.7593330  1.0000000

$lower
             a          b          c       mu_s    sigma_s
[1] 0.06998146 0.14157945 0.46268644 0.39837651 0.20637374

$upper
            a         b         c      mu_s   sigma_s
[1] 48.359459  1.024844 54.043057  6.885362  1.109916
```

E.5 Data analysis - Bone marrow transplant

For this dataset we obtained better results using the `parscale` option in the `optim`() function, which is not built in in the general `estSemi`() function. Therefore the call to `optim`() including `parscale` is shown instead of the call to `estSemi`().

E.5.1 Uniform model

```
> opt = optim(par = c(0.01,0.56,0.28,0.66), fn=lklhUnifSemi, x = x,
        z=z, xz = x_z, z_o = z_o, tau_o = tau_o, tau = tau,method =
        "L-BFGS-B",lower=c(0.01,0.1,0.1,0.1), upper= c(0.1,0.6,0.3,0.8),
        control =list(parscale = c(0.01,0.1,0.1,0.11)),hessian=T)
> p = opt$par
> hessian = opt$hessian
> std = sqrt(solve(hessian))
# make confidence intervals
> lower = p*exp(-1.96*diag(std)/p)
```

```
> upper = p*exp(1.96*diag(std)/p)
> list(par = p, std = diag(std), logL = -opt$value, cor =
  cov2cor(solve(hessian)), lower=lower, upper = upper)
$par
             a          b          c          A
[1] 0.01402081 0.56418105 0.27672411 0.65761364

$std
              a           b           c           A
[1] 0.006393862 0.046571754 0.130954706 0.335198531

$logL
[1] -1000.186

$cor
              a          b          c          A
   a  1.0000000 -0.8659325  0.7135718  0.6934817
   b -0.8659325  1.0000000 -0.3091098 -0.2965781
   c  0.7135718 -0.3091098  1.0000000  0.9704298
   A  0.6934817 -0.2965781  0.9704298  1.0000000

$lower
              a           b           c           A
[1] 0.005735822 0.479902062 0.109452023 0.242152394

$upper
             a          b          c          A
[1] 0.03427289 0.66326087 0.69963289 1.78588239
```

E.5.2 Exponential model

```
> opt = optim(par = c(0.01,0.57,0.27,1.82), fn=lklhExpSemi, x = x,
        z=z, xz = x_z, z_o = z_o, tau_o = tau_o, tau = tau,method =
        "L-BFGS-B",lower=c(0.01,0.1,0.1,0.1), upper= c(0.1,0.6,0.5,2),
        control =list(parscale = c(0.01,0.1,0.1,1)),hessian=T)
> p = opt$par
> hessian = opt$hessian
> std = sqrt(solve(hessian))
# make confidence intervals
> lower = p*exp(-1.96*diag(std)/p)
> upper = p*exp(1.96*diag(std)/p)
> list(par = p, std = diag(std), logL = -opt$value, cor =
  cov2cor(solve(hessian)), lower=lower, upper = upper)
$par
             a          b          c   lambda_s
[1] 0.01336946 0.56948652 0.27267194 1.82434961

$std
```

```
             a           b           c    lambda_s
[1] 0.005731145 0.045864928 0.117917819 0.886649699

$logL
[1] -1002.776

$cor
                   a          b          c   lambda_s
       a   1.0000000 -0.8684087  0.6667413 -0.6359217
       b  -0.8684087  1.0000000 -0.2561493  0.2421727
       c   0.6667413 -0.2561493  1.0000000 -0.9452784
lambda_s  -0.6359217  0.2421727 -0.9452784  1.0000000

$lower
             a           b           c    lambda_s
[1] 0.00577057  0.48632734  0.11682316  0.70373500

$upper
             a          b          c   lambda_s
[1] 0.03097483 0.66686545 0.63643189 4.72941023
```

E.5.3 Gamma model

```
> opt = optim(par = c(4.9,0.14,12.1,577,47.6), fn=lklhGamSemi, x = x,
        z=z, xz = x_z, z_o = z_o, tau_o = tau_o, tau = tau, method =
        "L-BFGS-B",lower=c(4.3,0.1,5,570,46.2),upper=c(5,0.2,18,590,50.1),
        control =list(parscale = c(1,0.1,1,100,10)),hessian=T)
> p = opt$par
> hessian = opt$hessian
> std = sqrt(solve(hessian))
# make confidence intervals
> lower = p*exp(-1.96*diag(std)/p)
> upper = p*exp(1.96*diag(std)/p)
> list(par = p, std = diag(std), logL = -opt$value, cor =
  cov2cor(solve(hessian)), lower=lower, upper = upper)
$par
             a           b           c     alpha_s      beta_s
[1]  4.8864181   0.1385993  12.0844878 576.9045891  47.6163380

$std
             a          b          c    alpha_s     beta_s
[1]  0.4992233  0.0124735  0.3303465 17.6879044  1.2182979

$logL
[1] -955.5955

$cor
                  a          b          c    alpha_s     beta_s
```

```
      a   1.0000000 -0.9348778  0.6419044  0.6337955 -0.6231028
      b  -0.9348778  1.0000000 -0.4538495 -0.4501035  0.4400281
      c   0.6419044 -0.4538495  1.0000000  0.9777696 -0.9676827
alpha_s   0.6337955 -0.4501035  0.9777696  1.0000000 -0.9748611
 beta_s  -0.6231028  0.4400281 -0.9676827 -0.9748611  1.0000000

$lower
              a           b           c     alpha_s      beta_s
[1]   3.9996832   0.1161861  11.4540488 543.2574121  45.2873590

$upper
              a           b           c     alpha_s      beta_s
[1]   5.9697432   0.1653361  12.7496266 612.6357367  50.0650888
```

E.5.4 Lognormal model

```
> opt = optim(par = c(4.9,0.14,12.1,2.5,0.04), fn=lklhLnormSemi,x =x,
        z=z, xz = x_z, z_o = z_o, tau_o = tau_o, tau =tau, method =
        "L-BFGS-B",lower=c(4.2,0.1,11.5,2.3,0.03), upper=
        c(5.1,0.2,12.7,2.7,0.07), control = list(parscale =
        c(1,0.1,1,1,0.01)), hessian=T)
> p = opt$par
> hessian = opt$hessian
> std = sqrt(solve(hessian))
# make confidence intervals
> lower = p*exp(-1.96*diag(std)/p)
> upper = p*exp(1.96*diag(std)/p)
> list(par = p, std = diag(std), logL = -opt$value, cor =
  cov2cor(solve(hessian)),lower=lower, upper = upper)
$par
              a           b           c        mu_s     sigma_s
[1]  4.92479405  0.13744123 12.08933865  2.49402800  0.04010462

$std
             a          b          c       mu_s    sigma_s
[1] 0.42730602 0.01259533 0.05700354 0.00731558 0.01204682

$logL
[1] -955.6274

$cor
                   a           b            c        mu_s      sigma_s
      a   1.00000000 -0.95128927  0.074676287 -0.01817957 -0.447598489
      b  -0.95128927  1.00000000 -0.023523498  0.06277815  0.490623767
      c   0.07467629 -0.02352350  1.000000000  0.63769614 -0.007057255
   mu_s  -0.01817957  0.06277815  0.637696138  1.00000000  0.282806130
sigma_s  -0.44759849  0.49062377 -0.007057255  0.28280613  1.000000000
```

```
$lower
            a          b          c       mu_s    sigma_s
[1]  4.1546183  0.1148445 11.9781264  2.4797306  0.0222588

$upper
            a          b          c       mu_s    sigma_s
[1]  5.8377436  0.1644841 12.2015835  2.5084078  0.0722582
```

E.5.5 Normal model

```
>  opt = optim(par = c(4.92,0.138,12.14,12.16,0.500), fn=lklhNormSemi, x
        = x, z=z, xz = x_z, z_o = z_o, tau_o = tau_o, tau = tau, method
        = "L-BFGS-B", lower = c(4.58,0.11,10.1,10.2,0.41), upper =
        c(5.65,0.17,14.2,14.5,0.59),control =list(parscale =
        c(1,0.1,1,1,0.01)),hessian=T)
> p = opt$par
> hessian = opt$hessian
> std = sqrt(solve(hessian))
# make confidence intervals
> lower = p*exp(-1.96*diag(std)/p)
> upper = p*exp(1.96*diag(std)/p)
> list(par = p, std = diag(std), logL = -opt$value, cor =
  cov2cor(solve(hessian)), lower=lower, upper = upper)
$par
            a          b          c       mu_s    sigma_s
[1]  4.9217850  0.1382357 12.1376051 12.1620335  0.5000000

$std
            a          b          c       mu_s    sigma_s
[1] 0.63725305 0.00614991 1.34752248 1.34533469 0.07989731

$logL
[1] -955.5186

$cor
                a          b          c       mu_s    sigma_s
      a  1.0000000 -0.5751481  0.9319768  0.9436757  0.5253518
      b -0.5751481  1.0000000 -0.3105646 -0.3496774 -0.2262832
      c  0.9319768 -0.3105646  1.0000000  0.9996007  0.5392284
   mu_s  0.9436757 -0.3496774  0.9996007  1.0000000  0.5052717
sigma_s  0.5253518 -0.2262832  0.5392284  0.5052717  1.0000000

$lower
           a         b         c      mu_s   sigma_s
[1] 3.8186551 0.1266925 9.7640609 9.7914410 0.3655527

$upper
            a          b          c       mu_s    sigma_s
[1]  6.3435861  0.1508307 15.0881338 15.1065669  0.6838958
```

E.6 Bootstrapping results

Here we present the bootstrapping results obtained in the data analysis for the different datasets. The estimates were found by running the bootstrapping scripts and `bca()` function given in appendix D.2.5. In each case the parameter estimates of the bootstrap samples are stored in the table `bootestimates`, where column 1 is for the parameter α, column 2 is for β, column 3 is for c and columns 4 and 5 are for the parameters in the distribution of S (α_S and β_S for the gamma model, μ_S and σ_S for the lognormal model.

E.6.1 VHF-data

E.6.1.1 Gamma model

```
# means
> mean(bootestimates[,1])
[1] 4.670817
> mean(bootestimates[,2])
[1] 0.2391655
> mean(bootestimates[,3])
[1] 16.71901
> mean(bootestimates[,4])
[1] 121.9531
> mean(bootestimates[,5])
[1] 6.970546
# biases
> mean(bootestimates[,1])-4.6497239
[1] 0.02109318
> mean(bootestimates[,2])-0.2397269
[1] -0.0005614308
> mean(bootestimates[,3])-16.7150800
[1] 0.003931557
> mean(bootestimates[,4])-121.9852362
[1] -0.03212197
> mean(bootestimates[,5])- 6.9725385
[1] -0.001992295
# standard deviatons
> sd(bootestimates[,1])
[1] 0.1566409
> sd(bootestimates[,2])
[1] 0.006375413
> sd(bootestimates[,3])
[1] 0.09760275
> sd(bootestimates[,4])
[1] 0.2024508
> sd(bootestimates[,5])
[1] 0.041866
# percentile intervals
```

```
> quantile(bootestimates[,1], c(0.025,0.975))
    2.5%    97.5%
4.301195 5.080174
> quantile(bootestimates[,2], c(0.025,0.975))
     2.5%     97.5%
0.2237287 0.2544450
> quantile(bootestimates[,3], c(0.025,0.975))
    2.5%    97.5%
16.46989 16.89710
> quantile(bootestimates[,4], c(0.025,0.975))
    2.5%    97.5%
121.4852 122.2910
> quantile(bootestimates[,5], c(0.025,0.975))
    2.5%    97.5%
6.891271 7.050023
# BCa intervals
$BCa1
2.854359% 97.81792%
 4.330626  5.106896

$BCa2
2.342231% 97.33243%
0.2231252 0.2541578

$BCa3
0.2144287%  86.28735%
  16.20541   16.78309

$BCa4
4.130605% 98.54297%
 121.5666  122.3566

$BCa5
1.747657% 96.44248%
 6.880526  7.038442
```

E.6.2 Carcinoma data

E.6.2.1 Gamma model

```
# means
> mean(bootestimates[,1])
[1] 3.083362
> mean(bootestimates[,2])
[1] 0.3901593
> mean(bootestimates[,3])
[1] 7.105571
> mean(bootestimates[,4])
[1] 6.689408
```

```
> mean(bootestimates[,5])
[1] 1.126241
# biases
> mean(bootestimates[,1])-2.1788318
[1] 0.9045303
> mean(bootestimates[,2])-0.3619981
[1] 0.02816118
> mean(bootestimates[,3])-5.6606065
[1] 1.444965
> mean(bootestimates[,4])-5.9300986
[1] 0.7593093
> mean(bootestimates[,5])-.9499155
[1] 0.1763256
# standard deviations
> sd(bootestimates[,1])
[1] 2.454286
> sd(bootestimates[,2])
[1] 0.1370978
> sd(bootestimates[,3])
[1] 4.43352
> sd(bootestimates[,4])
[1] 3.174334
> sd(bootestimates[,5])
[1] 0.6313188
# percentile intervals
> quantile(bootestimates[,1], c(0.025,0.975))
     2.5%     97.5%
0.1684627 8.1358613
> quantile(bootestimates[,2], c(0.025,0.975))
     2.5%     97.5%
0.2230286 0.6873895
> quantile(bootestimates[,3], c(0.025,0.975))
      2.5%      97.5%
 0.8508462 14.9927268
> quantile(bootestimates[,4], c(0.025,0.975))
     2.5%     97.5%
 1.721951 14.115596
> quantile(bootestimates[,5], c(0.025,0.975))
     2.5%     97.5%
0.3515232 2.8402181
# BCa intervals
$BCa1
0.3202994%  92.34479%
  0.112385   7.411666

$BCa2
3.977196% 98.51485%
0.2290606 0.7246130
```

```
$BCa3
0.1455884%  89.33868%
 0.5248217 14.1988195

$BCa4
0.8019389%  94.45508%
  1.415969  12.719724

$BCa5
2.571154% 97.57069%
0.3441586 2.8628440
```

E.6.2.2 Lognormal model

```
# means
> mean(bootestimates[,1])
[1] 3.125752
> mean(bootestimates[,2])
[1] 0.3941494
> mean(bootestimates[,3])
[1] 7.105593
> mean(bootestimates[,4])
[1] 1.728427
> mean(bootestimates[,5])
[1] 0.4936811
# biases
> mean(bootestimates[,1])-1.8396373
[1] 1.286115
> mean(bootestimates[,2])-0.3809158
[1] 0.01323355
> mean(bootestimates[,3])-5.0004989
[1] 2.105094
> mean(bootestimates[,4])-1.6561903
[1] 0.0722367
> mean(bootestimates[,5])-0.4785995
[1] 0.01508161
# standard deviations
> sd(bootestimates[,1])
[1] 2.642797
> sd(bootestimates[,2])
[1] 0.1393561
> sd(bootestimates[,3])
[1] 4.793424
> sd(bootestimates[,4])
[1] 0.8365234
> sd(bootestimates[,5])
[1] 0.1478255
```

```
# percentile intervals
> quantile(bootestimates[,1], c(0.025,0.975))
     2.5%     97.5%
0.1997137 7.9787774
> quantile(bootestimates[,2], c(0.025,0.975))
     2.5%     97.5%
0.2253478 0.6768152
> quantile(bootestimates[,3], c(0.025,0.975))
     2.5%     97.5%
 1.045873 15.398952
> quantile(bootestimates[,4], c(0.025,0.975))
     2.5%     97.5%
0.1154208 2.8294860
> quantile(bootestimates[,5], c(0.025,0.975))
     2.5%     97.5%
0.2817967 0.8636517
# BCa intervals
$BCa1
0.6823396%  94.08371%
 0.1500916  7.8040416

$BCa2
5.101903% 98.89132%
0.2301596 0.7066866

$BCa3
0.4625833%  92.62239%
 0.8575269 14.7481291

$BCa4
0.5641228%  92.40937%
0.02915044 2.76575410

$BCa5
5.088519% 98.91761%
0.3051378 0.9842683
```

Appendix F

Basic model results

In this appendix some of the results from the data analysis in the Master's project [35] is repeated. We begin by showing how to generate first passage times in the gamma process (copied from the project thesis).

F.1 Simulation of first passage times

In chapter 5 we found the cumulative distribution function of the first passage time in the gamma process. In order to simulate the gamma process models, we first need a method of sampling from this first passage time distribution. One way to do this is by using the probability integral transform, also called the inverse transformation method.

F.1.1 The probability integral transform

The probability integral transform is a general method for simulating random variables with continuous distribution functions [34]. The main idea is that if T is a continuous random variable with cumulative distribution function $F(t)$, then $u = F(t) \sim$ Unif[0,1]. Assuming that $F(t)$ has an inverse, we then have that $t = F^{-1}(u) \sim F(t)$. Thus, the approach is as follows:

1. Generate a random number u from the standard uniform distribution $Unif[0, 1]$
2. Calculate the value of t such that $F(t) = u \Rightarrow F^{-1}(u) = t$
3. Then t is a random variable with the same distribution as $F(t)$.

However, since our target distribution $F_T(t)$ (see equation (5.2)) is difficult to invert, an approximate technique built on linear interpolation will be used. This method is for instance described in chapter 6 in [14]. By using this technique, we have to search across a predetermined set of values for t. This means that we have to make sure that the chosen grid of t-values span all of the support of $f(t)$. For each chosen grid point we can calculate $u_i = F(t_i)$. Then, we can draw $U \sim$ Unif$[0, 1]$ and linearly interpolate between the two nearest grid points so that $u_i \leq u_j$:

$$T = \frac{u_j - U}{u_j - u_i} t_i + \frac{U - u_i}{u_j - u_i} t_j$$

The degree of accuracy of this approach can be improved by increasing the number of grid points. In summary, a procedure to generate N samples from our target distribution with critical level d and shape function $v(t) = \alpha \cdot t^{\beta}$ is shown in algorithm 3. This algorithm is implemented in the function `simdata()` which is included in

Algorithm 3 Algortihm to sample from $F_T(t; v(t), d)$

```
Define a grid t of possible values for t
for n=1 to N do
    U ~ Unif[0,1]
    for i=1 to length(t) do
        u[i] = F_T(t[i]; v(t[i]), d)
        if u[i] > U then
            T[n] = (u[i]-U)/(u[i]-u[i-1]) t[i-1] + (U-u[i-1])/(u[i]-u[i-1]) t[i]
            break
        end if
    end for
end for
```

appendix D.1.1.

F.2 Log-likelihood function for the basic model

$$
\begin{aligned}
l &= \ln \mathrm{L} \\
&= m \cdot \ln(1-q) + \sum_{i=1}^{m} \ln \left\{ v'(x_i) \left[\Psi(x_i) - \log(c) \right] \left(1 - \frac{\Gamma(v(x_i), c)}{\Gamma(v(x_i))} \right) \right. \\
&\left. + \frac{v'(x_i)}{v(x_i)^2 \Gamma(v(x_i))} c^{v(x_i)} {}_2F_2(v(x_i), v(x_i); v(x_i)+1, v(x_i)+1; -c) \right\} \\
&+ n \cdot \ln(q) + \sum_{j=1}^{n} \ln \left\{ v'(z_j) \left[\Psi(z_j) - \log(s) \right] \left(1 - \frac{\Gamma(v(z_j), s)}{\Gamma(v(z_j))} \right) \right. \\
&\left. + \frac{v'(z_j)}{v(z_j)^2 \Gamma(v(z_j))} s^{v(z_j)} {}_2F_2(v(z_j), v(z_j); v(z_j)+1, v(z_j)+1; -s) \right\} \\
&+ \sum_{k=1}^{r} \ln \left[(1-q) \left(1 - \frac{\Gamma(v(\tau_k), c)}{\Gamma(v(\tau_k))} \right) + q \left(1 - \frac{\Gamma(v(\tau_k), s)}{\Gamma(v(\tau_k))} \right) \right]
\end{aligned}
\tag{F.1}
$$

F.3 Fit of basic model to VHF data

Using the `condSurv()` function on the log-likelihood function in (F.1) for the VHF-data produced the results in table F.1.

TABLE F.1: Maximum likelihood estimates of the parameters α, β, c, s and q in the basic model for the VHF data. In addition, standard deviations from the Hessian matrix and 95% confidence intervals are included.

Parameter	Estimate	Standard deviation	Lower bound	Upper bound
α	3.8029	0.3138	3.2350	4.4704
β	0.2535	0.0120	0.2310	0.2783
c	14.7400	0.0770	14.5898	14.8917
s	13.7092	1.3046	11.3765	16.5202
q	0.3159	0.0294	0.2632	0.3791