Universiteit Leiden
The Netherlands

Leiden University Medical Center

Department of Mathematics

Master Thesis
Statistical Science for the Life and Behavioural Sciences

Inverse Probability Censoring Weights for Routine Outcome Monitoring Data

Author:
Sanne J.W. Willems

Thesis Advisor:
Dr. M. Fiocco (LUMC)

Supervisor:
Prof. Dr. A.W. van der Vaart (LU)

June 2014

Abstract

Many researchers who work in survival studies encounter censored data during their career. Unfortunately, censored data makes analysis more complicated, since exact event times are not observed.

One type of censoring is interval censoring, occurring in longitudinal studies where patients are observed at repeated visits. If a patient experiences an event, it is detected at the next visit. In this case analysis is more difficult because no precise event times are observed. Packages are developed for `R` to handle interval censored data. However, the results produced by these packages are not satisfactory. The parameters of the Cox proportional hazards model can be estimated, but no information about the precision of these estimates is returned. This prevents drawing conclusions about the significance of the covariates. In this thesis, a bootstrapping method is proposed to assess the precision of the parameter estimates, such that better conclusions can be drawn from them.

Standard survival methodology assumes that patients' withdrawal from a study is independent of patients' characteristics. However, clinical experience may suggest dependent censoring. The Inverse Probability Censoring Weighted (IPCW) Estimator was developed to take a censoring mechanism into account when performing survival analysis. Application of this method is complicated, because it involves many mathematical formulas and programs must be written by the researcher. In this thesis a special way to prepare the data is proposed such that standard survival packages in `R` can be used to perform the IPCW method. In this way, IPCW becomes more available for researchers with limited mathematical and programming skills.

Furthermore, a method to generate data including dependent censoring is proposed. This method is used to perform a simulation study to assess the performance of the IPCW method compared to standard survival analysis in the presence of dependent censoring.

Acknowledgements

I would like to express my sincere gratitude to my supervisor Dr. M. Fiocco for her support and advice during the writing of this thesis. This work would not have been possible without her guidance and late-night e-mails.

The department of psychiatry at the Leiden University Medical Center (LUMC) is gratefully acknowledged for providing the dataset and research questions for this thesis.

I want to thank Dr. M.A. Jonker (VUMC) for introducing me to the master track *Statistical Science for the Life and Behavioural Sciences* at Leiden University.

I am grateful to my professors from the *Statistical Science for the Life and Behavioural Sciences* master track for the two years of inspiration that they offered me.

I want to thank my fellow students for letting me join their close group of friends when I entered the master track in the second semester, and for their help and patience when I had to do some catching up.

I wish to thank my family and friends for their support and encouragement throughout my study.

Contents

Chapter 1

Introduction

In many fields, there is an interest in how long it takes before a certain event occurs. In medicine one may want to estimate the remaining life time of a patient with a certain disease. In the industrial field the life time of products may be important. These kind of problems can be assessed by using *survival analysis techniques*. The name *survival* suggests that only a survival time (i.e. time till death) can be estimated. However, the techniques of analysis can be used to estimate any kind of time span, e.g. time to recovery, time to infection, or length of pregnancy. In these cases the observations are often referred to as *time to event* data.

In an ideal situation, the time to the event of interest is known precisely for each of the subjects in a study. These event times can be used to estimate survival probabilities over time. However, in many studies the exact event times are unknown. A subject may be lost to follow-up before experiencing the event, or may experience the event between two consecutive visits. Survival techniques have been extended to deal with several types of situations in which the exact event times are unknown.

1.1 Motivation for this Thesis

In a large-scale study started in 2002, information was gathered on patients suffering from mood, anxiety and somatoform disorders. At each visit, subjects were given questionnaires to determine the severity of their disorder and to measure patients' characteristics. The main goal of this procedure was to make a precise diagnosis for each patient and to inform clinicians and patients about treatment progress. The dataset is referred to as the *Routine Outcome Monitoring* data.

Schat et al. [24] focused on the group of patients with moderate to severe anxiety disorders. The aim of their study was to get insight in patient characteristics that lead to a slow recovery from these disorders. Two scales were used to assess the severity of the anxiety disorder, BAS and BSI-12. The former is observer-rated and the latter is self-reported. For this study, 50% improvement on both the BAS and BSI-12 scales was defined as the event of interest. While analyzing the dataset, two main difficulties were encountered.

The dataset contains only interval and right censored observations. Patients either experience the event of interest in between two consecutive visits, or are lost to follow-up. Since the researchers are familiar with right censoring techniques, but not with interval censoring techniques, it was decided to solve the problem by choosing an event time for interval censored subjects. If a subject experienced an event between two visits, the event time was defined to be at the midpoint of these two visits. With this assumption, the data consists of only non-censored and right censored observations, which makes it possible to use standard statistical software for the analysis. The question remained whether this assumption has a large influence on the resulting time to event model.

The researchers suspected that some patients have a higher probability of being lost to follow-up than others. Clinical experience suggests that patients' withdrawal is likely to be related to their health status. It is expected that patients who start feeling better tend to stop visiting their psychiatrists, because they think treatment is no longer needed. This suggests dependent or informative censoring. Standard survival techniques for right censoring are based on the assumption of independent censoring and non-informative censoring, and may therefore lead to biased survival estimates in this case.

1.2 Aims of this Thesis

The goal of this thesis is to address the two problems described in section 1.1 by employing the appropriate statistical methodology and to investigate the possible presence of dependent censoring in the data.

Interval Censoring Analysis

Schat et al. [24] used the midpoint of the last two visits as the event time, such that the interval censored observations were converted into non-censored observations. The first goal is to perform proper interval censoring analysis and to investigate whether this midpoint approach is a good alternative.

`R` packages `Icens` and `intcox` can be used to analyse interval censored data. `Icens` is developed to estimate Product-Limit curves. In this study the interest is on the influence of the covariates on time to event. Hence, Product-Limit estimations are not of interest. The `R` package `intcox` fits the Cox proportional hazard model which incorporates the effect of covariates on time to event. The function `intcox` does not provide standard errors for the model parameters, which makes it impossible to assess which covariates have a significant influence on the time to event. To resolve this problem, a bootstrap procedure, which can assess the precision of the `intcox` estimates, is proposed. This bootstrap method is based on the procedure described by Henschel et al. [12], but the method has been generalized so that it can be applied to any dataset to fit a Cox model with any observed covariates.

Dependent Censoring

Based on clinical experience, the researchers suspect that patients who feel better may be more likely to be lost to follow-up than others. Feeling better can be measured on many different scales derived from answers to questionnaires, e.g. general health or severity of anxiety disorder. The researchers were most interested in the influence of the BAS measurements on the probability of being censored. BAS is one of the two questionnaires that were used to define the event (50% improvement compared to baseline measurements). Therefore, this expectation can be either interpreted as *dependent censoring* or *informative censoring*. In the case of dependent censoring, the probability of being lost to follow-up is dependent on the covariates (BAS). When censoring time is directly dependent on the event time, the censoring mechanism is referred to as informative censoring.

In this thesis, the dependence on the health status is interpreted as dependent censoring. In case of dependent censoring, all covariates can be incorporated in the censoring model. Therefore, this choice enables us to find all possible covariates that have an influence on the time to censoring, and the variable BAS will not be the only one to be considered. This may lead to other measures that can be used to assess patients' health status and possibly to be associated with the probability of being censored.

In literature, only one method to correct for dependent censoring was found, the *Inverse Probability of Censoring Weighted (IPCW)* estimator. The basic idea of this estimator is to correct for censored subjects by giving extra weight to subjects who are not censored. In this way, the survival model is fit as if censoring was absent. Its interpretation is very intuitive, and therefore easy to understand for clinicians.

The IPCW method is not incorporated in standard survival packages, and is therefore difficult to apply. It requires a lot of programming from the user and careful implementation of the IPCW method. One of the novelties of this thesis is a particular way to prepare the data, which makes it possible to use standard statistical packages in `R` for the application the IPCW method. This procedure enables clinicians to apply the IPCW method, because no implementation of complicated formulas is needed. With knowledge about the standard packages available for `R`, clinicians can assess the problem of dependent censoring. This method is applied to both a test dataset and the ROM data.

A method to generate data with dependent censoring is proposed. The new method is used to generate data for a simulation study to analyse the performance of the IPCW method compared to the standard Product-Limit Estimator in case of dependent censoring. Several scenarios are considered to evaluate the bias on the classical estimator which ignores dependent censoring.

1.3 Structure of this Thesis

This thesis starts by introducing the basic concepts about survival analysis, illustrated in chapter 2. This introductory chapter is intended for readers without or with little

knowledge of survival analysis, and can be skipped by readers with experience in survival analysis. In chapter 3 the motivating example is introduced and more information about patients in the ROM dataset and on data collection is given. An initial analysis based on the midpoint approach is applied. The bootstrap method to assess the distribution of the parameter estimates for the Cox model in the presence of interval censoring is described in chapter 4. In chapter 5, the IPCW method is described. The particular way of preparing the dataset so that the standard survival package in `R` can be used to perform the IPCW method is shown in chapter 6. The method to generate dependent censoring data used in the simulation study to assess the performance of the IPCW method is described in chapter 7. All `R` code written for this thesis can be found in the appendices.

Chapter 2

Basics of Survival Analysis

In this chapter it is illustrated how survival analysis is used to estimate the time to a certain event of interest, and the basic concepts are presented. Notation is based on the notation used by Klein and Moeschberger [16].

2.1 Basic Functions in Survival Analysis

To measure the time to an event, a time origin, denoted as t_0, is defined. This is the moment at which one starts counting the time to the event of interest. This could be day of birth, moment of diagnosis, etc. Denote by t_e the moment at which the event occurs. The survival time is defined as $X = t_e - t_0$, i.e. X is the random variable that represents the time to the event of interest, with $X \geq 0$.

2.1.1 The Survival Function

The *survival function* $S(x)$ expresses *the probability that an individual survives to time* x, and is defined as

$$S(x) = P(X > x). \tag{2.1.1}$$

Let $F(x)$ be the cumulative distribution function of X, i.e. $F(x) = P(X \leq x)$. This implies that $S(x) = 1 - F(x)$. The survival function can also be defined as the integral of the probability density function $f(x)$,

$$S(x) = P(X > x) = \int_x^\infty f(t)dt. \tag{2.1.2}$$

This implies

$$f(x) = -\frac{dS(x)}{dx}. \tag{2.1.3}$$

The survival curve is monotone and non-increasing, the probability of survival is 1 at t_0 (i.e. $S(0) = 1$) and goes to 0 when time goes to infinity (i.e. $\lim_{x\to\infty} S(x) = 0$). A steep (fast decreasing) survival curve means a high likelihood of an event.

2.1.2 The Hazard Function

The *(hazard rate function)*, denoted as $h(x)$, gives *the rate at which an individual, who has survived to time x, will experience the event in the next instant of time.* The hazard function is defined as

$$h(x) = \lim_{\Delta x \to 0} \frac{P(x \leq X \leq x + \Delta x \mid X \geq x)}{\Delta x}. \tag{2.1.4}$$

If X is a continuous random variable

$$h(x) = \frac{f(x)}{S(x)} = -\frac{d \ln[S(x)]}{dx}. \tag{2.1.5}$$

Function $h(x)$ represents a rate, hence $h(x) \geq 0$.

The hazard rate curve can take many different shapes. If the curve is increasing, the chances of experiencing the particular event is low near the starting point t_0 and increases with time. Decreasing hazard rates represent events which have a higher frequency close to the starting point, compared to the frequency at later time points. The hazard rate function is not necessarily monotone.

Closely related to the hazard rate is the *cumulative hazard function* $H(x)$, which is defined as

$$H(x) = \int_0^x h(t)dt = -\ln[S(x)]. \tag{2.1.6}$$

Hence, for continuous lifetimes it follows that

$$S(x) = \exp[-H(x)] = \exp\left[-\int_0^x h(t)dt\right]. \tag{2.1.7}$$

The cumulative hazard function does not represent a probability, but a measure of the risk of the occurrence of an event.

The density, survival and hazard function provide alternative characterizations of the distribution of time to event X.

2.2 Censoring and Truncation

Survival times are often incompletely observed. This means that the time to the event of interest, t_e, is not known. This phenomenon is called *censoring*. A second feature which can be present in survival data is *truncation*. Truncation occurs when only those subjects, whose event time lies within a certain observational interval, are observed. The three types of censoring and the two kinds of truncation are defined in this section.

2.2.1 Right Censoring

Right censoring occurs when exact lifetimes are only known if the event happens before a certain (pre specified) time point, the *censoring time* C_r. If the event does not occur before the censoring time, it will not be observed and the subject is *censored*.

If an individual is right censored, it is known that the event has not occurred before C_r, and, consequently, the event may happen *after* the censoring time, i.e. $X > C_r$. Thus, the event may occur in the time interval (C_r, ∞). An example of right censored data is presented in Figure 2.1.

In this figure both subject 1 and 3 are censored, therefore the event of interest is not observed. Subject 2 is not censored, since his or her exact survival time t_{e_2} is observed.

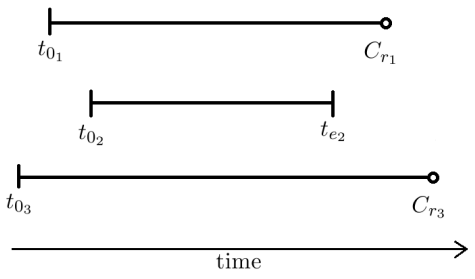


Figure 2.1: Visualization of right censoring. t_{0_i} represents time entry corresponding to subject i. t_{e_i} and C_{r_i} are respectively time to event (if it was observed) or time to censoring (if subject i was censored).

Survival data can be represented as pairs of random variables (T, δ), a pair for each subject in the study. δ indicates whether the event was observed or not, i.e. $\delta = 0$ for censored subjects and $\delta = 1$ if the event is observed. The random variable T represents the survival time X of a subject if the event is observed, and the censoring time C_r of the subject if the event is not observed, i.e. $T = \min(X, C_r)$. In Table 2.1 the data representation for the individuals illustrated in Figure 2.1 is given.

Subject	data (T, δ)
1	$(C_{r_1}, 0)$
2	$(t_{e_2}, 1)$
3	$(C_{r_3}, 0)$

Table 2.1: Data representation for the subjects illustrated in Figure 2.1.

2.2.2 Left Censoring

Left censoring is comparable to right censoring. However, for left censoring it is known that the event of interest happened *before* are certain time point C_l.

For example, if a subject experienced the event before his or her inclusion in the study, he or she is left censored. Sometimes the subject may be able to recall when the event occurred precisely. If the precise moment cannot be recalled, the only information is that the event happened before entering the study. Hence, in case of left censoring the event occurred in the interval $[0, C_l]$.

Left censored data can be represented as right censored data, i.e. by the pair (T, δ). Here, δ indicates whether the exact lifetime X is observed ($\delta = 1$) or not ($\delta = 0$). T represents the survival time X if the event is observed, and the censoring time C_l if the event is not observed, i.e. $T = \max(X, C_l)$.

2.2.3 Interval Censoring

For *interval censored* data the precise moment of the event is known to be within a time interval $(L_i, R_i]$. Thus, the event took place *after* time point L_i, but *before or at* R_i.

This type of censoring may occur when periodic inspections are done, for example, if a patient visits a doctor every few months. If at a specific visit it is detected that a patient has experienced the event, e.g. recovery, then it is not known when the event happened exactly. The patient may have recovered only a day before the current visit, or on the day after the last visit, or sometime in between. The exact event time is unknown, but it falls into the time interval between the last two visits.

All three types of censoring can be interpreted as interval censored data. Right censoring can be represented by (C_r, ∞) and left censoring by $[0, C_l]$. If exact event times are known, these can be denoted as the interval $[X, X]$. If information on a combination of uncensored subjects and right, left and interval censored subjects is gathered, this interval notation enables storage of data concerning all these types of subjects in the same manner. The observed data can be represented by an interval (either $[X, X]$, $[0, C_l]$, (C_r, ∞), or $(L, R]$) in which the event happened or may happen.

2.2.4 Left Truncation

Left truncation occurs when subjects are entered if their event time exceeds a certain left truncation time, Y_L. For example, if only patients from a certain age, who have not experienced the event of interest yet, are included in a study, patients who experience the event before this age are not observed. They are left truncated.

2.2.5 Right Truncation

Right truncation occurs when only subjects who experience the event of interest before a certain right truncation time, Y_R, are included in a study. For example, if patients who are diagnosed with a certain disease (= event) are included in a study only if they are below a pre specified age.

2.3 Likelihood Construction for Censored and Truncated Data

When constructing the likelihood function for censored and truncated data, the censoring and truncation mechanisms should be considered. An important assumption in the formulation of the likelihood is the independence between lifetimes and censoring times. In section 2.6 this aspect will be further discussed.

The likelihood function consists of the following components:

exact lifetimes	-	$f(x)$
right-censored observations	-	$S(C_r)$
left-censored observations	-	$1 - S(C_l)$
interval-censored observations	-	$[S(L) - S(R)]$
left-truncated observations	-	$f(x)/S(Y_L)$
right-truncated observations	-	$f(x)/[1 - S(Y_R)]$
interval-truncated observations	-	$f(x)/[S(Y_L) - S(Y_R)]$

The likelihood for censored data is constructed by combining the above parts as

$$L \propto \prod_{i \in D} f(x_i) \prod_{i \in R} S(C_{r_i}) \prod_{i \in L} (1 - S(C_{l_i})) \prod_{i \in I} (S(L_i) - S(R_i)), \tag{2.3.1}$$

where

D: set of individuals who experienced the event of interest;
R: set of right censored observations;
L: set of left censored observations;
I: set of interval censored observations.

If the data is left truncated with truncation intervals (Y_{L_i}, Y_{R_i}), which are independent from the event times, $f(x)$ is replaced by $f(x)/S(Y_L)$ and $S(C_i)$ by $S(C_i)/[S(Y_L) - S(Y_R)]$ in (2.3.1).

2.4 Estimation of the Survival and Cumulative Hazard Functions

The survival function and the cumulative hazard function can be estimated by a parametric or non-parametric approach. Both methods are briefly described in this section.

2.4.1 Non-Parametric Methods

For each of the n individuals in a dataset, a data pair represents a time point t_i and an indicator δ_i of whether this time point an event time or a left or right censoring time. Suppose that all time points t_i are distinct, i.e. $t_1 < t_2 < \cdots < t_n$. Furthermore, define d_i as the number of deaths at time point t_i. Let Y_i be the number of individuals that are at risk at time point t_i. The ratio d_i/Y_i gives an estimate of the chance of experiencing the event at time t_i. This ratio is used for the two non-parametric estimators described in this section.

Product-Limit Estimator

Kaplan and Meier proposed the *Product-Limit Estimator* as an estimator for the survival function (often referred to as the *Kaplan-Meier Estimator*) [15]. This estimator is defined

as

$$\hat{S}(t) = \begin{cases} 1 & \text{if } t < t_1, \\ \prod_{t_i \leq t} \left[1 - \frac{d_i}{Y_i}\right] & \text{if } t \geq t_1 \end{cases}. \tag{2.4.1}$$

The interpretation of this estimator is as follows. If an individual does not experience the event before time t, he or she should not experience the event at all time points $t_i \leq t$. Since $\frac{d_i}{Y_i}$ is an estimate of the chance of experiencing the event at time point t_i, $[1 - \frac{d_i}{Y_i}]$ is an estimate of the probability of experiencing the event at this time point, i.e. $\hat{P}(X > t_i)|X \geq t_i) = 1 - \frac{d_i}{Y_i}$.

Note that

$$\begin{aligned} S(t_i) &= \tfrac{S(t_i)}{S(t_{i-1})} \tfrac{S(t_{i-1})}{S(t_{i-2})} \cdots \tfrac{S(t_2)}{S(t_1)} \tfrac{S(t_1)}{S(t_0)} S(t_0) \\ &= P(X > t_i | X \geq t_i) P(X > t_{i-1} | X \geq t_{i-1}) \cdots \\ &\quad P(X > t_2 | X \geq t_2) P(X > t_1 | X \geq t_1), \end{aligned}$$

because $S(0) = 1$, and $S(t_{i-1}) = P(X > t_i) = P(X \geq t_i)$ for discrete distribution S. Simplifying this gives $\prod_{t_i \leq t}[1 - \frac{d_i}{Y_i}]$, the Product-Limit estimator.

This estimator results in a step function with jumps at each of the time points t_i. The size of the jumps depends on the number of events, as well as the number of censored subjects at that time point. This estimator is well defined for values of t up to the largest observed event time.

The Product-Limit estimator can be used to estimate the cumulative hazard function. $H(t) = -\ln[S(t)]$, so

$$\hat{H}(t) = -\ln[\hat{S}(t)], \tag{2.4.2}$$

with $\hat{S}(t)$ as defined in (2.4.1).

Nelson-Aalen Estimator

Nelson [18] first suggested this estimator for the cumulative hazard function and Aalen [1] rediscovered it. The Nelson-Aalen estimator is defined as

$$\tilde{H}(t) = \begin{cases} 0 & \text{if } t \leq t_1, \\ \sum_{t_i \leq t} \frac{d_i}{Y_i} & \text{if } t \geq t_1 \end{cases}. \tag{2.4.3}$$

It is the sum over the probabilities of experiencing the event at $t_i \leq t$. It is well defined up to the largest observed time.

By employing (2.4.3) as the cumulative hazard function, the Nelson-Aalen estimator for the survival function can be calculated as

$$\tilde{S}(t) = \exp[-\tilde{H}(t)]. \tag{2.4.4}$$

2.4.2 Parametric Methods

Parametric distributions can be used as a model to fit the survival data by assuming that the survival times are distributed in a parametric way. Commonly used distributions are listed in Table 2.2.

Distribution	Hazard Rate $h(x)$	Survival Function $S(x)$	Probability Density Function $f(x)$
Exponential $\lambda > 0, x \geq 0$	λ	$\exp[-\lambda x]$	$\lambda \exp[-\lambda x]$
Weibull $\alpha, \lambda > 0, x \geq 0$	$\alpha\lambda x^{\alpha-1}$	$\exp[-\lambda x^{\alpha}]$	$\alpha\lambda x^{\alpha-1} \exp[-\lambda x^{\alpha}]$
Log normal $\sigma > 0, x \geq 0$	$\frac{f(x)}{S(x)}$	$1 - I(\lambda x, \beta)$	$\frac{\lambda^{\beta} x^{\beta-1} \exp[\lambda x]}{\Gamma(\beta)}$

Table 2.2: Commonly used parametric distributions in survival analysis.

2.5 Semi-Parametric Proportional Hazards Regression

Often, one of the goals of research is to investigate which covariates influence the chance of survival. For example, one might want to compare the effect of treatments, or the influence of some demographical characteristics. Cox [6] proposed the *Proportional Hazards Model* to incorporate covariates in the survival model.

2.5.1 Time-Independent Covariates

Let $\boldsymbol{Z} = (Z_1, \ldots, Z_p)$ be the vector of p covariates. Assume that all covariates are time independent, i.e. the values Z_k, for $k = 1, \ldots, p$, do not change over time. This yields to a dataset consisting of triplets $(T_j, \delta_j, \boldsymbol{Z}_j)$, with $j = 1, 2, \ldots, n$, and n the number of subjects.

The proportional hazards model is defined as

$$h(t|\mathbf{Z}) = h_0(t)c(\boldsymbol{\beta}^t\mathbf{Z}), \tag{2.5.1}$$

where $h_0(t)$ is an arbitrary baseline hazard rate, $\boldsymbol{\beta}^t = (\beta_1, \ldots, \beta_p)$ is the parameter vector, and $c(\boldsymbol{\beta}^t\mathbf{Z})$ is a known function. No structure is imposed on the *baseline hazard* $h_0(t)$ which gives the model a great flexibility. A common model for $c(\boldsymbol{\beta}^t\mathbf{Z})$ is

$$c(\boldsymbol{\beta}^t\mathbf{Z}) = \exp(\boldsymbol{\beta}^t\boldsymbol{Z}) = \exp\left[\sum_{k=1}^{p} \beta_k Z_k\right], \tag{2.5.2}$$

yielding

$$h(t|\boldsymbol{Z}) = h_0(t)\exp(\boldsymbol{\beta}^t\boldsymbol{Z}) = h_0(t)\exp\left[\sum_{k=1}^{p} \beta_k Z_k\right]. \tag{2.5.3}$$

The survival function can be derived by using expressions (2.5.3) and (2.1.7):

$$\begin{aligned} S(t|\boldsymbol{Z}) &= \exp\left[-\int_0^x h(u|\boldsymbol{Z})du\right] \\ &= \exp\left[-\int_0^x h_0(u)\exp(\boldsymbol{\beta}^t\boldsymbol{Z})du\right] \\ &= \exp\left[-\int_0^x h_0(u)du\exp(\boldsymbol{\beta}^t\boldsymbol{Z})\right] \\ &= \exp\left[-\int_0^x h_0(u)du\right]^{\exp(\boldsymbol{\beta}^t\boldsymbol{Z})} \\ &= S_0(t)^{\exp(\boldsymbol{\beta}^t\boldsymbol{Z})}. \end{aligned} \tag{2.5.4}$$

This function is in the likelihood construction of the proportional hazards model.

The survival chances of individuals with covariate values $\boldsymbol{Z}$ and $\boldsymbol{Z}^*$ can be compared by looking at the proportion of their hazards:

$$\frac{h(t|\boldsymbol{Z})}{h(t|\boldsymbol{Z}^*)} = \frac{h_0(t)\exp\left[\sum_{k=1}^p \beta_k Z_k\right]}{h_0(t)\exp\left[\sum_{k=1}^p \beta_k Z_k^*\right]} = \exp\left[\sum_{k=1}^p \beta_k(Z_k - Z_k^*)\right], \tag{2.5.5}$$

which is a constant. (2.5.5) is called the *hazard ratio* and it represents the *relative risk* of an individual with risk factor $\boldsymbol{Z}$ experiencing the event as compared to an individual with risk factor $\boldsymbol{Z}^*$. Each regression coefficient β_k for $k = 1, \ldots, p$ in the model indicates the change in the relative risk if Z_k is increased by one unit (for quantitative covariates), or the difference in relative risk between the reference group and other groups (for categorical covariates).

2.5.2 Time-Dependent Covariates

The proportional hazards model can be extended to incorporate time-dependent covariates, $\boldsymbol{Z}(t)$. The data consists of the triple $(T_j, \delta_j, [\boldsymbol{Z}_j(t), 0 \leq t \leq T_j])$ for $j = 1, \ldots, n$. Here, T_j is the time on study for patient j, δ_j is the event indicator for patient j and $\boldsymbol{Z}_j(t) = (Z_{j1}(t), \ldots, Z_{jp}(t))$ is the vector of covariates for patient j. It is assumed that the value of $\boldsymbol{Z}_j(t)$ is known for any time at which the subject is under observation. The proportional hazards model including time-dependent and time fixed covariates is defined as

$$h(t|\boldsymbol{Z}(t)) = h_0(t)\exp(\boldsymbol{\beta}^t\boldsymbol{Z}(t)) = h_0(t)\exp\left[\sum_{k=1}^p \beta_k Z_k(t)\right]. \tag{2.5.6}$$

2.6 Censoring Assumptions

The theory described is this chapter is based on several assumptions about the censoring mechanism, *random censoring*, *independent censoring*, and *non-informative censoring*. If these assumptions do not hold, standard survival theory does not apply and a proper analysis should be performed. The three types of censoring assumptions have similarities, but are not identical. However, the definitions are often mixed up, and the term *informative censoring* is frequently used to refer to any of the three types of censoring.

For this thesis, the definition stated by Kleinbaum and Klein [17] are assumed:

Random Censoring: Subjects censored at time t are representative of all subjects who remain at risk at time t with respect to their survival experience. Hence, the chance of experiencing an event should be the same for both censored subjects and subjects remaining at risk. It is as if the censored subjects were randomly selected from all subjects.

Independent Censoring: Within any subgroup of interest, subjects censored at time t are representative of all the subjects in their subgroup who remain at risk at time t with respect to their survival experience. Hence, it is as if the censored subjects were randomly selected from the subgroup.

Non-Informative Censoring: For each individual assume there is a potential random censoring time C and a potential random lifetime X. Let X and C be respectively the random variables describing the distributions of time to event and time to censoring. When the distribution of the survival times X provides no information about the distribution of censoring C, and vice versa, the censoring is said to be non-informative.

A situation where the non-informative censoring condition is met is when subjects are repeatedly observed at scheduled visits, but modifications in the schedule are not related to the event time. Examples of unrelated modifications are changes due to illness of the researcher, intervening appointments unrelated to the status of the subject, etc.

Chapter 3

Motivating Example

Since 2002, the regional mental health care provider Rivierduinen and the psychiatry department at Leiden University Medical Center (LUMC) have been working together on *Routine Outcome Monitoring (ROM)*. ROM is a method to gather information concerning the results of a treatment by repeatedly measuring severity of disease and patients' complaints. The primary goal of the ROM procedure is to diagnose patients and to inform clinicians and patients about the treatment progress. The data was anonymized for research purposes. The data includes right and interval censored observations. In this chapter, the dataset is described and an initial analysis is illustrated. In later chapters, analyses incorporating interval and dependent censoring will be performed.

3.1 Background Information

Mood, anxiety and somatoform (MAS) disorders are highly prevalent in the European Union [31]. Patients with these disorders experience high disease burden and often recover slowly. Individuals suffer from high disability, long duration of the illness and a high risk of recurrence. Many studies have been done to find (baseline) predictors for poor outcome. With information on baseline predictors, patients that need extra attention or special treatments can be identified at diagnosis. Several predictors as being unmarried, unemployment, a low level of education, and personality disorders, seem to have a negative influence on the recovery process [24, 29].

Most of the published results are based on randomized controlled trails (RCT's) which usually focus on a narrowly defined patient population. These trails use strict in- and exclusion criteria which often result in the exclusion of patients suffering from more than one MAS disorder (comorbidity) and those who use drugs or alcohol. However, the real-world patients often suffer from multiple disorders and regularly use drugs or alcohol. Hence, the results of the RCT's cannot be generalized to patients seen in everyday clinical practice. In the large-scale ROM study, information was gathered on a large naturalistic cohort of outpatients suffering from MAS disorders. The in- and exclusion criteria are less strict, and therefore the results based on this study can be extended to other patients populations.

The ROM procedure was applied to approximately 80% of the patients at Rivierduinen and LUMC suffering from MAS disorders. Insufficient knowledge of the Dutch language was an exclusion criteria. Furthermore, patients for who ROM was expected to be too demanding were excluded.

3.2 Data Collection

Patients included in the ROM study repeatedly visited the health care providers. Several self-reported and observer-rated questionnaires were used to measure the severity of disease and to gather patients' characteristics at baseline and during follow-up. In Table 3.1, the demographical variables registered for each patient are listed.

Variable	**Categories (if applicable)**
Age	In years
Gender	Male Female
Ethnicity	Dutch (both parents born in the Netherlands) Non-Dutch
Marital status	Married or living together Not married or cohabiting
Living situation	Living independently with partner and/or children Living independently and alone Residing with family
Education Level	High Medium Low
Occupation	Daily occupation No daily occupation
Single Anxiety	Single anxiety disorder Multiple anxiety disorders
Alcohol	No alcohol abuse or dependence Alcohol abuse or dependence
Drug	No drug abuse or dependence Drug abuse or dependence
Age of onset	Adult onset Pre-adult onset

Table 3.1: Demographical variables registered for each patient in the ROM procedure.

There are both generic and disorder specific questionnaires. The generic questionnaires were given to all patients, but the disorder specific ones only to patients who were

diagnosed with a specific disorder. All questionnaires used in this study are commonly used questionnaires. The questions are standardized and given answers are translated to scores on several scales which indicate patient's characteristics or severity of complaints. In this way the severity of complaints are represented by a number.

A summary of several measuring instruments used during ROM is given in Table 3.2. This list is not complete, it contains only tests that are relevant for the analyses performed in this thesis. More information can be found in the documentation about ROM, see [7]. Note that for the questionnaires BAS, MADRS, REM, BSI-12 and DAPP-SF, reported in Table 3.2, a high score indicates a high severity of the disorder. However, high scores on the SF36 scale indicate good quality of life. To make outcomes easier to compare, scores on SF36 scale were reversed for analysis. At baseline, Mini-Plus was used to diagnose the patient, i.e. determine his or her disorder. All questionnaires were administered by independent and trained nurses.

Measuring Instrument	**Description**
Mini-plus:	A standardized and structured interview which is used to determine whether a patient satisfies the DSM-IV (Diagnostic and Statistical Manual of Mental Disorders, fourth edition) criteria for a number of psychiatric disorders.
BAS:	A 10-item observer-rated scale derived from the abbreviated Comprehensive Psychopathological Rating Scale (CPRS). A high score indicates much tension and anxiety.
MADRS:	An observer-rated scale derived from the abbreviated Comprehensive Psychopathological Rating Scale (CPRS). A high score indicates many complaints about depression.
REM:	An observer-rated scale derived from the abbreviated Comprehensive Psychopathological Rating Scale (CPRS). A high score indicates high motivational inhibition.
BSI-12:	A self-reported measure comprising items of the anxiety and somatization subscales of the Brief Symptom Inventory 18-item version (BSI-18).
DAPP-SF:	A shorter version of the self-reported questionnaire on personality disorders Dimensional Assessment of Personality Psychopathology (DAPP-BQ). A high score on a subscale indicates a personality disorder for that subscale.
SF36:	A self-reported questionnaire which measures the quality of life of a patient. It contains several subscales and a high score on a subscale indicates good quality of life for that specific subscale.

Table 3.2: Short summary of several measuring instruments used during Routine Outcome Monitoring (ROM).

3.3 Research Purpose

Schat et al. [24] indicate some patients' baseline characteristics which predict a small chance of recovery or a slow recovery from an anxiety disorder. With this information, patients with a small chance of recovery can be identified at baseline and can be given extra or special treatment. Baseline predictors are treatment independent, therefore no information about the treatment and therapist characteristics were taken into account.

Since the focus was on patients with anxiety disorders, only patients with moderate to severe anxiety disorders at baseline were included in this study, i.e. only a subset of the ROM dataset was used for analysis. Moderate to severe anxiety disorders were defined as bigger or equal to 10.38 on the BAS-scale *and* bigger or equal to 6 on the BSI-12 scale at baseline. Possible anxiety disorders are *panic disorder with or without agoraphobia (PD/A)*, *agoraphobia (AP)*, *social phobia (SP)*, and *generalized anxiety disorder (GAD)*.

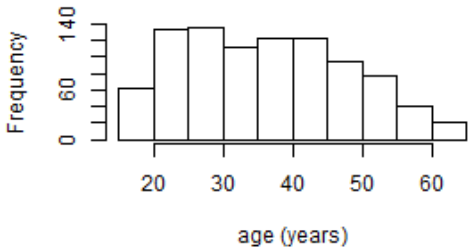


Figure 3.1: Age distribution of the subjects suffering from moderate to severe anxiety disorders.

The original dataset contained 1889 patients with moderate to severe anxiety disorders. After excluding subjects who had no follow-up assessment or had missing data (resulting from the incidental failure to administer complete questionnaires), 917 patients were left for analysis. The majority (64.1%) of the patients in the dataset is female. The histogram in Figure 3.1 gives an indication of the age distribution among the patients.

3.4 Survival Analysis Method and Results

To find the baseline characteristics that influence recovery, survival analysis can be employed. The analysis described in this section is a modified version of the analysis by Schat et al. [24]. For example, the model in this analysis corrects for the baseline severity of the disorder by allowing the baseline measurement on the BAS or the BSI-12 to be included in the model.

A semi-parametric proportional hazards regression (section 2.5) is used to assess the influence of baseline covariates on recovery. To make the resulting hazard ratios comparable, Z-scores of the continuous variables were calculated and further analysis is based on these scores.

The following definitions are made for analysis.

Event of interest The event of interest is defined as the first time that there is at least 50% improvement, compared to the baseline measurements, on the

BSI-12 and the BAS scales simultaneously.

Time origin The time origin t_0 is the day of intake at which the Mini-plus detected one or more anxiety disorders.

Event time The exact time to event is not observed since the measurements are done at periodic inspections. If an event is observed, it is known that it happened between the last two visits. In the preliminary analysis the midpoint between the two last visits is taken as event time t_e.

Censoring time If a patient is lost to follow-up, the date of his or her last visit is taken as censoring time. If a subject has not experienced the event after two years, the disorder is assumed to be chronic and the subject is censored (at 730 days).

The first step in the analysis is to fit an univariate regression model for each of the covariates. The influence of the baseline BAS and BSI-12 measurements is also assessed, since they can correct the results for baseline severity. All variables that achieve significance levels of ≤ 0.10 in univariate analysis are entered in the multivariate analysis. However, the covariates age, gender and the four dichotomized main diagnostic categories (PD/A, AP, SP and GAD) are always incorporated in the model. Covariates that achieved significance (p-value ≤ 0.10) in the univariate analyses are shown in Table 3.3.

Results show that the BSI-12 measurement at baseline does not influence the outcome significantly, while the BAS measurement at baseline does. Both measurements indicate the baseline severity of the anxiety disorder(s). Since it was expected that the baseline severity influences the recovery time, it is remarkable that only the baseline measurement of BAS has a significant effect on time to improvement. Since BSI-12 measurement has no significant influence, but BAS does, only the BAS baseline measurement is included in the multivariate analysis.

All variables shown in Table 3.3 are included in the first multivariate model. The backward stepwise procedure is performed. Variables are removed from the model if its p-value is ≥ 0.10, and if removal does not increase the AIC value of the model. Variables that remain in the model after this procedure are given in Table 3.4.

Univariate estimated survival functions for some significant variables included in the final model are illustrated in Figure 3.2. For *conduct problems* three groups (low, medium and high) are created.

bodily pain (SF36)	general health (SF36)
living situation	education
occupation	ethnicity
alcohol	BAS
submissiveness (DAPP-SF)	cognitive distortion (DAPP-SF)
identity problems (DAPP-SF)	affective lability (DAPP-SF)
stimulus seeking (DAPP-SF)	oppositionality (DAPP-SF)
intimacy problems (DAPP-SF)	anxiousness (DAPP-SF)
conduct problems (DAPP-SF)	suspiciousness (DAPP-SF)
social avoidance (DAPP-SF)	insecure attachment (DAPP-SF)
self-harm (DAPP-SF)	agoraphobia (DSM-IV-TR)
panic disorder (DSM-IV-TR)	single anxiety

Table 3.3: Covariates that attained significance at 0.10 in the univariate analyses on the right censored data.

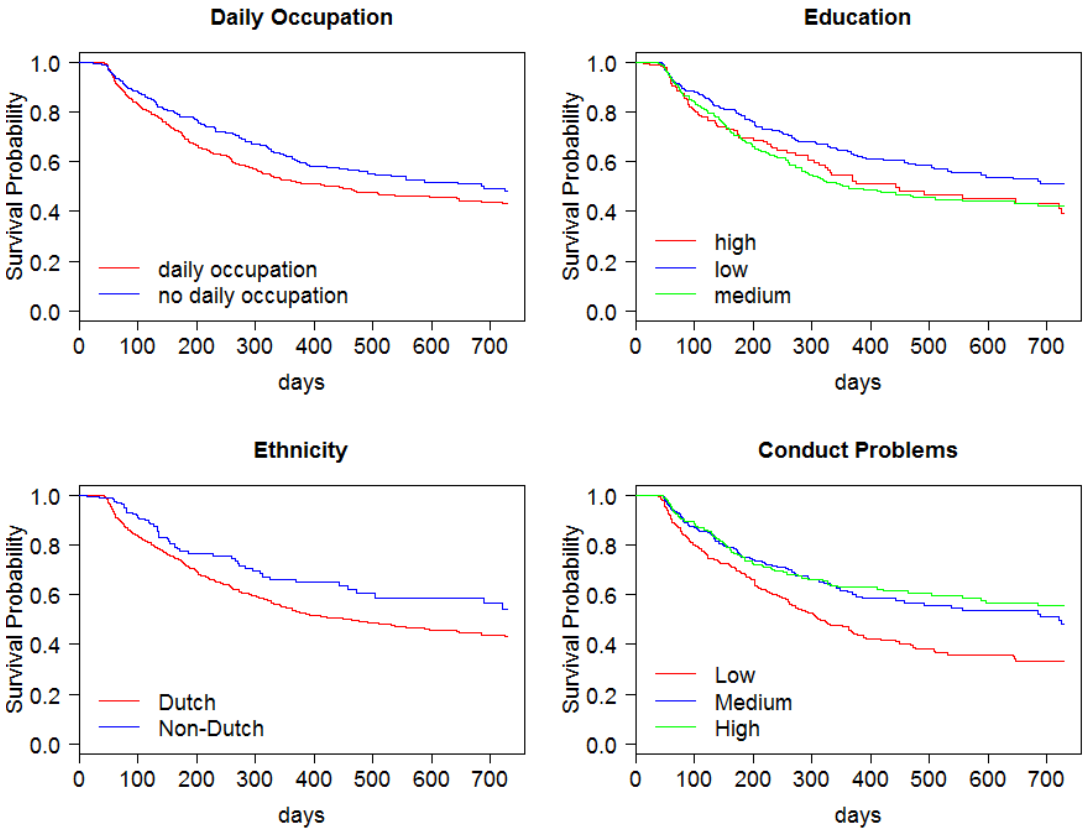


Figure 3.2: Product-Limit curves for some of the significant variables included in the final multivariate model derived in section 3.4.

Covariate	HR	95% CI	p-value
Age	0.989	0.872 - 1.122	0.867
Gender			
- female	1		
- male	1.153	0.905 - 1.468	0.249
DSM-IV-TR agoraphobia			
- no AP	1		
- AP	0.741	0.467 - 1.175	0.202
DSM-IV-TR panic disorder with or without agoraphobia			
- no PD	1		
- PD	1.187	0.799 - 1.762	0.396
DSM-IV-TR social phobia			
- no SP	1		
- SP	1.060	0.727 - 1.546	0.760
DSM-IV-TR generalized anxiety disorder			
- no GAD	1		
- GAD	1.162	0.787 - 1.714	0.450
SF36 general health	0.888	0.791 - 0.997	0.044
Living situation			
- independently with partner and/or children	1		
- independently alone	0.993	0.756 - 1.304	0.959
- with family	1.455	1.045 - 2.025	0.027
Education			
- high	1		
- medium	0.959	0.719 - 1.278	0.775
- low	0.719	0.531 - 0.974	0.033
Occupation			
- daily occupation	1		
- no daily occupation	0.751	0.599 - 0.942	0.013
Ethnicity			
- Dutch	1		
- non-Dutch	0.688	0.508 - 0.931	0.015
Alcohol abuse or dependence			
- no alcohol abuse or dependence	1		
- alcohol abuse or dependence	0.572	0.288 - 1.134	0.110
BAS	1.316	1.180 - 1.468	0.000
DAPP-SF affective lability	0.825	0.729 - 0.933	0.002
DAPP-SF oppositionality	0.912	0.807 - 1.030	0.137
DAPP-SF conduct problems	0.857	0.745 - 0.985	0.030
Number of simultaneous DSM-IV-TR anxiety disorders			
- single anxiety disorder	1		
- multiple anxiety disorders	0.797	0.590 - 1.076	0.138

Table 3.4: Multivariate hazard ratios of response (50% improvement on both the BAS and BSI-12 scales) with their 95% confidence interval and p-value.

Chapter 4

Interval Censoring

In the preliminary analysis discussed in chapter 3, a simple approach has been taken by assuming the midpoint between the last two visits as time to event, t_e. An appropriate analysis requires the use of specific techniques developed for interval censored data. The aim of the analyses in this chapter is to assess whether using the midpoint between the two consecutive visits is a good substitute for interval censoring techniques for the ROM dataset. Therefore, some survival techniques on interval censoring will be discussed. First, a short theoretical introduction is given. Next, R packages `Icens` and `intcox` are used to, respectively, estimate Product-Limit curves and a Cox proportional hazard model for the ROM dataset. Finally, a bootstrapping procedure is described to assess the precision of the estimated parameters for the Cox proportional hazard model.

4.1 Theory on Interval Censoring

For interval censoring, events are known to occur within an interval between two time points, i.e. event time t_e is within $(L, R]$. There are several types of interval censored data:

Case I interval censored data (*current status data*) The subject is observed only once, hence time to event X is known to be smaller or larger than the observed inspection time C. This results in either a left ($X \leq C$) or right censored ($X > C$) observation.

Case II interval censored data Subjects are observed at two time points, U and V with $U < V$. It is observed whether the event was before the first visit ($X \leq U$), between the two visits ($U < X \leq V$) or after the last visit ($X > V$).

Case *K* interval censored data Subjects are observed K times, hence it is only observed that the event has occurred between two consecutive observation moments.

In the ROM dataset, outpatients were observed at repeated visits. If a patient was observed at baseline only, he or she was excluded from the analysis. Hence, the dataset

contains Case K interval-censored data, with $K \geq 2$. In order to perform interval censoring analysis on the ROM dataset, the observable data is denoted as $D = \{(l_1, r_1], (l_2, r_2], \ldots, (l_n, r_n]\}$, with n the number of subjects. Right censoring is represented by the event interval (C_r, ∞), left censoring by $[0, C_l)$ and interval censoring by $[L, R]$.

Likelihood Construction

The likelihood for censored data is constructed as (2.3.1) in section 2.3. Using the notation above, the likelihood for right, left and interval censored data can be rewritten as

$$L(S(\cdot)|D) \propto \prod_{i \in R} S(l_i) \prod_{i \in L} (1 - S(r_i)) \prod_{i \in I} (S(l_i) - S(r_i)). \tag{4.1.1}$$

For left censored observations $l_i = 0$, so $S(l_i) - S(r_i) = 1 - S(r_i)$. This corresponds to the left censored part of the likelihood in (4.1.1). For right censoring $r_i = \infty$, hence $S(l_i) - S(r_i) = S(l_i) - 0 = S(l_i)$. This corresponds to the right censored part of the likelihood in (4.1.1). Therefore, the likelihood can be simplified as

$$L(S(\cdot)|D) \propto \prod_{i=1}^{n} (S(l_i) - S(r_i)). \tag{4.1.2}$$

The likelihood in (4.1.2) only holds under the assumption of random, independent, and non-informative censoring (see section 2.6).

Likelihood Maximization

When calculating the maximum likelihood estimator (MLE), the goal is to find a monotonically decreasing function $\hat{S}(x)$ which maximizes the likelihood function $L(S(\cdot)|D)$ in (4.1.2). If this is done non-parametrically, the solution is called a *non-parametric maximum likelihood estimator (NPMLE)*. A popular method to obtain a NPMLE for the survival function for interval censored data is the algorithm proposed by Turnbull [28]. This algorithm is a special case of the EM algorithm.

Turnbull's Algorithm

Let $0 = \tau_0 < \tau_1 < \cdots < \tau_m$ be the grid of all time points l_i, r_i for $i = 1, \ldots, n$ in data D. Define α_{ij} as the indicator of whether the event, which occurred in the interval $(l_i, r_i]$, could have occurred at τ_j, $j = 1, \ldots, m$. α_{ij} is equal to 1 if $(\tau_{j-1}, \tau_j] \subset (l_i, r_j]$ and 0 otherwise. Initial values are given to $S(\tau_j)$ by distributing the mass of $\frac{1}{n}$ for the ith individual equally to each possible value $\tau_j \in (l_i, r_i]$. The algorithm is as follows:

Step 1: Compute the probability p_j that an event occurred at time τ_j by $p_j = S(\tau_{j-1}) - S(\tau_j)$, $j = 1, \ldots, m$.

Step 2: Estimate the number of events occurring at τ_j by

$$d_j = \sum_{i=1}^{n} \frac{\alpha_{ij} p_j}{\sum_k \alpha_{ik} p_k}.$$

Step 3: Compute the estimated number at risk at time τ_j by $Y_j = \sum_{k=j}^{m} d_k$.

Step 4: Update the Product-Limit estimator (section 2.4.1) using the values of d_j and Y_j found in steps 2 and 3. Stop the iterative process if the updated estimate of S is close to the old version of S for all τ_j's, otherwise repeat steps 1-3 using the updated estimate of S.

The NPMLE for the survival function is a decreasing step function, with gaps corresponding to the unidentifiability of the function within each interval $(\tau_{j-1}, \tau_j]$. These gaps can be filled by making additional assumptions, for example left continuity.

The convergence of Turnbull's algorithm can be very slow for datasets with large sample sizes. More efficient algorithms to calculate the NPMLE have been proposed, e.g. the ICM [11, 19] and EM-ICM [30]. A short introduction to these algorithms can be found in [10]. The main advantage of these algorithms is that their global convergence is guaranteed.

Cox Proportional Hazards Model for Interval Censored Data

To assess the influence of covariates on the survival time, the Cox proportional hazards model can be used. In terms of the hazard function, this model is defined as

$$h(t|\boldsymbol{Z}) = h_0(t) \exp(\boldsymbol{\beta}^t \boldsymbol{Z}) = h_0(t) \exp\left[\sum_{k=1}^{p} \beta_k Z_k\right], \tag{4.1.3}$$

(see section 2.5). Here, $\boldsymbol{Z}$ is the vector of covariates and $h_0(t)$ is an unknown baseline hazard function. The proportional hazards model can be defined in terms of the survival function as

$$S(t) = S_0(t)^{\exp(\boldsymbol{\beta}^t \boldsymbol{Z})}, \tag{4.1.4}$$

where $S_0(t)$ is the baseline survival function. Using this notation, the simplified likelihood (4.1.2) for interval censored data is as follows:

$$L(S(\cdot), \boldsymbol{\beta}|D) \propto \prod_{i=1}^{n} \left(S_0(l_i)^{\exp(\boldsymbol{\beta}^t \boldsymbol{Z})} - S_0(r_i)^{\exp(\boldsymbol{\beta}^t \boldsymbol{Z})} \right). \tag{4.1.5}$$

The goal of MLE is to find a function $S_0(t)$ and coefficients $\boldsymbol{\beta}$ that maximize the likelihood $L(S(\cdot), \boldsymbol{\beta}|D)$. Pan [19] extended the ICM algorithm to obtain the NPMLE of $S_0(t)$ and $\boldsymbol{\beta}$ by assuming that $S_0(t)$ is piecewise constant. Simulation studies have shown a slight positive bias in the regression coefficients $\boldsymbol{\beta}$ estimated with this algorithm.

4.2 Estimation of the Survival Curve for Interval Censored Data with `R`

The R package `Icens` is developed to estimate the survival function $S(x)$ for interval censored data. It incorporates many functions for computing the NPMLE for censored and truncated data, [9].

The ROM dataset consists of right and interval censored data, therefore the `Icens` package can be used for this dataset. Intervals $(l, r]$ for each observed event are constructed, and right censored observations are represented by the interval $(c, \infty]$ with c the censoring time.

Since the EM-ICM algorithm is more efficient in estimating survival curves than the EM algorithm, the function `EMICM` from the package `Icens` was used to fit the model for the ROM data. In Figure 4.1 the estimated survival curves are shown for four variables that were significant in the multivariate Cox model in section 3.4. These curves suggest that the chances of survival are different for several baseline categories.

The survival probabilities of categories appear to differ, as in section 3.4. However, the difference in survival between the groups is less obvious due to the uncertainties caused by the intervals.

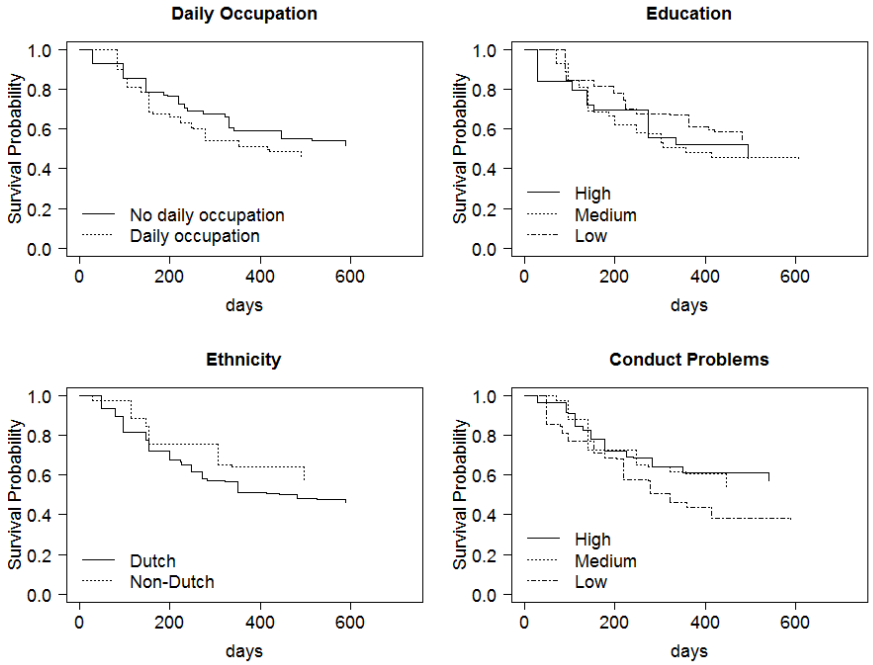


Figure 4.1: Estimated survival curves for several significant variables incorporated in the multivariate Cox model section 3.4.

4.3 Estimating the Cox Model for Interval Censored Data with R

The R package `intcox` [13] can be used to estimate the coefficients $\boldsymbol{\beta}$ and the baseline hazard function $h_0(t)$ in the proportional hazards model for interval-censored data. The algorithm maximizes the log-likelihood with a modified Newton-Raphson algorithm. More details on the mathematical background can be found in [12]. The starting values of the algorithm are calculated by treating the data as right censored data and using the classical proportional hazards method to fit the first model. An event within the interval $(l, r]$ is treated as an event at r and a right censored observation with interval representation $(c, \infty]$ is treated as a right-censored observation at c.

Unfortunately, there are several problems with the algorithm which make `intcox` less useful.

During simulations studies, the authors observed problems for data with a high percentage of right censored data (over 30%). In this case the likelihood, computed from the starting values, cannot be improved by the ICM algorithm. Hence, the algorithm does not move away from the starting values.

Additionally, simulation studies showed that the regression coefficients β estimated by `intcox` are light positively biased. This has to be taken into account when the

estimates are used for interpretation.

Furthermore, the function `intcox` does not provide standard errors for the model parameters $\boldsymbol{\beta}$.

These problems cannot be solved easily. However, a bootstrap procedure can be developed to assess the precision of the model parameter estimates. This procedure is described in the next section.

As noted by the authors, simulation studies showed problems with `intcox` for high attrition rates ($> 30\%$). Since the ROM dataset has a very high attrition rate (60%), it is likely that problems will occur during the bootstrap analysis of this dataset. Therefore, the bootstrap procedure was only applied to the ROM dataset to test the bootstrap procedure. The results should not be interpreted and are therefore not shown. Although the bootstrap procedure cannot be applied to the ROM dataset, it can be applied to other datasets which include interval censored observations, but contain a maximum of 30% right censored subjects.

4.4 Bootstrapping

In this section a short introduction to bootstrapping is given and the `R` procedure to perform a bootstrap to estimate the precision of regression coefficients in the case of interval censored data is provided.

4.4.1 Short Introduction to Bootstrapping

Bootstrapping procedures can be used for statistical inference on model estimates $\hat{\theta}$. The idea is to draw many samples from a dataset and to estimate the model parameters for each sample. All estimates together give an impression of the distribution of $\hat{\theta}$. For example, the confidence interval of the estimator can be estimated [8].

Let $x_1, \ldots, x_n$ be n observations in the dataset under study. A *bootstrap sample* $\boldsymbol{x}^*$ is a random sample of size n drawn *with replacement* from the population of observations $x_1, \ldots, x_n$. For each of the B bootstrap samples, the *bootstrap replication* of parameter $\hat{\theta}$ is calculated as

$$\hat{\theta}^* = s(\boldsymbol{x}^*), \tag{4.4.1}$$

where $\boldsymbol{x}^* = (x_1^*, \ldots, x_n^*)$ is the bootstrap sample and $s(\cdot)$ is a function to estimate the parameter of interest. Hence, in the bootstrap procedure, B bootstrap replications are estimated. These replications together given an impression of the distribution of $\hat{\theta}$.

The empirical confidence interval of $\hat{\theta}$ can be derived from the bootstrap replications. For this, the definition of the cumulative distribution function is used.

Let $\hat{G}$ be the cumulative distribution function of $\hat{\theta}^*$. Then the $1-\alpha$ percentile interval is defined by the $\alpha/2$ and $1-\alpha/2$ percentiles of $\hat{G}$,

$$[\hat{\theta}_{lo}, \hat{\theta}_{up}] = [\hat{G}^{-1}(\alpha/2), \hat{G}^{-1}(1-\alpha/2)], \tag{4.4.2}$$

where $\hat{\theta}_{lo}$ and $\hat{\theta}_{up}$ represent the upper and lower bounds of the confidence interval of $\hat{\theta}$. By definition, $\hat{G}^{-1}(\alpha/2) = \hat{\theta}^{*(\alpha/2)}$ is the $100 \cdot \alpha/2$th percentile of the bootstrap distribution, yielding

$$[\hat{\theta}_{lo}, \hat{\theta}_{up}] = [\hat{\theta}^{*(\alpha/2)}, \hat{\theta}^{*(1-\alpha/2)}]. \tag{4.4.3}$$

The $100 \cdot \alpha/2$th empirical percentile $\hat{\theta}_B^{*(\alpha/2)}$ is the $B \cdot \alpha/2$th value of the ordered list of the B replications of $\hat{\theta}^*$. Likewise, the $B \cdot (1-\alpha/2)$th value of the ordered list of the B replications of $\hat{\theta}^*$ is the $100 \cdot (1-\alpha/2)$th empirical percentile $\hat{\theta}_B^{*(1-\alpha/2)}$ of the $\hat{\theta}^*$s. These percentiles are the empirical lower and upper limit of the confidence interval. If $B \cdot \alpha/2$ is not an integer, use the following procedure. Let $k = \lfloor (B+1)(\alpha/2) \rfloor$. Then the empirical $\alpha/2$ and $1-\alpha/2$ quantiles are respectively defined by the kth largest and the $(B+1-k)$th largest values of the $\hat{\theta}^*$s.

This estimated confidence interval of a model parameter is called the *percentile interval*.

The algorithm can be summarized as:

Bootstrap algorithm for estimating confidence intervals for model parameters

Step 1: Select B independent bootstrap samples $x^{*1}, x^{*2}, \ldots, x^{*B}$ of size n with replacement from the n observations.

Step 2: Estimate the parameter of interest for each bootstrap sample

$$\hat{\theta}^*(b) = s(\boldsymbol{x}^{*b}), \qquad b = 1, 2, \ldots, B.$$

Step 3: Estimate the confidence interval by taking the empirical percentiles from the B replications,

$$[\hat{\theta}_{lo}, \hat{\theta}_{up}] = [\hat{\theta}_B^{*(\alpha/2)}, \hat{\theta}_B^{*(1-\alpha/2)}].$$

4.4.2 Bootstrap for Interval Censoring

The bootstrap algorithm can be used to bootstrap the 95% confidence intervals for the model parameters for the Cox proportional hazards model on interval censored data. The `R` code corresponding with this algorithm has been described by Henschel et al. [12] and was slightly modified such that it can be used for any dataset and model. Two functions are written to perform the bootstrap procedure, `intcox.boot` and `intcox.bootstrapping`. The former estimates the parameters of interest for one bootstrap sample, and the latter repeats this process and calculates the empirical confidence intervals.

In this section only part of the R code is shown. The complete procedure can be found in Appendix A.

Function `intcox.boot`

The function `intcox.boot` draws a bootstrap sample from the data and uses `intcox` to estimate the regression coefficients for the model based on that sample, i.e. it performs step 1 and 2 of the bootstrap algorithm. The input for the function are the Cox proportional hazard model, specified in `formula`, the data containing the values of the variables, `data`, and an integer `i`. This integer is not used in the function itself, but it simplifies repetition of the function, as will be seen in the description of `intcox.bootstrapping`.

First, a bootstrap sample of size n is drawn with replacement from the subjects in the dataset.

```
boot.sample <- sample(1:nrow(data), size = nrow(data),
                      replace = T)
data.sample <- data[boot.sample,]
```

Then `intcox` is applied on the data sample to estimate regression coefficients for the model specified in `formula`.

```
boot.fit <- intcox(formula, data = data.sample, no.warnings = T)
```

The function output are the estimated regression coefficients and a termination indicator, indicating whether the algorithm performed by `intcox` converged.

```
return(list(coef = coefficients(boot.fit),
            term = boot.fit$termination
            )
       )
```

The complete function `intcox.bootstrap` can be found in Appendix A.1.

Function `intcox.bootstrapping`

`intcox.bootstrapping` is used to iterate the sampling of bootstrap samples and estimation of regression coefficients. Furthermore, it calculates the empirical confidence intervals of the regression coefficients and the corresponding hazard ratios.

The input for `intcox.bootstrapping` is the Cox proportional hazards model of interest, `formula`, the dataset, `data`, the number of iterations, `n.iter`, and the significance level `alpha`. `n.iter` has a default value of 1000 iterations, and `alpha` has a default value of 0.05.

The function `intcox.bootstrapping` estimates the regression coefficients on `n.iter` bootstrap samples by calling `intcox.boot` `n.iter` times. Regression coefficients and termination indicators are stored in matrix `boot`.

```
boot <- lapply(1:n.iter, intcox.boot, formula = formula,
               data = data)
cov.names <- names(boot[[1]]$coef)
boot <- matrix(unlist(boot), byrow = T, nrow = n.iter)
colnames(boot) <- c(cov.names, "termination")
```

For each covariate in the model, `n.iter` regression coefficient estimates are found. This results in the matrix `boot`, which has the following structure:

iter	cov_1	...	cov_p	term
1	$\hat{\theta}_1(1)$	...	$\hat{\theta}_p(1)$	term_1
2	$\hat{\theta}_1(2)$	...	$\hat{\theta}_p(2)$	term_2
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
B	$\hat{\theta}_1(B)$	...	$\hat{\theta}_p(B)$	term_B

The empirical percentiles are used to find the percentile interval. The empirical percentiles can be found in the ordered list of estimates at positions k and $(B + 1 - k)$, with $k = \lfloor (B + 1)(\alpha/2) \rfloor$.

```
k <- floor((n.iter + 1)*(alpha/2))
pos.lower <- k
pos.upper <- n.iter + 1 - k
```

These percentiles are calculated for each of the regression coefficients and are returned as the lower and upper bounds of the CI of the coefficients in `CI.coefficients`. Output `CI.HR` are the confidence interval boundaries of the corresponding hazard ratios.

```
for(i in 1:length(cov.names))
{
  cov <- cov.names[i]
  cov.ord <- order(boot[,cov])

  CICoef$CICoefLower[i] <- boot[cov.ord, cov][pos.lower]
  CICoef$CICoefUpper[i] <- boot[cov.ord, cov][pos.upper]
  CIHR$CIHRLower[i] <- exp(CICoef$CICoefLower[i])
  CIHR$CIHRUpper[i] <- exp(CICoef$CICoefUpper[i])
}

# Return results.
return(list(CI.coefficients = CICoef,
            CI.HR = CIHR,
            terminationtypes = terminationtypes))
```

The complete function can be found in Appendix A.2.

It is important to note that this is an application of the bootstrap in a non-standard situation, so the consistency is not guaranteed. It is beyond the scope of this thesis to further investigate this here.

Chapter 5

Dependent Censoring with Inverse Probability of Censoring Weighted (IPCW) Estimator

In the ROM dataset, the attrition rate was high, i.e. many patients were lost to follow-up during the study. It is possible that subjects lost to follow-up are not representative for the patients at risk. They may have patient characteristics that differ from the average patient characteristics (*dependent* or *non-random censoring*). If patients no longer come to their appointments because they feel better, there is a trend that subjects with a low severity of disease are lost to follow-up and only the severe patients remain at risk. Clinicians believe that patients' withdraw in the ROM dataset is likely to be related to their health status. The covariate that best determines the health status (severity of anxiety disorder) is also one of the covariates used in the definition of the event time. Therefore, this expectation can be interpreted as either dependent or informative censoring. For this thesis it was decided to choose the dependent censoring interpretation. In the dependent censoring framework, all covariates can be incorporated in the censoring model. In this way all covariates that have an influence on the time to censoring can be found (i.e. the focus is not on the BAS measurement only).

The survival techniques described in chapter 2 and chapter 4 assume independent and random censoring, and may under- or overestimate the survival time in case of dependent censoring. Survival analysis techniques have been extended such that more types of data can be analysed. Although many methods have been developed to analyse data in the presence of informative censoring [4, 14, 32], for dependent censoring, only the *Inverse Probability of Censoring Weighted (IPCW)* estimator has been developed. This method corrects for dependent censoring in right censored data by giving extra weight to subjects who are not censored. With these weights, the survival function in the absence of censoring can be estimated.

Inverse probability weighting (IPW) is more often used in analyses on datasets including missing data. A solution to missing data is complete case (CC) analysis. However, this gives biased results if the excluded individuals are different from those included.

When complete cases are weighted by their inverse probability of being a complete case (IPW), this bias may be reduced.

Another application of IPW is in survey analysis. If a survey sample is representative of the entire population, only few individuals with rare characteristics will be included. This can be problematic if researchers are interested in the rare cases. By increasing the sampling probability for these individuals, many of them will be included in the sample. However, the survey sample is no longer representative of the entire population. Therefore, if one wants to estimate population characteristics, the sampled subjects have to be weighted according to their inverse sampling probability.

A detailed review of IPW for dealing with missing data can be found in Seaman and White [25].

This chapter is organized as follows: in section 5.1 an introduction to IPCW will be given. Then the algorithm for this method is described. In chapter 6 a procedure proposed in this thesis to apply IPCW method by using the standard survival package in R will be described.

5.1 Basic Principle

IPCW estimators correct for censored subjects by giving extra weight to subjects who are not censored. Each subject j is weighted by the inverse of an estimate of the conditional probability of having remained uncensored until time t. This conditional probability can be estimated with the Product-Limit estimator for time to censoring or by fitting the Cox model on the data. The weight for subject j at time t is denoted as $\hat{W}_j(t)$.

To illustrate how IPCW works an example will be given. Let S be a sample of 4 individuals and suppose that the estimated chance of remaining uncensored until time $t = 5$ is $\frac{1}{4}$, hence 3 out of each 4 subjects are censored before time $t = 5$. For each subject at risk, there would have been 3 extra subjects at risk for the event of interest at time $t = 5$ if censoring was absent. By weighting the contribution of subjects that are not censored at time $t = 5$ by $1/\frac{1}{4} = 4$, the censored subjects are included in the computations to estimate the quantity of interest. Since the conditional probabilities of being censored increases with time, the weights increase with time as well.

The IPCW method can be described as follows.

IPCW to correct for dependent censoring

Step 1: Model the censoring mechanism.

Step 2: Estimate the Product-Limit estimator and Cox proportional hazards estimator for time to censoring for each subject j at each timepoint t, $\hat{K}_j^0(t)$ and $\hat{K}_j^{\boldsymbol{Z}}(t)$.

Step 3: Calculate the unstabilized and/or stabilized IPCW weights for each of the subjects j, i.e. $W_j^{unstab}(t) = 1/\hat{K}_j^{\boldsymbol{Z}}(t)$ and $W_j^{stab}(t) = \hat{K}_j^0(t)/\hat{K}_j^{\boldsymbol{Z}}(t)$.

Step 4: Estimate the survival and/or Cox model for time to event in the absence of censoring with the IPCW weights, S_{IPCW}.

These steps will be explained in detail in the remaining of this chapter.

5.2 Step 1 and 2: Modelling the Censoring Model and Estimating the Probability of being Censored

When modelling the censoring mechanism, one wants to assess how long a subject stays in a study without being lost to follow-up, implying that a model is required for the time to censoring. This is a standard survival analysis with the moment of censoring as the event of interest. Hence, the time to censoring can be modelled in the same way as time to any event of interest. The theory described in chapter 2 can therefore be applied in this context as well.

5.2.1 Data Representation

In the censoring model, the event of interest is censoring, hence the subjects who are lost to follow-up have an 'event'. Subjects that are not censored, i.e. those who experience the original event of interest, are now considered 'censored' since their censoring time is not observed. Normally, the data representation of subject j for the time to event model is the triple $(T_j, \delta_j, \boldsymbol{Z}_j(t))$ (see section 2.5).

To represent the data as time to censoring data, only the indicator δ has to be inverted ($\delta_{C_j} = 1 - \delta_j$), since the definitions of censoring and events are inverted. Using the new data representation $(T_j, \delta_{C_j}, \boldsymbol{Z}_j(t))$, the methods described in chapter 2 can be applied to the data to estimate the probability of being censored.

5.2.2 Estimation of the Survival Function for Censoring

To assess which covariates have an influence on the probability of being censored, the subjects' covariates have to be included in the model. The probability of being censored given covariates $\boldsymbol{Z}_j(t)$ can be estimated in a non-parametric or parametric manner.

Non-Parametric Approach

The Cox proportional hazards model (see section 2.5) for censoring time is as follows.

$$h_C(t|\boldsymbol{Z}(t)) = h_{C_0}(t)\exp(\boldsymbol{\beta}_C^t \boldsymbol{Z}(t)), \tag{5.2.1}$$

where h_{C_0} is the baseline hazard of censoring, $\boldsymbol{\beta}_C$ is the vector of model parameters, and $\boldsymbol{Z}(t)$ represents the covariates. The hazard $h_C(t|\boldsymbol{Z})$ indicates the estimated probability

for a patient with characteristics $\boldsymbol{Z}(t)$ being censored in the next instant of time, if this subject was not censored until time t.

A Product-Limit estimator for time to censoring that includes covariates for each subject j, is derived as

$$\hat{K}_j^{\boldsymbol{Z}}(t) = \prod_{\{i;t_i<t,\delta_i=0\}} [1 - \hat{h}_{C_0}(t_i)\exp(\hat{\boldsymbol{\beta}}_C^t \boldsymbol{Z}_j(t_i))], \tag{5.2.2}$$

where the t_i's are the observation times, and $\hat{\boldsymbol{\beta}}_C$ is the estimated vector of parameters in (5.2.1). The baseline hazard for censoring at observation time t_i is estimated by

$$\hat{h}_{C_0}(t_i) = \frac{1-\delta_i}{\sum_{k=1}^n \exp(\hat{\boldsymbol{\beta}}_C^t \boldsymbol{Z}_k(t_i))R_k(t_i)}, \tag{5.2.3}$$

where $R_k(t_i)$ indicates whether subject k is still at risk at time t_i, i.e. $R_k(t_i) = I(X_k \geq t_i)$.

The estimator is denoted as $\hat{K}_j^{\boldsymbol{Z}}(t)$ to emphasize its dependence on $\boldsymbol{Z}_j(t)$. Let $\hat{K}_j^0(t)$ denote the traditional Product-Limit estimator for the probability of being uncensored at time t independent of the covariates $\boldsymbol{Z}_j(t)$. Then $\hat{K}_j^0(t)$ is equal to $\hat{K}_j^{\boldsymbol{Z}}(t)$ if $\boldsymbol{\beta}_C$ is the zero vector.

Parametric Approach

If a parametric approach is applied, a parametric model for time to censoring is assumed. In case a Weibull model is assumed, the baseline hazard for censoring is defined as

$$h_{C_0}(t) = \lambda\eta t^{\eta-1}, \tag{5.2.4}$$

and the survival function for the time to censoring for subject j can be estimated by

$$\hat{S}_{C_j}(t) = \exp\left(-\exp(\hat{\boldsymbol{\beta}}_W^t \boldsymbol{Z}_j(t))\hat{\lambda}t^{\hat{\eta}}\right), \tag{5.2.5}$$

where $\hat{\beta}_W$ is obtained by using the Weibull proportional hazards model.

This thesis focusses on the non-parametric approach.

5.2.3 Models for the Censoring Mechanism

When modelling the censoring mechanism, one has to decide which covariates to include in the model. Three types of models for time to censoring have been proposed [**?**]:

Model 1: Include only time-dependent covariates that are significant for both time to failure and time to censoring.

Model 2: Include all baseline covariates that are significant for time to failure, together with time-dependent variables.

Model 3: Include all baseline covariates that are significant for time to censoring, together with time-dependent variables.

All three models can be considered when modelling the censoring process.

5.3 Step 3: Calculating the IPCW Weights

The weights for each subject j are computed as $W_j(t) = 1/\hat{K}_j^{\boldsymbol{Z}}(t)$. Hence, they are the inverse of an estimate of the conditional probability of having remained uncensored until time t ($\hat{K}_j^{\boldsymbol{Z}}(t)$). These weights can become very large in case of heavy censoring. To deal with this problem, Robins [21] proposed another type of weight, $W_j(t) = \hat{K}_j^0(t)/\hat{K}_j^{\boldsymbol{Z}}(t)$. This weight will be close to 1 if there is no informative censoring. The weights are respectively called the *unstabilized* and *stabilized* weights, i.e. $W_j^{unstab}(t) = 1/\hat{K}_j^{\boldsymbol{Z}}(t)$ and $W_j^{stab}(t) = \hat{K}_j^0(t)/\hat{K}_j^{\boldsymbol{Z}}(t)$. Stabilized weights stay small and will be close to one for all t if $\boldsymbol{Z}(t)$ does not influence the probability of censoring. The unstabilized weights, which are more intuitive, may be appropriate when there is only light to moderate censoring.

5.4 Step 4: Estimating Time to Event with IPCW Weights

The Product-Limit estimator is given by

$$\hat{S}(t) = \begin{cases} 1 & \text{if } t < t_1, \\ \prod_{t_i \leq t} \left[1 - \frac{d_i}{Y_i}\right] & \text{if } t \geq t_1 \end{cases}, \tag{5.4.1}$$

where d_i is the number of subjects that experience the event at time t_i and Y_i is the number of subjects at risk at t_i (see section 2.4.1). If subjects are weighted by the inverse probability of remaining uncensored until time t, the Product-Limit estimator for distinct-event time data can be written as follows:

$$\hat{S}_{IPCW}(t) = \begin{cases} 1 & \text{if } t < t_1, \\ \prod_{\{i;t_i<t\}} \left[1 - \frac{\delta_i \hat{W}_i(t_i)}{\sum_{k=1}^n R_k(t_i)\hat{W}_k(t_i)}\right] & \text{if } t \geq t_1 \end{cases}. \tag{5.4.2}$$

Here, $\delta_i \hat{W}_i(t_i)$ represents a subject who experiences an event at time t_i, with his or her contribution weighted by $\hat{W}_i(t_i)$. The sum $\sum_{k=1}^n R_k(t_i)\hat{W}_k(t_i)$ represents the weighted contribution of subjects at risk at time t_i. For the Product-Limit estimator, either unstabilized ($1/\hat{K}_j^{\boldsymbol{Z}}(t)$) or stabilized weights ($\hat{K}_j^0(t)/\hat{K}_j^{\boldsymbol{Z}}(t)$) can be used, since $\hat{K}^0(t)$ cancels out in both the numerator and denominator.

In some studies, subjects experience the event of interest at the same time points. The Product-Limit estimator (5.4.2) only holds for datasets without ties. Estimator (5.4.2) can be generalized to situations when ties are present. In the presence of ties, all subjects j who experience the event at t_i are individually weighted by their weights $\hat{W}_j(t_i)$ and their contribution is summed to estimate the number of subjects that would

have experienced the event at t_i in the absence of censoring. Hence, for this situation the survival probability can be estimated as

$$\hat{S}_{IPCW}(t) = \begin{cases} 1 & \text{if } t < t_1, \\ \prod_{t_i \leq t} \left[1 - \frac{\sum_{\{j;\delta_j(t_i)=1\}} \hat{W}_j(t_i)}{\sum_{k=1}^{n} R_k(t_i)\hat{W}_k(t_i)} \right] & \text{if } t \geq t_1 \end{cases} . \tag{5.4.3}$$

As before, the chosen type of weights does not influence the results of the Product-Limit estimator.

Estimator (5.4.3) can be implemented in `R` as follows:

```
survIPCW <- function(Tstart, Tstop, status, weights)
{
  ord <- order(Tstop)
  Tstart <- Tstart[ord]
  Tstop <- Tstop[ord]
  status <- status[ord]
  weights <- weights[ord]

  tout <- Tstop[status==1]
  tout <- unique(tout)

  nout <- length(tout)

  di <- Yi <- rep(NA, nout)
  for (i in 1:nout)
  {
    di[i] <- sum((Tstop==tout[i] & status==1)*weights)

    Yi[i] <- sum((Tstop >= tout[i] & Tstart < tout[i])*weights)
  }

  surv <- cumprod(1 - di/Yi)

  return(data.frame(time=tout, surv=surv))
}
```

The functions `survfit` (and `coxph`) in the `survival` package in `R` can handle weighted subjects. Therefore, the `R` function above should give the same result as `survfit`. This was tested with a test dataset.

5.5 Assumptions concerning IPCW

The IPCW method relies on the following assumptions [21, 23]:

1. There are no unmeasured cofounders for censoring;
2. If the hazard of censoring is conditioned on the recorded history $\bar{\boldsymbol{Z}}(t)$, it does not further depend on X (sequential ignorability of censoring);
3. The data is coarsened at random (CAR), i.e. the censoring mechanism does not depend on the outcome, but may depend on the covariates.

If all these assumptions are satisfied and all prognostic factors are recorded in $\boldsymbol{Z}(t)$, IPCW estimators will correct the bias due to dependent censoring completely.

Chapter 6

Inverse Probability Censoring Weighted Estimator in R by using the Survival Package

In chapter 5 the classical IPCW estimator was introduced. An extension of the classical theory and a way to implement the survival function in this context in R were provided in this thesis, since there is no R library available.

In this chapter an alternative method to implement S_{IPCW} will be illustrated. It is one of the novelties in this thesis. The idea is based on a particular way of preparing the data in such a way that the function `survfit` from the library `survival` can be used and implementation of mathematical details can be avoided.

6.1 Application of IPCW in R

The algorithm can be summarized in the following steps:

Preparation: Transform the data into a long data format.

Step 1: Model the censoring mechanism.

Step 2: Estimate the Product-Limit estimator and Cox proportional hazards estimator for time to censoring for each subject j at each timepoint, $\hat{K}_j^0(t)$ and $\hat{K}_j^{\boldsymbol{Z}}(t)$.

Step 3: Calculate the unstabilized and/or stabilized IPCW weights for each of the subjects j, i.e. $W_j^{unstab}(t) = 1/\hat{K}_j^{\boldsymbol{Z}}(t)$ and $W_j^{stab}(t) = \hat{K}_j^0(t)/\hat{K}_j^{\boldsymbol{Z}}(t)$.

Step 4: Estimate the survival and/or Cox model for time to event in the absence of censoring with the IPCW weights, S_{IPCW}.

To implement steps 1 - 4 in R careful bookkeeping is required. The method proposed in this thesis will be illustrated by means of a test dataset presented in Table 6.1. Since the test dataset is small, the calculations can easily be done by hand to check the R code. In this dataset, *id* represents the subject's identification, *time* is time to event or censoring, *status* is the event indicator, and *cov* the measurements of a covariate.

id	time	status	cov
1	18	0	1
2	23	1	2
3	27	0	1
4	32	1	2
5	57	0	3
6	64	1	2

Table 6.1: Test data used to illustrate the application of the IPCW estimator in R by using the survfit package.

6.1.1 Preparation: Transform the Data into a Long Data Format

Many survival studies have their data stored initially in a one-row-per-subject (wide) format. In this section the wide format will be transformed in a long format. This format will then be used to estimate the function of interest, S_{IPCW}.

For each individual in the data set the interval from time origin until time to event or censoring is divided in subintervals. The boundaries of these intervals are the event and censoring times of all subjects in the dataset. Each individual j needs r_j number of rows, one for each time interval in which a specific subject is still at risk. For time fixed covariates, the value of the covariates is repeated in each r_j rows. The status value takes value 0 in each of these subintervals until the last one, where it changes into 1 if an event has occurred. In Table 6.2 the long data format of the test data is illustrated. This data format can be interpreted as the counting process style, and a Cox's regression model can be fitted on this type of data (Andersen and Gill [2]).

Transforming the dataset from wide to long format can be done by employing `survSplit` from the library `survival`. This function splits each time interval in subintervals with boundaries at `Tstart` and `Tstop`.

Note that it is wrong to define the *censoring status* as $\delta_C = 1 - \delta$, where δ is the indicator for the event of interest. For example, in Table 6.2 subject 3 in the subinterval $[18, 23]$ is still at risk. Implementing the correct censoring status can be done by applying the `survSplit` function two times, as shown below.

```
test$censored <- 1-test$status
```

id	Tstart	Tstop	status	censored	cov
1	0	18	0	1	1
2	0	18	0	0	2
2	18	23	1	0	2
3	0	18	0	0	1
3	18	23	0	0	1
3	23	27	0	1	1
4	0	18	0	0	2
4	18	23	0	0	2
4	23	27	0	0	2
4	27	32	1	0	2
5	0	18	0	0	3
5	18	23	0	0	3
5	23	27	0	0	3
5	27	32	0	0	3
5	32	57	0	1	3
6	0	18	0	0	2
6	18	23	0	0	2
6	23	27	0	0	2
6	27	32	0	0	2
6	32	57	0	0	2
6	57	64	1	0	2

Table 6.2: Test data in long format. This test data is used to illustrate the application of the IPCW estimator in R by using the survfit package.

```
# Split data over all event and censoring times
test.long <- survSplit(test, cut=times, end="time",
                       start="Tstart", event="status",
                       id = "id")
# Order the data first by id, then by time interval
test.long <- test.long[order(test.long$id,test.long$time),]

# Repeat for censoring times
test.long.cens <- survSplit(test, cut=times, end="time",
                            start="Tstart", event="censored",
                            id = "id")
test.long.cens <- test.long.cens[order(test.long.cens$id,
                                       test.long.cens$time),]

test.long$censored <- test.long.cens$censored
```

6.1.2 Step 1 and 2: Fit the Censoring Model and calculate $\hat{K}_j^0(t)$ and $\hat{K}_j^{\boldsymbol{Z}}(t)$

In order to find the required weights necessary to implement the IPCW method, censoring probabilities should be calculated for each subject j at each time point t, for both censoring models with and without covariates, $\hat{K}_j^{\boldsymbol{Z}}(t)$ and $\hat{K}_j^0(t)$. Those weights can be estimated by using the functions `coxph` and `survfit` from the `survival` package.

Let C_0 and C_Z denote the censoring models without and with covariates respectively. Since C_0 does not depent on the covariates, the survival estimates are the same for all subjects at each time point. Note that these models are fitted on the long format data. If no time dependent covariates are included in the model, C_0 and $C_{\boldsymbol{Z}}$ can be fitted on the short format data.

```
C0 <- coxph(Surv(Tstart, Tstop, censored) ~ 1,   data = test.long)
CZ <- coxph(Surv(Tstart, Tstop, censored) ~ cov, data = test.long)
```

In the model C_0, the censoring probabilities are equal for each subjects. In this case, the estimators $\hat{K}_j^0(t)$ and $\hat{K}_j^{\boldsymbol{Z}}(t)$ can be simply estimated as follows.

```
C0fit <- summary(survfit(C0), times = test.long$Tstart)
test.long$K0ti <- C0fit$surv
```

For the model with (time-dependent) covariates, $C_{\boldsymbol{Z}}$, the estimator has to be computed on each seperate row since covariates can change at any time point. This can be done as follows:

```
test.long$KZti <- NULL

for(i in 1:nrow(test.long))
{
  datai <- test.long[i,]

  sfiCZ <- survfit(CZ, newdata = datai)
  ssfiCZ <- summary(sfiCZ, times = datai$Tstart)
  test.long$KZti[i] <- ssfiCZ$surv
}
```

In case only baseline covariates are included in the model, it is enough to call the `survfit` function only one time per subject, since the baseline covariates do not chance over time. The survival estimates are calculated for each time interval in which the individual is still in the study.

6.1.3 Step 3: Compute IPCW Weights

The next step is to calculate the IPCW weights from the estimated probabilities of being censored. Both stabilized and unstabilized weights need to be computed (see section 5.3).

```
test.long$WUnStab <- 1/test.long$KZti
test.long$WStab <- test.long$K0ti/test.long$KZti
```

To the dataset in long format shown in Table 6.2, columns corresponding to the specific weight for each interval are added. The final result is illustrated in Table 6.3. Pen and paper calculation were performed by using formulas given in section 5.4 to check results obtained in R.

id	Tstart	Tstop	status	censored	cov	K0ti	KZti	WUnStab	WStab
1	0	18	0	1	1	1.00	1.00	1.00	1.00
2	0	18	0	0	2	1.00	1.00	1.00	1.00
2	18	23	1	0	2	0.85	0.91	1.10	0.93
3	0	18	0	0	1	1.00	1.00	1.00	1.00
3	18	23	0	0	1	0.85	0.71	1.41	1.19
3	23	27	0	1	1	0.85	0.71	1.41	1.19
4	0	18	0	0	2	1.00	1.00	1.00	1.00
4	18	23	0	0	2	0.85	0.91	1.10	0.93
4	23	27	0	0	2	0.85	0.91	1.10	0.93
4	27	32	1	0	2	0.66	0.77	1.31	0.86
5	0	18	0	0	3	1.00	1.00	1.00	1.00
5	18	23	0	0	3	0.85	0.97	1.03	0.87
5	23	27	0	0	3	0.85	0.97	1.03	0.87
5	27	32	0	0	3	0.66	0.93	1.08	0.71
5	32	57	0	1	3	0.66	0.93	1.08	0.71
6	0	18	0	0	2	1.00	1.00	1.00	1.00
6	18	23	0	0	2	0.85	0.91	1.10	0.93
6	23	27	0	0	2	0.85	0.91	1.10	0.93
6	27	32	0	0	2	0.66	0.77	1.31	0.86
6	32	57	0	0	2	0.66	0.77	1.31	0.86
6	57	64	1	0	2	0.40	0.35	2.85	1.14

Table 6.3: Test data in long format, including the estimated $\hat{K}_j^0(t)$'s, $\hat{K}_j^{\mathbf{Z}}(t)$'s and stabilized and unstabilized weights.

6.1.4 Step 4: Estimate the Survival Model for Time to Event in the Absence of Censoring with IPCW Weights

Once the IPCW weights are calculated, the survival model for time to event in the absence of censoring can be estimated with (5.4.3). The functions `survfit` and `coxph` take the argument `weights`, which enables one to give weights to the subjects in the dataset. Therefore, these functions can be used to weigh the contributions of the subjects, as in (5.4.3). The `R` function `survfit` can be used to plot the survival curves for time to event, estimated with and without the IPCW.

Since both types of weights result in the same Product-Limit estimator, the survival plots will give the same result, therefore only one of these plots is needed. Plotting the results will visualize the estimated survival curve PL_{IPCW} that corrects for dependent censoring.

```
# Survival without IPCW weights:
res <- survfit(Surv(Tstart, Tstop, status) ~ 1,
               data = test.long)
res$surv

# survival with IPCW stabilized weights:
resWStab <- survfit(Surv(Tstart, Tstop, status) ~ 1,
                    data = test.long, weights = WStab)
resWStab$surv

# survival with IPCW unstabilized weights:
resWUnStab <- survfit(Surv(Tstart, Tstop, status) ~ 1,
                      data = test.long, weights = WUnStab)
resWUnStab$surv
```

To incorporate covariates in the analysis, a Cox model with option `weights` can be fitted. In this way the IPCW weights will be used in the model for time to event. The resulting regression coefficients can be compared to those found with the standard method to assess the influence of the IPCW method. For the Cox model, the two types of weights will give different survival estimates S_{IPCW}.

```
# Classical analysis without IPCW
CoxW0 <- survfit(coxph(Surv(Tstop, status) ~ cov,
                 data = test.long))
CoxW0$surv
```

```
# IPCW with stabilized weights
CoxWStab <- survfit(coxph(Surv(Tstart, Tstop, status) ~ cov,
                          data = test.long, weights = WStab))
CoxWStab$surv

# IPCW with unstabilized weights
CoxWUnStab <- survfit(coxph(Surv(Tstart, Tstop, status) ~ cov,
                            data = test.long, weights = WUnStab))
CoxWUnStab$surv
```

6.2 Application of IPCW to a Datasets With and Without Dependent Censoring

To show the effect of IPCW, two longer test datasets were created, one without and one with dependent censoring. These are given in wide format in Table 6.4. In the dataset in Table 6.4a, the covariate values are approximately evenly distributed over the subjects, regardless their status. In the dataset in Table 6.4b, nearly all subjects who experience the event have $cov = 1$ and nearly all subjects who are censored have $cov = 2$. This suggests that being censored is related to the covariate, i.e. dependent censoring is present in this dataset.

After applying the IPCW method as described in the previous subsections, the Product-Limit curves estimated with (PL_{IPCW}) and without (PL) the IPCW weights are plotted in Figure 6.1. For the test dataset without dependent censoring (see Figure 6.1a), the two estimated survival curves are nearly identical. This was expected, since there was no dependent censoring to correct for. Because of the presence of dependent censoring in the second test dataset, there is a difference between the two estimated survival curves (see Figure 6.1b). This shows that the IPCW weights do indeed detect dependent censoring and correct for it.

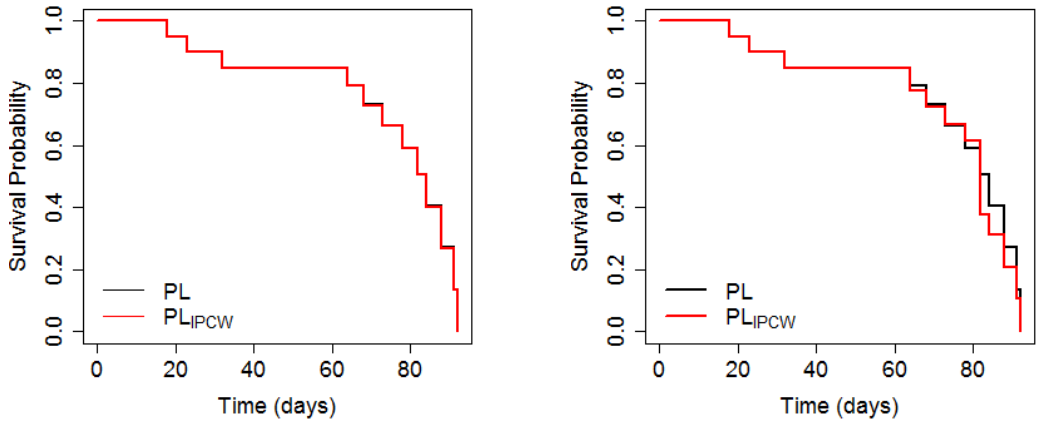
id	time	status	cov
1	18	1	1
2	23	1	2
3	27	0	1
4	32	1	2
5	57	0	1
6	64	1	1
7	65	0	2
8	68	1	3
9	70	0	2
10	73	1	1
11	75	0	1
12	78	1	2
13	80	0	2
14	82	1	2
15	83	0	1
16	84	1	2
17	86	0	3
18	88	1	1
19	91	1	1
20	92	1	1

(a) Test dataset without dependent censoring.

id	time	status	cov
1	18	1	1
2	23	1	1
3	27	0	2
4	32	1	1
5	57	0	2
6	64	1	3
7	65	0	2
8	68	1	1
9	70	0	2
10	73	1	1
11	75	0	2
12	78	1	1
13	80	0	2
14	82	1	3
15	83	0	2
16	84	1	1
17	86	0	2
18	88	1	1
19	91	1	1
20	92	1	1

(b) Test dataset with dependent censoring.

Table 6.4: Two test datasets.



(a) Estimated survival curves for the test dataset in Table 6.4a.

(b) Estimated survival curves for the test dataset in Table 6.4b.

Figure 6.1: Estimated survival cures with (red) and without (black) the IPCW weights for the two test datasets given in Table 6.4.

6.3 Application of IPCW to the ROM Dataset

In the ROM dataset, many subjects were lost to follow-up and clinicians believe that the dropout is likely to be related to the health status of the subjects. The estimated survival curve for time to censoring is shown in Figure 6.2. Subjects who did not experience an event after 730 days (2 years) were considered as chronic patients and removed from the study, i.e. censored at 730 days. This is shown in Figure 6.2 by the vertically decreasing line at 730 days.

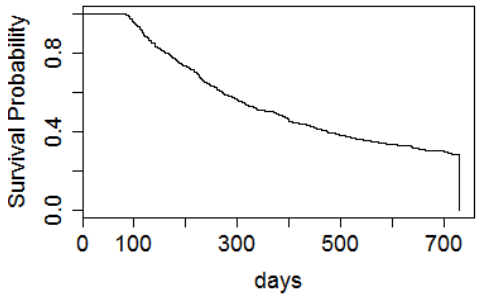


Figure 6.2: Product-Limit curve for the time to censoring in the ROM dataset.

As already mentioned in subsection 5.2.3, many different models for the censoring mechanism can be considered. In this thesis, only the non-parametric version of the IPCW method, whose implementation in `R` was shown in the latter section, will be applied. Since the IPCW method described in this thesis is not applicable to interval censored data, the midpoint approach is used to transfer the interval censored data into right censored data.

6.3.1 Model 1: Time Dependent Covariates

The first model for time to censoring includes only time dependent covariates BAS and BSI-12 measurements which were measured at each visit. As is shown in Table 6.5 these covariates have no significant effect on censoring time.

	HR	**95% CI**	**p-value**
BAS	1.031	$[0.942 - 1.129]$	0.506
BSI-12	0.993	$[0.904 - 1.091]$	0.883

Table 6.5: Multivariate hazard ratios for the time to censoring model with their 95% confidence interval and p-value.

The estimated survival curves with and without the IPCW method ($S(t)$ and $S_{IPCW}(t)$) are shown in Figure 6.3. The difference between the survival curves is very small.

Since the time dependent covariates do not have a significant effect on the censoring time and prolong the calculation time, it was decided to leave out these covariates in the next two models.

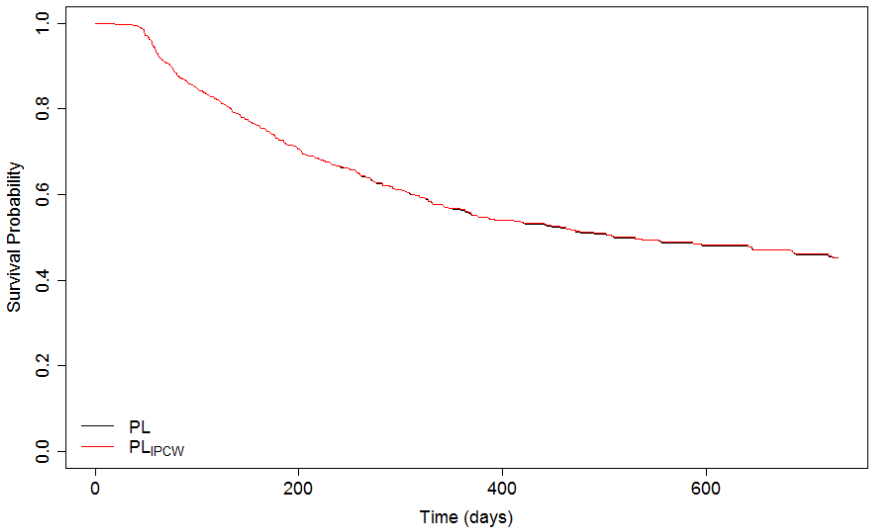


Figure 6.3: Product-Limit curves for the time to response (50% improvement on both BAS and BSI-12 scores). The estimated survival curve when using (un)stabilized weights and a Cox model with time-fixed covariates (in red), and the estimated survival curve without using IPCW weights (in black).

6.3.2 Model 2: Baseline Covariates with a Significant Influence on the Time to Failure

The second model for time to censoring includes only baseline covariates that have a significant influence on time to event (see Table 3.4 for the complete list of these covariates). Fitting the Cox model for time to censoring shows that none of these covariates are significant for time to censoring, see Table 6.6.

The fitted Cox model was used to implement the IPCW method. In Figure 6.4 the estimated survival curves for time to event with and without correcting for censoring by the IPCW method are shown. The difference between the two curves is extremely small, which suggests that for this data set, modelling the censoring mechanism does not give any additional information.

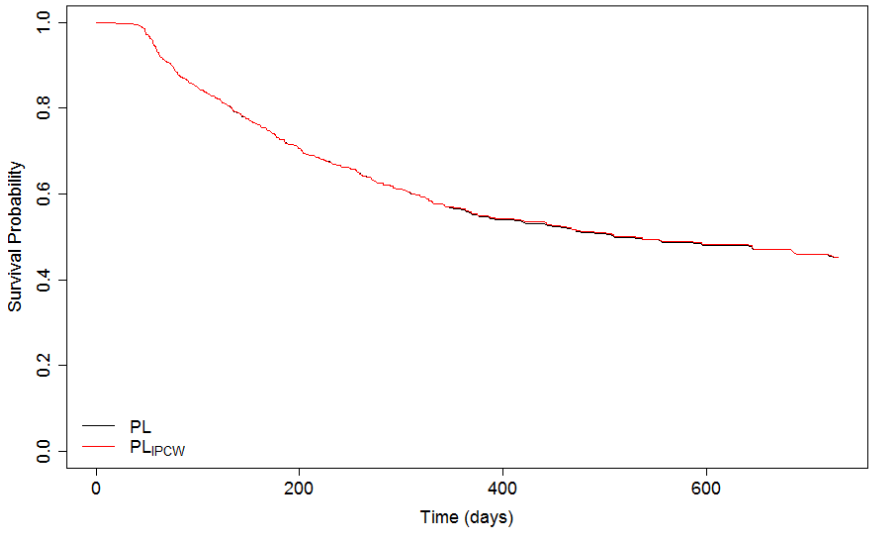


Figure 6.4: Product-Limit curves for the time to response (50% improvement on both BAS and BSI-12 scores). The estimated survival curve when using (un)stabilized weights and a Cox model with time-fixed covariates (in red), and the estimated survival curve without using IPCW weights (in black).

	HR	**95% CI**	**p-value**
Age	0.951	0.856 – 1.056	0.346
Gender			
- female	1		
- male	0.913	0.745 – 1.119	0.380
DSM-IV-TR agoraphobia			
- no AP	1		
- AP	1.045	0.748 – 1.461	0.796
DSM-IV-TR panic disorder with or without agoraphobia			
- no PD	1		
- PD	0.787	0.579 – 1.071	0.128
DSM-IV-TR social phobia			
- no SP	1		
- SP	0.879	0.664 – 1.164	0.368
DSM-IV-TR generalized anxiety disorder			
- no GAD	1		
- GAD	0.955	0.713 – 1.279	0.756
SF36 general health	0.967	0.877 – 1.066	0.501
Living situation			
- independently with partner and/or children	1		
- independently alone	1.013	0.819 – 1.254	0.904
- with family	1.180	0.884 – 1.575	0.261
Education			
- high	1		
- medium	0.983	0.762 – 1.268	0.892
- low	1.061	0.824 – 1.367	0.645
Occupation			
- daily occupation	1		
- no daily occupation	0.904	0.752 – 1.086	0.280
Ethnicity			
- Dutch	1		
- non-Dutch	0.987	0.795 – 1.224	0.903
Alcohol abuse or dependence			
- no alcohol abuse or dependence	1		
- alcohol abuse or dependence	0.897	0.610 – 1.319	0.581
BAS	0.993	0.902 – 1.093	0.880
DAPP-SF affective lability	0.939	0.849 – 1.039	0.222
DAPP-SF oppositionality	1.036	0.942 – 1.140	0.469
DAPP-SF conduct problems	1.049	0.957 – 1.149	0.311
Number of simultaneous DSM-IV-TR anxiety disorders			
- single anxiety disorder	1		
- multiple anxiety disorders	1.153	0.920 – 1.445	0.218

Table 6.6: Multivariate hazard ratios with their 95% confidence interval and p-value of the first censoring model with time fixed covariates.

6.3.3 Model 3: Baseline Covariates with a Significant Influence on Censoring Time

The third model includes baseline measurements of covariates that have a significant influence on time to censoring. These covariates have been found by applying the two step approach (univariate analysis and backward stepwise removal of covariates) which was used to determine the significant covariates for time to event. Age, gender and the four dichotomized main diagnostic categories (PD/A, AP, SP and GAD) have been included in the model since they were clinical relevant. The estimated regression coefficients are given in Table 6.7. In Table 6.7 only mental health at baseline appears to have a significant effect on the censoring mechanism.

	HR	**95% CI**	**p-value**
Age	0.936	$0.854 - 1.025$	0.152
Gender			
- female	1		
- male	0.944	$0.789 - 1.130$	0.530
DSM-IV-TR agoraphobia			
- no AP	1		
- AP	1.140	$0.869 - 1.497$	0.345
DSM-IV-TR panic disorder with or without agoraphobia			
- no PD	1		
- PD	0.858	$0.660 - 1.116$	0.254
DSM-IV-TR social phobia			
- no SP	1		
- SP	0.994	$0.785 - 1.259$	0.963
DSM-IV-TR generalized anxiety disorder			
- no GAD	1		
- GAD	1.027	$0.799 - 1.319$	0.835
Mental Health	0.900	$0.828 - 0.979$	0.014

Table 6.7: Multivariate hazard ratios with their 95% confidence interval and p-value corresponding to the second censoring model.

This model is also used to compute the IPCW weights. Resulting estimated survival curves are shown in Figure 6.5. The difference between the two curves is again extremely small.

Based on Figure 6.3 - Figure 6.5 it can be concluded that it does not matter which censoring model is used. All lead to the same estimated survival curve as for the standard analysis without IPCW weights.

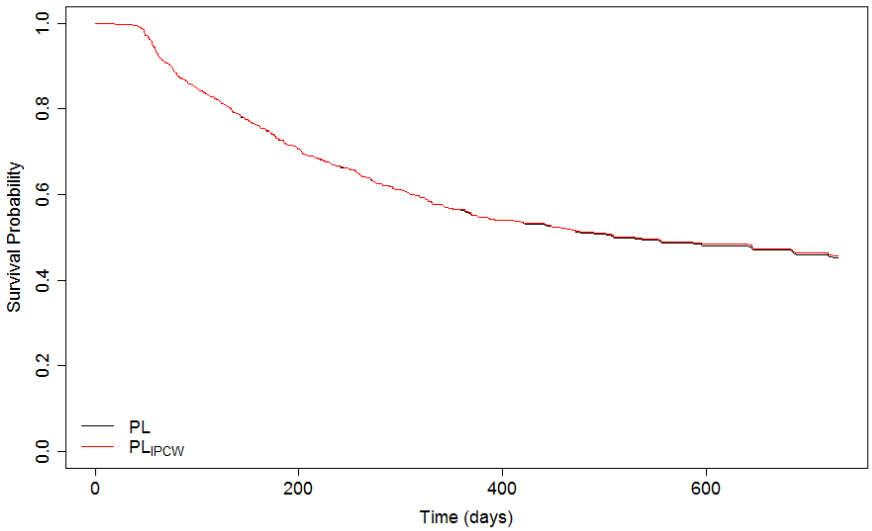


Figure 6.5: Product-Limit curves for the time to response (50% improvement on both BAS and BSI-12 scores). The estimated survival curve when using (un)stabilized weights and a Cox model with time-fixed covariates (in red), and the estimated survival curve without using IPCW weights (in black).

6.4 Comparing the Estimated Regression Coefficients

Each of the described models for IPCW can be used to fit a Cox model for time to event. All covariates included in the final model in section 3.4 are put in the Cox model for time to event which is fitted with IPCW weights. Since the models for time to censoring lead to different weights, the estimated regression coefficients in the Cox model may differ for each IPCW analysis. The estimated regression coefficients are shown in Figure 6.6 (unstabilized weights) and Figure 6.7 (stabilized weights). The estimated regression coefficients based on IPCW do not differ substantially from those estimated without the IPCW weights.

Although it appears that the effects of the IPCW method on the parameter estimates of the Cox model is very small, the effect might be large on the survival functions for specific individuals in the dataset. In Figure 6.8 the estimated survival curves under the Cox model without IPCW and the IPCW Cox models 2 and 3 are compared for the individual who has the largest value for

$$|\hat{\beta}_X^{IPCW} Z_j - \hat{\beta}_X^0 Z_j|, \tag{6.4.1}$$

where $\hat{\beta}_X^{IPCW}$ and $\hat{\beta}_X^0$ are the estimated regression coefficients for the Cox model with IPCW and without IPCW respectively. For model 2, the biggest value of $\hat{\beta}_X^{IPCW} Z_j - \hat{\beta}_X^0 Z_j$ is positive and for model 3 it is negative, therefore, the S_{IPCW} curve (PL_{IPCW})

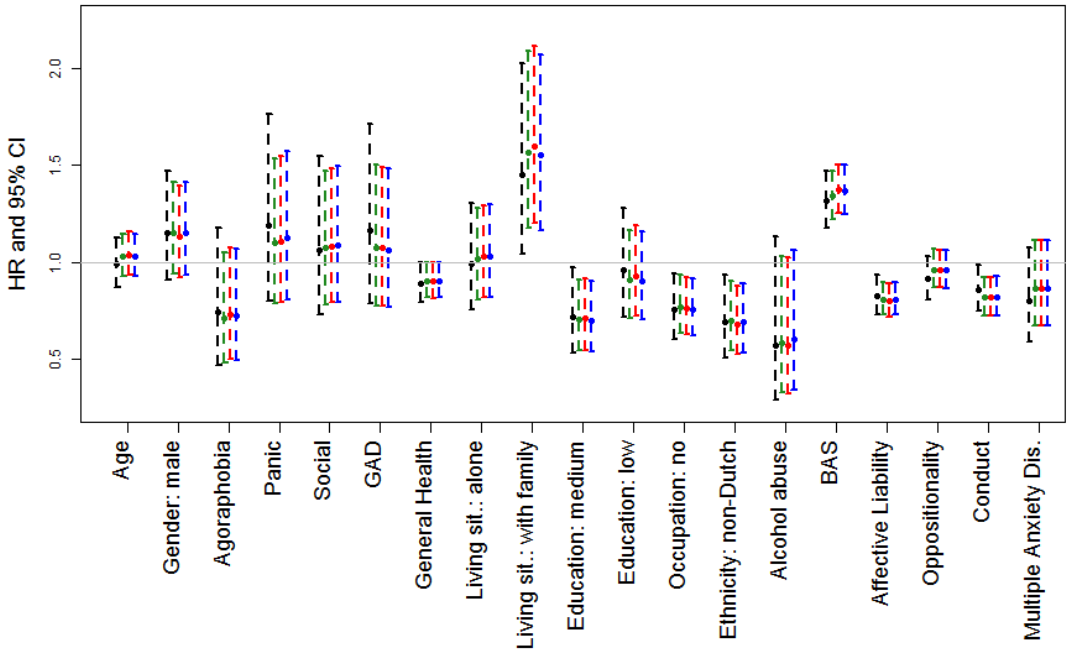


Figure 6.6: Point estimates and 95% confidence intervals for parameters in time to 50% improvement on the BAS and BSI-12 scale, for the unweighed Cox model (black) and IPCW Cox model using Cox models 1 (green), 2 (red) and 3 (blue) for time to censoring. All weights used in IPCW are unstabilized.

is below the classical S curve (PL) for model 2 and the S_{IPCW} (PL_{IPCW}) is above the classical S curve (PL) for model 3. These analyses suggests that the IPCW method has only little effect on the estimated overall survival curve, however, it does seem to have an effect on the estimated survival curves for specific individuals in the study.

Schat et al. [24] expected that either patients who are nearly recovered or severe patients would be lost to follow-up. Clinicians believe that patients are ashamed to visit a psychiatrist and will therefore stop the visits as soon as they feel good enough. This results in a trend of censoring patients who are nearly recovered. On the other hand, patients who are severely ill will lose faith in the therapy and might stop visiting their psychiatrist. If both assumptions are true, the two mechanisms may cancel eachother out. This may be the reason why the effect of the IPCW estimator is small.

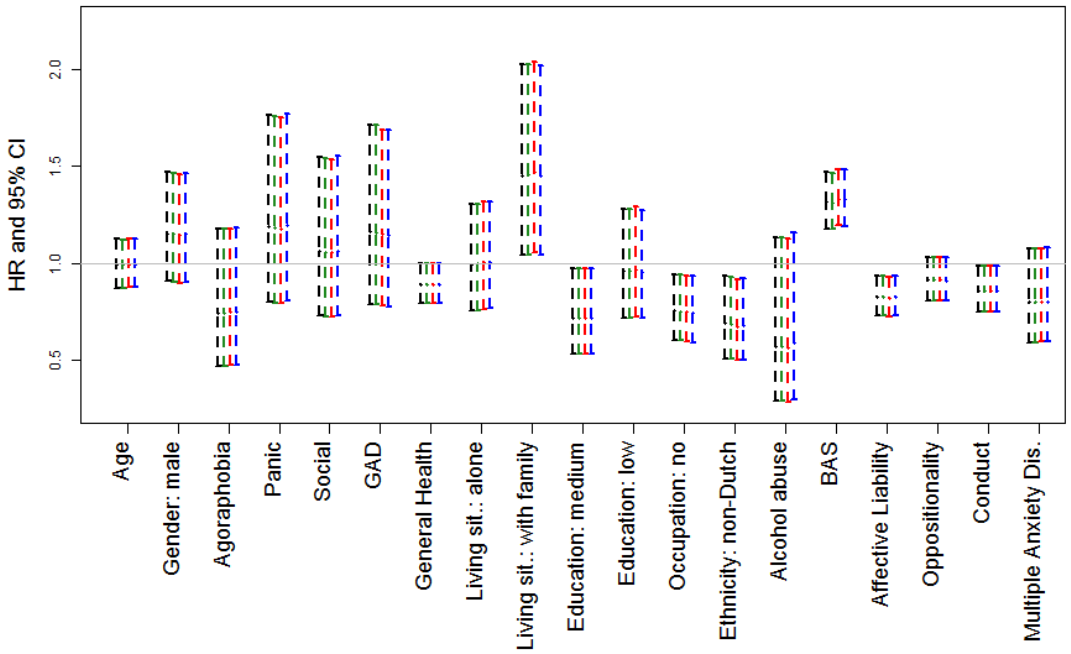
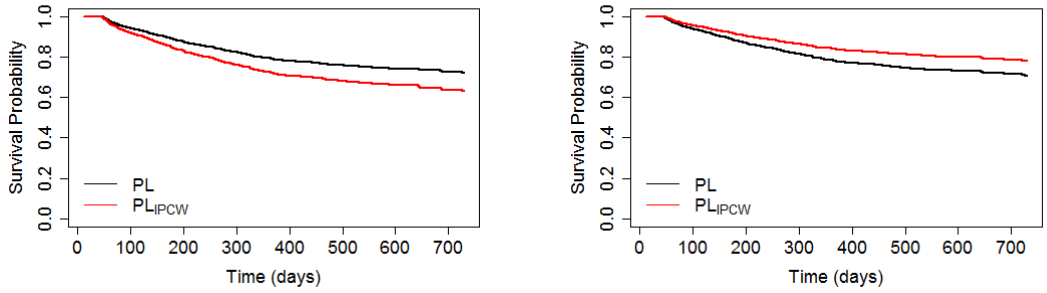


Figure 6.7: Point estimates and 95% confidence intervals for parameters in time to 50% improvement on the BAS and BSI-12 scale, for the unweighed Cox model (black) and IPCW Cox model using Cox models 1 (green), 2 (red) and 3 (blue) for time to censoring. All weights used in IPCW are stabilized.



(a) Cox model 2. *(b) Cox model 3.*

Figure 6.8: Estimated survival cures with (red) and without (black) the IPCW weights for the individual that has the largest value of $|\hat{\beta}_X^{IPCW} Z_j - \hat{\beta}_X^0 Z_j|$. Unstabilized weigths are used for both models.

Chapter 7

Simulation Study

In the previous chapter, IPCW was used to perform survival analysis in the presence of dependent censoring. This method should perform better in this situation than the standard Product-Limit Estimator that assumes independent censoring. In this chapter, a Monte Carlo simulation study is carried out to assess the performance of IPCW. A short introduction to Monte Carlo simulations is given and then the algorithm developed in this thesis to simulate data including dependent censoring is described.

In this chapter, the classical Product-Limit Estimator, that does not include the IPCW weights in its estimations, is referred to as the *standard method*.

7.1 Introduction to Monte Carlo Simulations

In research statistical methods are applied to datasets which are a sample of the population under examination. However, the mathematical theory behind the models and methods is based on many assumptions, as the assumption that the method is applied to the whole population. Therefore, the method will not give the true parameter θ, but an estimation of this parameter, $\hat{\theta}$. When a different sample from the population is used, a different estimate for the parameter θ will be found. Both estimates are samples from the sampling distribution of $\hat{\theta}$.

Monte Carlo simulations can be used to investigate the sampling distribution of $\hat{\theta}$. Data is repeatedly drawn from an artificial population and for each sample, $\hat{\theta}$ is calculated. The vector of estimates estimates $\hat{\theta}_1, \ldots, \hat{\theta_M}$ can be used to estimate the sampling distribution of $\hat{\theta}$. The Monte Carlo simulation procedure can be described as follows.

Monte Carlo Procedure

Step 1: Specify the artificial population, i.e. develop an algorithm to generate data samples.

Step 2: Sample data from the artificial population of interest.

Step 3: Calculate the parameter estimate $\hat{\theta}$ of interest and store in vector $\hat{\boldsymbol{\theta}}$.

Step 4: Repeat Step 2 and 3 M times, where M is the number of trials.

Step 5: Draw conclusions from the distribution of vector $\hat{\theta}_1, \ldots, \hat{\theta_M}$, which is the Monte Carlo estimate of the sampling distribution of $\hat{\theta}$.

Usually large numbers as $10,000$ or $25,000$ are choosen for M.

7.2 Monte Carlo Simulations to test IPCW Method

In this section it is described how Monte Carlo Simulations is used to compare the results of the IPCW method with the results of the standard method.

7.2.1 Defining the Artificial Population

For this simulation study, a patient population is defined. For each individual covariates *Age* and *Gender* are generated. *Age* is normally distributed with mean 50 and standard deviation 10. The Z-values for *Age* are used in calculations and are denoted by *ZAge*. 50% of the subjects in the population is male (*Gender* = 0) and 50% is female (*Gender* = 1).

7.2.2 Data Generation

For each subject i in the data sample, the observed pairs (t_i, δ_i) have to be generated, where $t_i = \min(x_i, c_i)$ and $\delta_i = \mathbb{1}_{\{t_i = x_i\}}$. To obtain the observed data (t_i, δ_i), time to event x_i and time to censoring c_i for each patient should first be generated.

Cox proportional hazards models can be simulated by generating exponentially distributed survival times (Bender et al. [3]). Therefore, Exponential distributions are used to generate individual times, i.e. $X \sim \exp(\lambda_X)$ and $C \sim \exp(\lambda_C)$, with the survival functions

$$S_X(t) = \exp(-\lambda_X t), \tag{7.2.1}$$
$$S_C(t) = \exp(-\lambda_C t), \tag{7.2.2}$$

where λ_X and λ_C are the hazard rates for event and censoring respectively. If λ_X is dependent on the covariates, the probability of experiencing an event is associated with the covariates. In case of dependent censoring, not all subjects in the study have the same probability of being censored, i.e. the hazard rate λ_C depends on the covariates. For this simulation study, λ_X and λ_C are choosen as follows.

$$\lambda_X(t|\boldsymbol{Z}) = \lambda_{0_X}(t) \exp(\boldsymbol{\beta}^t \boldsymbol{Z}), \tag{7.2.3}$$
$$\lambda_C(t|\boldsymbol{W}) = \lambda_{0_C}(t) \exp(\boldsymbol{\phi}^t \boldsymbol{W}), \tag{7.2.4}$$

where λ_{0_X} and λ_{0_C} are the baseline hazards for time to event and time to censoring, and $\boldsymbol{\beta}$ and $\boldsymbol{\phi}$ are the parameter vectors for time to event and time to censoring respectively. The covariates for time to event ($\boldsymbol{Z}$) and time to censoring ($\boldsymbol{W}$) do not necessarily have to be the same. For these simulations, both $\boldsymbol{Z}$ and $\boldsymbol{W}$ include *ZAge* and *Gender*. It is assumed that the baseline hazards for time to event and time to censoring are constant over time, i.e. $\forall t \geq 0, \lambda_{0_X}(t) = \lambda_{0_X}$ and $\lambda_{0_C}(t) = \lambda_{0_C}$, which result in constant hazard rates λ_X and λ_C for each subject.

The dependence between the variables X and C can be adjusted by varying $\boldsymbol{\beta}$ and $\boldsymbol{\phi}$. The percentage of censored subjects can be varied by adjusting λ_{0_X} and λ_{0_C}.

The event time x_i, the censoring time c_i and the observed data (t_i, δ_i) for each patient i in the data sample with covariates $Gender_i$ and $ZAge_i$, $i = 1, \ldots, n$, can be generated by using the following algorithm.

Generating x_i, c_i and observed data (t_i, δ_i) for patient i:

Step 1: Generate $u_{i_1}, u_{i_2} \sim U(0, 1)$.

Step 2: Generate x_i and c_i with the theorem of probability integral transform:

$$x_i = F_i^{-1}(u_{i_1})$$
$$c_i = G_i^{-1}(u_{i_2}),$$

where $F_i(t) = 1 - \exp(\lambda_X(t|Z_i))$ and $G_i(t) = 1 - \exp(\lambda_C(t|W_i))$. $\lambda_X(t|Z_i)$ and $\lambda_C(t|W_i)$ are derived from (7.2.3) and (7.2.4).

Step 3: Derive t_i and c_i:

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

and

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

The procedures for generating the hazard rates and the observed data are applied to all M generated data samples.

7.2.3 Estimating Survival Curves and Model Parameters

The aim of this simulations study is to analyse whether the IPCW method performs better than the standard method in case of dependent censoring, i.e. whether the survival curve estimated with the IPCW method estimates the real survival curve better than the standard method. This can be done by comparing the survival curves estimated by both

methods with the true survival curve. The Product-Limit estimator will be computed at p predefined time points $\tau_1, \ldots, \tau_p$.

Real Survival Curve

The real survival curve can be estimated parametrically or from the Monte Carlo simulation results.

In the parametric approach, the survival curve can be derived as follows.

$$\begin{aligned} S(x) &= \int S(x|\boldsymbol{Z})\, f_{\boldsymbol{Z}}\, d_{\boldsymbol{Z}} \\ &= \int\int S(x|age, gender)\, f_{ZAge}\, f_{Gender}\, d_{ZAge}\, d_{Gender} \\ &= \int \{S(x|age, male)\, f_{ZAge}\, p_{male} + \\ &\qquad S(x|age, female)\, f_{ZAge}\, p_{female}\}\, d_{ZAge} \\ &= p_{male} \int S(x|age, male) f_{ZAge}\, d_{ZAge} + \\ &\qquad p_{female} \int S(x|age, female) f_{ZAge}\, d_{ZAge}, \end{aligned}$$

where p_i is the probability of i, f_i the probability density function of i, and $S(x|\boldsymbol{Z}) = S_0(x)^{\exp(\boldsymbol{\beta}^t \boldsymbol{Z})}$ (see (2.5.4)).

In the Monte Carlo simulation, the survival model is fit on the event times $x_1, \ldots, x_n$ in each simulation j. The survival probabilities are estimated with the Product-Limit estimator for the predefined time points $\tau_1, \ldots, \tau_p$, resulting in survival estimates $\hat{S}_j(\tau_1)$, $\ldots, \hat{S}_j(\tau_p)$ for each simulation j, $j = 1, \ldots, M$. This approach represents the real life situation in which the parameter models are unknown, but the event times of all subjects in the dataset are observed.

In this simulation study, the Monte Carlo approach was used to estimate the real survival curve.

The estimated survival probabilities for the real survival curve are denoted as $\hat{S}_j(\tau_k)$ for $j = 1, \ldots, M$ and $k = 1, \ldots, p$. The estimates of the the Cox proportional hazards model parameters are denoted as $\hat{\beta}_j$.

Survival Curve estimated with the Standard Method

For each sample j, the observed data $(\boldsymbol{t}, \boldsymbol{\delta})$ for all patients in the data sample is used to estimate the survival curve and Cox proportional hazards model with the standard method. Results are denoted as $\tilde{S}_j(\tau_k)$ and $\tilde{\beta}_j$.

Survival Curve estimated with the IPCW Method

The generated observed data is also used to estimate the survival probabilities and Cox proportional hazards model with the IPCW method, using the `R` procedure described in chapter 6. In each simulation j, the parameter $\breve{\phi}_j$ of the censoring model is estimated. Stabilized and unstabilized weights lead to the same Product-Limit estimator, denoted

as $\check{S}_j(\tau_k)$. The two types of weights will give different parameter estimates for the Cox model, denoted as $\check{\beta}_j^{Stab}$ and $\check{\beta}_j^{Unstab}$ respectively.

Summary Results

Let $\hat{S}(\tau_k)$, $\tilde{S}(\tau_k)$, and $\check{S}(\tau_k)$ be the estimated survival probabilities obtained with the event times, standard model, and IPCW respectively. The M simulations are summarized by taking the mean survival probability at each time point τ_k,

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

where $k = 1, \ldots, p$. These means can be used to plot the survival curves estimated by each method.

The parameter estimates $\hat{\beta}_j$, $\tilde{\beta}_j$, $\check{\phi}$, $\check{\beta}_j^{Stab}$ and $\check{\beta}_j^{Unstab}$ are estimated in each simulation, resulting in M values for each estimator. The distribution of these estimators can be assessed by estimating the mean value and standard deviation of the M estimates.

7.3 Expectations

The IPCW method estimates the censoring model and compensates for the subjects that are censored by giving extra weight to those subjects that remain at risk. In the case of independent censoring, both IPCW and standard method should, give perfect estimates of the true survival curve and model parameters. In case of dependent censoring, IPCW should give a perfect estimate of the true survival curve. The standard method will not work properly, because it is based on the assumption of independent censoring. Different scenarios will be considered, such that the behaviour of both standard and IPCW method can be assessed under different situations.

In the sequel all quantities that will be changed in the simulation study are described.

7.3.1 Number of Subjects

One parameter that can be changed in the simulations is the number of subjects, n. The bigger the number of subjects, the more precise the estimates $\tilde{\beta}$, $\check{\phi}$, $\check{\beta}^{Stab}$, and $\check{\beta}^{Unstab}$, hence, the more precise the estimated survival models. If the ϕ^{IPCW} parameter is estimated inprecisely, the IPCW weights will be imprecise as well, causing some subjects to be weighted too lightly and some too heavily. This may result in less accurate estimates for the time to event model compared to the standard method.

7.3.2 Strength of the Censoring Model

The strength of the censoring model can be varied by choosing different values for the parameter ϕ, the regression coefficient in the time to censoring model. If the censoring model is weak, i.e. the difference in censoring probabilities between subjects is small, the assumption of independent censoring is almost met. Hence, the standard model may work as good as the IPCW model, making the IPCW a cumbersome method resulting in nearly the same model estimates.

7.3.3 Percentage of Censored Subjects

The percentage of censored subjects can be varied by adjusting the values of λ_{0_X} and/or λ_{0_C} the baseline hazards for the time to event and time to censoring model respectively. In the presence of dependent censoring, the percentage of censored subjects might be important to assess the benefits of the IPCW method. Obviously, if the percentage of censored subject is low, it is less important to compensate for the censored subjects than when a high percentage of subjects is censored. The standard method may be just as usefull as the IPCW.

In this situation, the censoring model may not be precisely estimated and this may cause imprecise weights, leading to a wrong estimated time to event model.

Different values corresponding to the parameters discussed above will be considered in the simulations study.

7.4 Simulation Results

As previously indicated, the quantities that were varied in the simulation study are the number of subjects n, the strength of the censoring model (ϕ), and the percentage of censored subjects. For this simulations study, an exponential distribution for time to event, X, was assumed, i.e.

$$S_X(t) = \exp(-\lambda_X t),$$

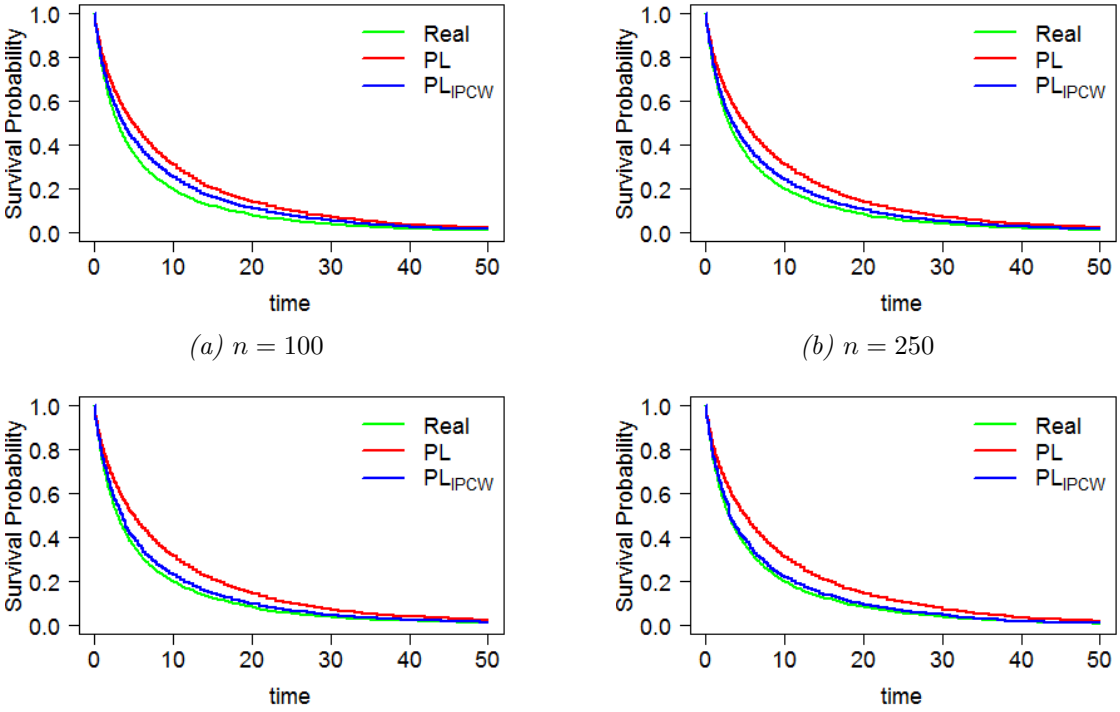
with $\lambda_X(t|\boldsymbol{Z}) = \lambda_{0_X}(t)\exp(\boldsymbol{\beta}^t\boldsymbol{Z})$. Here, $\boldsymbol{Z}_i = (ZAge_i, Gender_i)$, with $Gender = 0$ for males and $Gender = 1$ for females. For each simulation, time to event is simulated with $\beta = (0.5, 1.5)$ and $\lambda_{0_X} = 0.1$. Hence, old women have the highest probability of experiencing the event of interest.

7.4.1 Number of Subjects

To investigate the effect of sample size n on the IPCW result, Monte Carlo simulation was performed with censoring model $\phi = (1.5, 5)$, the same censoring percentage 35% ($\lambda_{0_C} = 0.00557$), but for varying n. Simulations become extremely time consuming as the sample size increases.

As expected, the survival curve estimated by employing IPCW becomes closer to the real survival curve for increasing n, while the standard model's performance does not

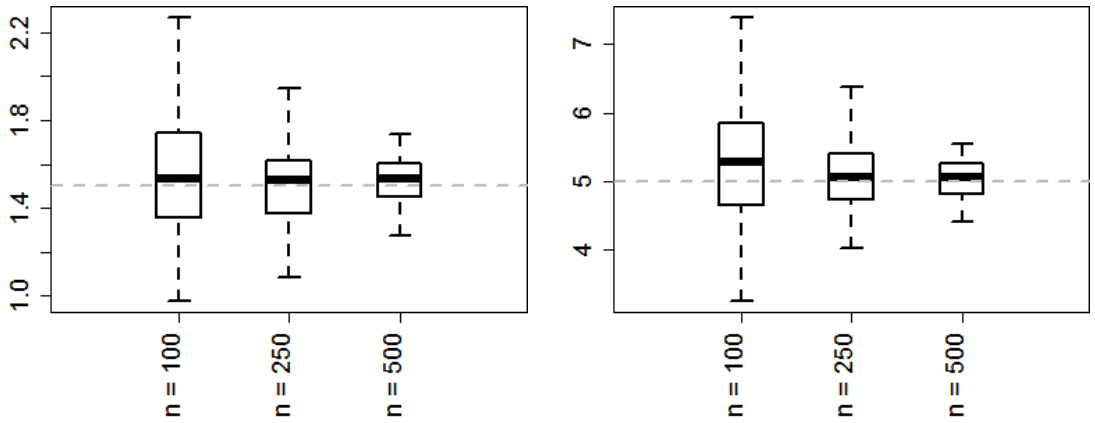
change. Results are illustrated in Figure 7.1. The improvement of the IPCW result is due to more precise regression coefficients estimates for big sample sizes. In Figure 7.2 boxplots of the M estimated regression parameters ($\phi = (1.5, 5)$) are shown for sample size equal to 100, 250, and 500.



(a) $n = 100$ *(b)* $n = 250$

(c) $n = 500$ *(d)* $n = 2500$

Figure 7.1: The real Product-Limit curve (green) and the survival curves estimated with the standard method (red) and IPCW (blue) for varying number of subjects, n.



(a) $\check{\phi}_1$ *(b)* $\check{\phi}_2$

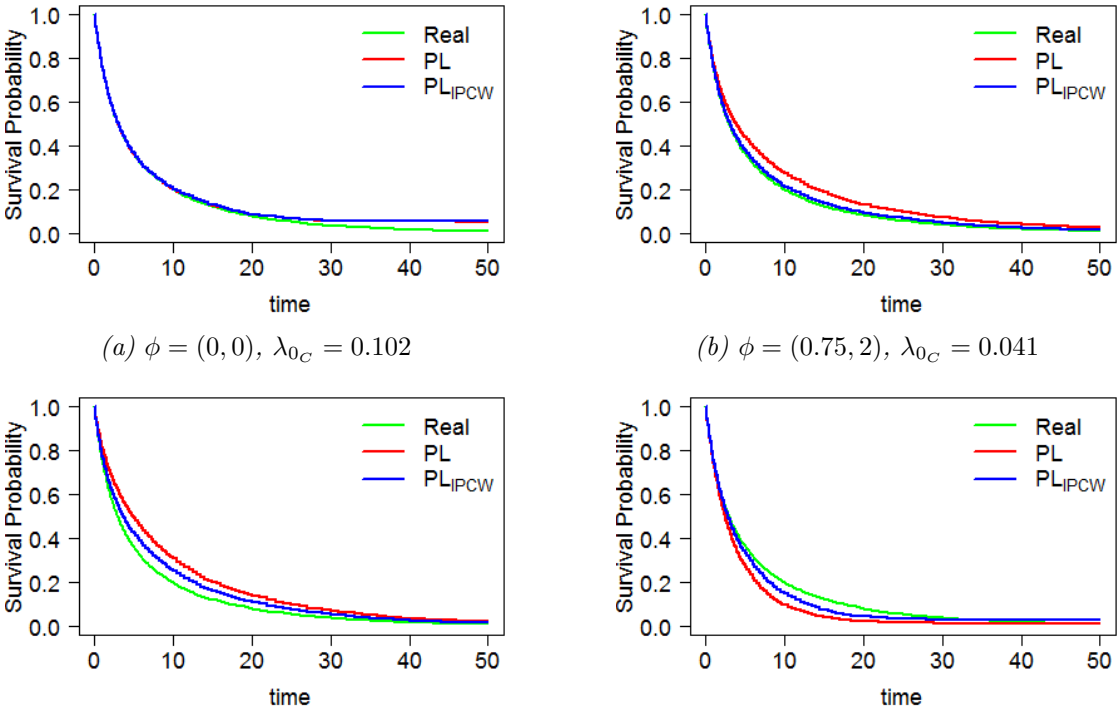
Figure 7.2: Boxplots of the M estimated regression parameters $\check{\phi} = (\check{\phi}_1, \check{\phi}_2)$ *for the four chosen sample sizes n.*

7.4.2 Strength of the Censoring Model

The strength of the censoring model is defined by the parameter $\phi = (\phi_1, \phi_2)$, where ϕ_1 is the regression coefficient for $ZAGe$ and ϕ_2 indicates the gender effect. For bigger values of ϕ_1 and ϕ_2, the censoring is more dependent on the covariates. For these simulations sample size $n = 100$ was chosen and λ_{0_C} was adjusted for each value of ϕ to maintain 35% censoring.

In Figure 7.3, results corresponding to four different censoring models are shown. Simulations were performed for data without dependent censoring ($\phi = (0, 0)$), which results in good approximations to the real survival curve by both the standard and IPCW method. Results are shown in Figure 7.3a. For the censoring models with ϕ equal to (0.75, 2) or (1.5, 3), subjects who have a higher probability of experiencing the event (old women) also have a higher chance of being censored. This is confirmed by the positive correlation between the generated x_i's and c_i's, which is equal to 0.354 and 0.350 respectively. In this case, less events are observed and survival probabilites are overestimated. However, overestimation by IPCW is smaller than by the standard method. Results are illustrated in Figure 7.3b and Figure 7.3c.

If ϕ_1 and ϕ_2 are negative, e.g. $\phi = (-1.5, -3)$, subjects with a high probability of experiencing the event, have a low probability of being censored. This causes a negative correlation (-0.170) between the generated x_i's and c_i's. Many events are observed and survival probabilities are underestimated. Underestimation is smaller when employing IPCW, see Figure 7.3d .



(a) $\phi = (0, 0)$, $\lambda_{0_C} = 0.102$ *(b)* $\phi = (0.75, 2)$, $\lambda_{0_C} = 0.041$

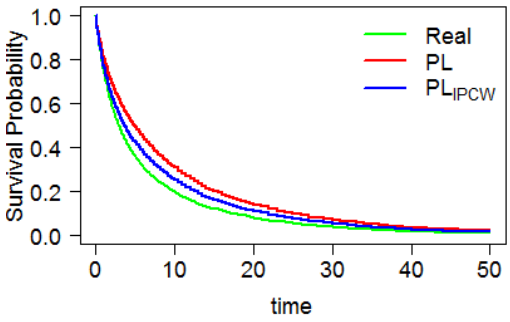
(c) $\phi = (1.5, 5)$, $\lambda_{0_C} = 0.00557$ *(d)* $\phi = (-1.5, -3)$, $\lambda_{0_C} = 0.222$

Figure 7.3: The real Product-Limit curve (green) and the survival curves estimated with the standard method (red) and IPCW (blue) for different parameters ϕ in the censoring model.

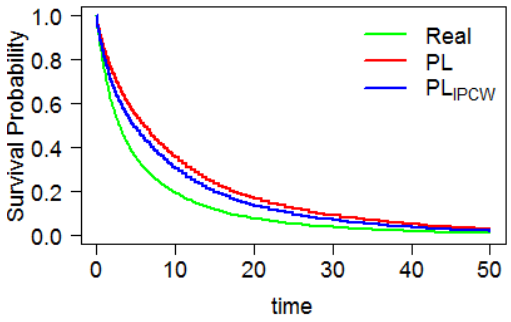
7.4.3 Percentage of Censored Subjects

By adjusting the parameter λ_{0_C}, the percentage of censored subjects can be varied. Simulations were performed with parameter ϕ in the censoring model equal to (1.5, 5) on datasets with sample size $n = 100$.

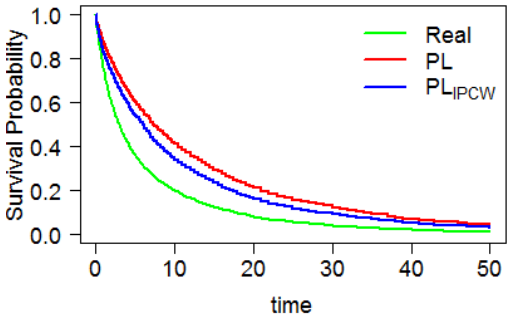
Results in Figure 7.4 show that the performance of both methods become worse with increasing percentage of censoring. The parameter ϕ in the censoring model is estimated more precise for high censoring percentages. Due to the occurrence of less events, as expected, the precision of the parameter estimates of the time to event model declines with increasing censoring percentage. In Figure 7.5 boxplots for the regression parameters estimated with unstabilized and stabilized weights for the three chosen censoring percentages are shown.



(a) 35%

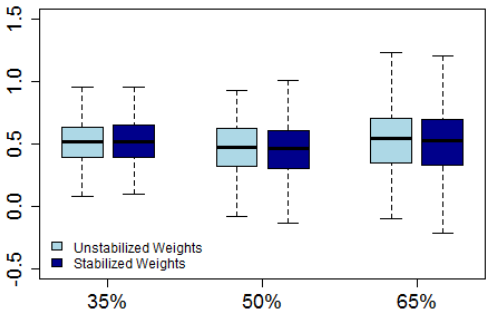


(b) 50%

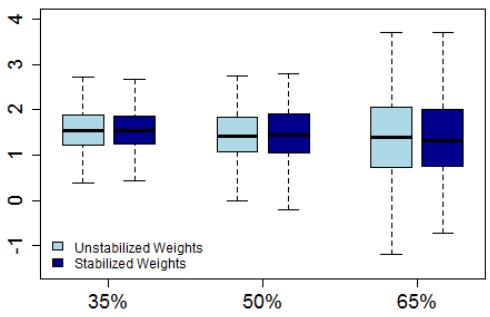


(c) 65%

Figure 7.4: The real Product-Limit curve (green) and the survival curves estimated with the standard method (red) and IPCW (blue) for varying censoring percentages.



(a) $\check{\beta}_1^{Unstab}$ *and* $\check{\beta}_1^{Stab}$



(b) $\check{\beta}_2^{UnStab}$ *and* $\check{\beta}_2^{Stab}$

Figure 7.5: Boxplots for the M estimated regression parameters $\check{\beta} = (\check{\beta}_1, \check{\beta}_2)$ based on with unstabilized and stabilized weights for the three chosen censoring percentages.

7.4.4 Conclusions

The simulations study carried out in this thesis show the performance of IPCW for situations where different number of patients and percentage of censored subjects are considered. As expected IPCW performs better than the standard method in case of dependent censoring. Different values of the censoring model have been considered in the simulations.

The function `coxph` in the survival library in `R` does not give a perfect fit for the time to censoring model. This leads to imprecise weights for IPCW. For the Product-Limit estimator for time to censoring, imprecise weights still result in a better performance with IPCW. To get optimal results, a precise fit for the censoring model seems crucial.

Chapter 8

Discussion

The focus of this thesis was on two different problems, one related to interval censored data and the other concerning possible presence of dependent censoring in the Routine Outcome Monitoring (ROM) data. To address the first problem, a bootstrap procedure was proposed. With this method, the precision of the hazard ratio estimates can be assessed. To deal with the second aspect, the Inverse Probability Censoring Weight (IPCW) methodology was used. A proper algorithm was developed to generate dependent data. Simulations were performed to assess the bias in the estimation of the survival function in case dependent censoring is ignored.

8.1 Interval Censoring

`R` packages `Icens` and `intcox` can fit the Product-Limit estimator and Cox model on interval censored data. However, no standard errors of the estimated regression coefficients are given. In this thesis, a bootstrap procedure was proposed to assess the precision of the parameter estimates. This method applies `intcox` on M samples from the dataset, resulting in M parameter estimates. The distribution of these estimates gives an idea of the precision of the estimator. Unfortunately, `intcox` is not applicable on datasets containing more than 30% right censoring. Therefore, it was not applied to the ROM data, since the attrition rate in this dataset was higher than 30%. Although it was not applied to the ROM dataset, the bootstrap procedure is useful, because it can be applied to datasets with a low attrition rate.

Researcher A. Schat is working on a new project on a different sub population of patients in the ROM dataset. It is expected that the attrition rate in this sub population will be very low. Therefore, the bootstrap procedure may be applicable to this dataset, and an interval analysis will be performed with the bootstrap method. Results will be presented in an article.

8.2 Dependent Censoring

Clinicians' experience suggests that for patients in the ROM dataset, the probability of being lost to follow-up is related to health status. IPCW was developed to account for this type of dependent censoring data. The IPCW method compensates for censored subjects by increasing the weights of subjects that remain at risk. For the ROM dataset, only covariate *Mental Health* was found to have a significant effect on the censoring time (see Table 6.7), but the final survival curve did not alter considerably by accounting for this. This suggests that there is no strong censoring mechanism in the ROM dataset.

The simulation study showed that a good fit for the censoring model is crucial for the performance of IPCW. Imprecise censoring probabilities result in incorrect censoring weights. This leads to a bad fit for the Cox model for time to event. However, less precise estimates of the censoring probabilities are needed to improve the Product-Limit estimator for time to event. Usually interest is in the effect of covariates on the time to event, hence Cox's model is required.

Clinicians' expectation concerning the censoring mechanism could be interpreted as either dependent or informative censoring. In this thesis the focus was on dependent censoring. Therefore, the direct dependence between event and censoring times was not assessed. Several methods have been developed to detect and correct for informative censoring. These can be applied to the ROM dataset to investigate whether the event and censoring times are directly dependent instead of dependent through covariates. This can be investigated in future research.

Appendix A

R Code for Bootstrapping Procedure

In this appendix, the full versions of `intcox.boot` and `intcox.bootstrapping`, the functions for the bootstrapping procedure described in subsection 4.4.2, are given.

A.1 Function `intcox.boot`

```
> # Function to perform one intcox estimation
> intcox.boot <- function(i, formula, data)
 {
   # input:
   # i = unused integer
   # formula = formula to be implemented in the intcox function,
   #           Surv object (type = interval2) ~ covariates.
   # data = data that includes the left and right boundaries of the
   #        censoring intervals, together with the covariates.
   #
   # output:
   # coef = parameter estimates for the Cox PH for interval censored
   #        data, estimated with the function intcox.
   # term = indicator for the reason of termination,
   #         1 - algorithm converged
   #         2 - no improvement of likelihood possible, the iteration
   #             number is shown
   #         3 - algorithm did not converge - maximum number of
   #             iteration reached
   #         4 - inside precondition(s) are not fulfilled at this
```

```
  #             iteration

  # Take bootstrap sample
  boot.sample <- sample(1:nrow(data), size = nrow(data),
                        replace = T)
  data.sample <- data[boot.sample,]

  # Estimate regression parameters for bootstrap sample
  boot.fit <- intcox(formula, data = data.sample, no.warnings = T)

  # Return results
  return(list(coef = coefficients(boot.fit),
              term = boot.fit$termination
              )
         )
}
```

A.2 Function `intcox.bootstrapping`

```
> intcox.bootstrapping <- function(formula, data, n.iter = 1000,
                                   alpha = 0.05)
 {
  # input:
  # formula = formula to be implemented in the intcox function,
  #           Surv object (type = interval2) ~ covariates.
  # data = data that includes the left and right boundaries of the
  #        censoring intervals, together with the covariates.
  # n.iter = number of iterations in the bootstrap, default = 1000
  # alpha = significance level, default = 0.05, corresponding to a
  #         95% CI
  #
  # output:
  # CI.coefficients = upper and lower bound of the confidence
  #                   interval of the parameter estimates for each
  #                   of the covariates.
  # CI.HR = upper and lower bounds of the confidence interval of
  #         the Hazard Ratios for each of the covariates.
  # terminationtypes = vector of indicators for the reason of
```

```
#                    termination, for each of the bootstrap
#                    replications.

# Load the intox-library
library(intcox)

# Apply intcox to n.iter samples from the data, using
# intcox.boot function. Save output in "boot".
boot <- lapply(1:n.iter, intcox.boot, formula = formula,
               data = data)
cov.names <- names(boot[[1]]$coef)
boot <- matrix(unlist(boot), byrow = T, nrow = n.iter)
colnames(boot) <- c(cov.names, "termination")

# Calculate the CI bounds of the coefficients and the Hazard
# ratios and store them in CICoef and CIHR. Save the termination
# types as well.
CICoef <- data.frame(
  CICoefLower = rep(NA, times = length(cov.names)),
  CICoefUpper = rep(NA, times = length(cov.names))
)

CIHR <- data.frame(
  CIHRLower = rep(NA, times = length(cov.names)),
  CIHRUpper = rep(NA, times = length(cov.names))
)

row.names(CICoef) <- row.names(CIHR) <- cov.names

# The CI bounds can be found at the following positions
k <- floor((n.iter + 1)*(alpha/2))
pos.lower <- k
pos.upper <- n.iter + 1 - k

# Compute the bounds for each of the covariates and
# hazard ratios in the model
for(i in 1:length(cov.names))
{
  cov <- cov.names[i]
  cov.ord <- order(boot[,cov])

  CICoef$CICoefLower[i] <- boot[cov.ord, cov][pos.lower]
```

```
    CICoef$CICoefUpper[i] <- boot[cov.ord, cov][pos.upper]
    CIHR$CIHRLower[i] <- exp(CICoef$CICoefLower[i])
    CIHR$CIHRUpper[i] <- exp(CICoef$CICoefUpper[i])
  }

  return(list(CI.coefficients = CICoef,
              CI.HR = CIHR,
              terminationtypes = boot[,"termination"]))
}
```

Appendix B

R Code for IPCW Method

In this appendix, the full versions of `survIPCW` is given (see section 5.4), together with the IPCW procedure described in chapter 6.

B.1 Function `survIPCW`

```
> survIPCW <- function(Tstart, Tstop, status, weights)
 {
   # Order the long format data according to time.
   ord <- order(Tstart)
   Tstart <- Tstart[ord]
   Tstop <- Tstop[ord]
   status <- status[ord]
   weights <- weights[ord]

   # Time points at which the events happen.
   tout <- Tstop[status==1]
   tout <- unique(tout)

   # Number of time points at which the event happen.
   nout <- length(tout)

   di <- Yi <- rep(NA, nout)
   for (i in 1:nout)
   {
     # The number of event at time point tout[i].
     di[i] <- sum((Tstop==tout[i] & status==1)*weights)
```

```
    # Number at risk (weighted) at time point tout[i].
    Yi[i] <- sum((Tstop == tout[i])*weights)
  }

  # Calculate the Product-Limit estimator.
  surv <- cumprod(1 - di/Yi)

  # Return the timepoints and survival estimates.
  return(data.frame(time=tout, surv=surv))
}
```

B.2 IPCW method in R

```
> # Required libraries.
> library(survival)
> ##############################################################
> # Step 1: Transforming the data 'test' into a long format. #
> ##############################################################
> # Define new column to store starting points of time intervals
> test$Tstart <- 0
> # Times at which an events/censoring happens.
> times <- sort(unique(test$time[test$censored==1 ||
                               test$status == 1]))
> # Define new column with the censoring indicator.
> test$censored <- 1-test$status
> # Split data over all event and censoring times and order
> # according to id's and time intervals.
> test.long <- survSplit(test, cut=times, end="time",
                         start="Tstart", event="status",
                         id = "id")
> test.long <- test.long[order(test.long$id,test.long$time),]
> test.long.cens <- survSplit(test, cut=times, end="time",
                              start="Tstart", event="censored",
                              id = "id")
> test.long.cens <- test.long.cens[order(test.long.cens$id,
                                         test.long.cens$time),]
> # Include the correct censoring times:
> test.long$censored <- test.long.cens$censored
```

```
> test.long$id <- as.numeric(test.long$id)
> ################################################################
> # Step 2: Fit the models for time to censoring, and calculate #
> # Product-Limit estimator.                                     #
> ################################################################
> # Censoring model without covariates.
> C0 <- coxph(Surv(Tstart, Tstop, censored) ~ 1,   data=test.long)
> # Censoring model with covariates.
> CZ <- coxph(Surv(Tstart, Tstop, censored) ~ cov, data=test.long)
> # Calculate K0ti's.
> C0fit <- summary(survfit(C0), times = test.long$Tstart)
> test.long$K0ti <- C0fit$surv
> # Calculate KZti's (for time-independent covariates).
> test.long$KZti <- NULL
> for(i in 1:nrow(test.long))
 {
   datai <- test.long[i,]

   sfiCZ <- survfit(CZ, newdata = datai)
   ssfiCZ <- summary(sfiCZ, times = datai$Tstart)
   test.long$KZti[i] <- ssfiCZ$surv
 }
> ################################
> # Step 3: Compute IPCW weights #
> ################################
>
> # Unstabilized weights.
> test.long$WUnStab <- 1/test.long$KZti
> # Stabilized weights.
> test.long$WStab <- test.long$K0ti/test.long$KZti
> ##########################################################
> # Step 4: Estimate the survival model for time to event #
> # in the absence of censoring using the IPCW Weights    #
> ##########################################################
> # Product-Limit estimates:
>
> # Survival without IPCW weigths:
> res <- survfit(Surv(Tstart,Tstop,status) ~ 1,
                 data=test.long)
> # survival with IPCW stabilized weights:
> resWStab <- survfit(Surv(Tstart,Tstop,status) ~ 1,
                      data=test.long, weights=WStab)
```

```
> # survival with IPCW unstabilized weights:
> resWUnStab <- survfit(Surv(Tstart,Tstop,status) ~ 1,
                        data=test.long, weights=WUnStab)
> # Cox Models:
> # Original analysis without IPCW
> CoxW0 <- survfit(coxph(Surv(Tstop,status) ~ cov,
                         data=test.long))
> # IPCW with stabilized weights
> CoxWStab <- survfit(coxph(Surv(Tstart,Tstop,status) ~ cov,
                            data=test.long, weights = WStab))
> # IPCW with unstabilized weights
> CoxWUnStab <- survfit(coxph(Surv(Tstart,Tstop,status) ~ cov,
                              data=test.long, weights = WUnStab))
>
```

Appendix C

All R Code used for this Thesis

In this appendix, all code used to do the calculations in this thesis is given.

C.1 R Code for Chapter 3

C.1.1 Load and Transform ROM data

First, the ROM data was loaded and all variables were adjusted to fit the analysis by Schat et al. [24].

```
> library(foreign)
> ########################################
> ## Loading and transforming the data ##
> ########################################
>
> data <- read.spss("DataTotal.sav")
> data <- as.data.frame(data)
> ########################
> # Categorical Variables #
> ########################
>
> #  Make factors:
> data$preadultonset <- as.factor(data$preadultonset)
> data$agorafobie <- as.factor(data$agorafobie)
> data$paniekagfob <- as.factor(data$paniekagfob)
> data$socfob <- as.factor(data$socfob)
> data$GAS <- as.factor(data$GAS)
> #data$herstelkrap <- as.logical(data$herstelkrap)
> data$alcohol <- as.factor(data$alcohol)
> data$middelen <- as.factor(data$middelen)
> # Set the same reference categories as in the article, to make
> # comparisons with the original article easier:
```

```
> data$gendernum <- relevel(data$gendernum, ref="vrouw")
> data$ethnicity <- relevel(data$ethnicity, ref="dutch")
> data$occupation <- relevel(data$occupation,
              ref="employed studying primary caregiver children")
> data$education <- relevel(data$education, ref="high")
> data$livingsituation3 <- relevel(data$livingsituation3,
              ref="living independently with partner and or children")
> data$singleanx <- relevel(data$singleanx,
                          ref="single anxiety disorder")
> data$relationalstatus <- relevel(data$relationalstatus,
                                ref="living with partner or married")
> ########################
> # Continuous variables #
> ########################
>
> # Calculate the correct Z-values for each of the continuous variables
> data$ZLeeftijd <- (data$Leeftijd - mean(data$Leeftijd))/
                    sd(data$Leeftijd)
> data$ZMADRS.1 <- (data$MADRS.1 - mean(data$MADRS.1))/sd(data$MADRS.1)
> data$ZREM.1 <- (data$REM.1 - mean(data$REM.1))/sd(data$REM.1)
> data$ZBSI_12.1 <- (data$BSI_12.1 - mean(data$BSI_12.1))/
                    sd(data$BSI_12.1)
> data$ZBAS.1 <- (data$BAS.1 - mean(data$BAS.1))/sd(data$BAS.1)
> # DAPP-SF
> data$ZVOLG <- (data$VOLG - mean(data$VOLG))/sd(data$VOLG)
> data$ZCOGVER <- (data$COGVER - mean(data$COGVER))/sd(data$COGVER)
> data$ZIDENT <- (data$IDENT - mean(data$IDENT))/sd(data$IDENT)
> data$ZAFFLAB <- (data$AFFLAB - mean(data$AFFLAB))/sd(data$AFFLAB)
> data$ZBEPRIK <- (data$BEPRIK - mean(data$BEPRIK))/sd(data$BEPRIK)
> data$ZORDDWA <- (data$ORDDWA - mean(data$ORDDWA))/sd(data$ORDDWA)
> data$ZGESLO <- (data$GESLO - mean(data$GESLO))/sd(data$GESLO)
> data$ZGEMPH <- (data$GEMPH - mean(data$GEMPH))/sd(data$GEMPH)
> data$ZPASSAG <- (data$PASSAG - mean(data$PASSAG))/sd(data$PASSAG)
> data$ZNINTIM <- (data$NINTIM - mean(data$NINTIM))/sd(data$NINTIM)
> data$ZDOMINA <- (data$DOMINA - mean(data$DOMINA))/sd(data$DOMINA)
> data$ZANGSTI <- (data$ANGSTI - mean(data$ANGSTI))/sd(data$ANGSTI)
> data$ZGEDRAG <- (data$GEDRAG - mean(data$GEDRAG))/sd(data$GEDRAG)
> data$ZWANTR <- (data$WANTR - mean(data$WANTR))/sd(data$WANTR)
> data$ZSOCONT <- (data$SOCONT - mean(data$SOCONT))/sd(data$SOCONT)
> data$ZNARCIS <- (data$NARCIS - mean(data$NARCIS))/sd(data$NARCIS)
> data$ZONHECH <- (data$ONHECH - mean(data$ONHECH))/sd(data$ONHECH)
> data$ZZBESUI <- (data$ZBESUI - mean(data$ZBESUI))/sd(data$ZBESUI)
> # SF-36
```

```
> # (The original outcomes of these variables have been "transposed" such
> # that high values represent many problems / bad health.)
> data$ZSFfyscorr <- (data$SFfyscorr - mean(data$SFfyscorr))/
                     sd(data$SFfyscorr)
> data$ZSFsoccorr <- (data$SFsoccorr - mean(data$SFsoccorr))/
                     sd(data$SFsoccorr)
> data$ZSFbliccorr <- (data$SFbliccorr - mean(data$SFbliccorr))/
                      sd(data$SFbliccorr)
> data$ZSFbemocorr <- (data$SFbemocorr - mean(data$SFbemocorr))/
                      sd(data$SFbemocorr)
> data$ZSFggzcorr <- (data$SFggzcorr - mean(data$SFggzcorr))/
                     sd(data$SFggzcorr)
> data$ZSFvitcorr <- (data$SFvitcorr - mean(data$SFvitcorr))/
                     sd(data$SFvitcorr)
> data$ZSFpijcorr <- (data$SFpijcorr - mean(data$SFpijcorr))/
                     sd(data$SFpijcorr)
> data$ZSFalgcorr <- (data$SFalgcorr - mean(data$SFalgcorr))/
                     sd(data$SFalgcorr)
```

Unfortunately, the authors of the article have made a mistake in the calculations of the time points. The correct time points are calculated with the R-code below, so these can be used in further analysis. Furthermore, the censoring intervals for the subjects who experienced an event between two visits are calculated, such that these can be used in interval censoring analysis.

```
> ####################################
> ## Calculations of corrected times ##
> ####################################
>
> # Get the needed information from the data:
>
> n <- nrow(data) # number of subjects
> # There is an event if both the BAS-score as well as the BSI_12-score
> # are <= 50% of their baseline scores (event = first time this happens).
> # Extract these scores from the data:
>
> # create vectors with only the BSI-results (BSIv) and the BAS-results
> # (BASv):
> BSIv <- data[grep("BSI_12.",names(data), fixed = T)]
> BASv <- data[grep("BAS.", names(data), fixed = T)]
> BASv <- BASv[,which(names(BASv) != "ZBAS.1")]
> BSIv <- BSIv[,which(names(BSIv) != "ZBSI_12.1")]
> # Check for each patient whether the BAS and BSI score is reduced to
> # 50% of the baseline score:
```

```
> responseBSI <- BSIv <= 0.5*BSIv[,1]
> responseBAS <- BASv <= 0.5*BASv[,1]
> # If both scores are <= 50% of the baseline score, the patient has
> # experienced the "event". For each visit, indicate whether there is an
> # event or not (NA = visit has not taken place).
> response <- responseBSI * responseBAS
> # column names will be the visit numbers:
> colnames(response) <- 1:dim(response)[2]
> # Now calculate the number of days between each of the visits for each
> # patient and each visit:
> datesBSI <- as.matrix((data[grep("BSIDATE.",names(data), fixed = T)] -
                         data$BSIDATE.1) / (24*60*60))
> datesBAS <- as.matrix((data[grep("vCPRSDATE.",names(data), fixed = T)] -
                         data$vCPRSDATE.1) / (24*60*60))
> # Sometimes, the two tests are done on different dates, resulting into
> # different dates. In such cases, the longest event time (last date)
> # will be chosen as visit date.
> combinedDates <- pmax(datesBSI, datesBAS)
> colnames(combinedDates) <- paste("ObsTIME.", 1:15, sep = "")
> # Store these times in the data
> data <- cbind(data,combinedDates)
> ##############################################
> # Calculating the event and censoring times ##
> ##############################################
>
> # If there is interval censoring, the middlepoint of the interval will
> # be taken as the event time. If there is right-censoring, the last
> # recoreded vistit will be taken as the censoring time point.
>
> # Find the visit number at which a response was detected for the first
> # time and for first dropout (visitnumber of the visit that was supposed
> # to take place but did not) for each of the patients:
> visitNumbers <- apply(response, 1, FUN = match, x = c(1,NA))
> # Vector to store the time points:
> time <- NULL
> # Now, for each patient, check whether there was an event
> # (visitNumbers[1,i] = 1). If so, take the middle point of the interval
> # (last visit and current visit) as the time point).
> # If not, censor at last visit.
> for(i in 1:n)
 {
   if(is.na(visitNumbers[1,i])) # if no event occured
   {
```

```
    time[i] <- combinedDates[i,visitNumbers[2,i]-1]
    ifelse(time[i] > 730, time[i] <- 730, NA)
  }

  else # if an event occured, take middle point:
  {
    time[i] <- 0.5*(combinedDates[i,visitNumbers[1,i]] +
                    combinedDates[i,visitNumbers[1,i]-1])
    ifelse(time[i] > 730, time[i] <- 730, NA)
  }
 }
> # Store these event/censoring times in the dataframe:
>
> data$CorrectedTimes <- time
> ######################################################################
> ## Calculating the right and left boundaries of the intervals: ##
> ######################################################################
>
> # In order to perform interval censoring in the correct way, we have
> # to know the left and right boundaries of the intervals. The boundary
> # values will be stored in the following two vectors:
> IntL <- IntR <- NULL
> # Calculate the visit numbers which represent the visits of the left
> # and right boundary of the intervals.
> for(i in 1:n)
 {
  if(is.na(visitNumbers[1,i])) # if no event occured
  {
    IntL[i] <- combinedDates[i,visitNumbers[2,i] - 1] # censored at
                                                      # last visit
    IntR[i] <- NA                  # before drop-out
  }

  else # if an event occured, it occured between the:
  {
    IntL[i] <- combinedDates[i,visitNumbers[1,i] - 1] # visit before
    IntR[i] <- combinedDates[i,visitNumbers[1,i]]     # the visit at
                                                      # which response
                                                      # was first
                                                      # detected
  }
  # Censor at 730 days -> Left boundary becomes 730, right boundary
  # becomes Inf (NA):
```

```
   if(IntL[i] > 730) IntL[i] <- 730
   if(is.na(IntR[i]) || IntR[i] > 730) IntR[i] <- NA
 }
> data$IntL <- IntL
> data$IntR <- IntR
> data$CorrectedTimes[1:15]
> ##############################
> ## Save data for later use ##
> ##############################
>
> # Now save the data so that the time points can be used later.
> save(data,
      file = "DataCorrected.Rdata"
 )
>
```

C.1.2 Univariate Analysis

The first part of the analysis is the univariate analysis. For each of the covariates in the dataset, it is checked whether it has a significant ($\alpha < 0.1$) on time to 50% improvement.

```
> library(survival)
> library(xtable)
> ##############################
> ## Univariate Hazard Ratios ##
> ##############################
>
> # The first step in the analysis in the article is to calculate the
> # univariate Hazard Ratios of reponse, to see which variables are
> # significant by themselves ( alpha = 0.10).
>
> # The names of all the variables are:
>
> univars <- names(data)[1:49][-c(which(names(data) == "ID"),
                           which(names(data) == "herstelkrap"),
                           which(names(data) =="timekrap_max2jr")
                         )]
> # The results of the univariate Hazard Ratio's will be stored in the
> # following two dataframes. Their rows are given the name of the
> # corresponding variable.
> univarResults <- data.frame(
                   coef = rep(NA, times = length(univars)),
                   expCoef = rep(NA, times = length(univars)),
                   pValue = rep(NA, times = length(univars))
```

```
              )
> row.names(univarResults) <- univars
> # Now, for each of these variables, the univariate Proportional Hazard
> # will be fitted and the results will be stored in the dataframe
> # defined above, the package survival is needed for that.
> j <- 1
> for (i in univars)
 {
   command1 <- paste("univarPH <- coxph(Surv(time = CorrectedTimes, event =
                     herstelkrap) ~ 1 +", i, " , data = data)")

   eval(parse(text=command1))

   # Save results:
   # Fort the factors with more than 2 levels, only the firts
   # coeficient is stored.
   # The likelihoodratio test is stored, so that the complete model is
   # tested, so if a factor with more than 2 levels is significant for
   # only one factor level, it's p-value is still "significant".
   univarResults[j,] = c(univarPH$coefficients[1],
                         exp(univarPH$coefficients[1]),
                         anova(univarPH)[2,4]
                        )

   # update j:
   j <- j + 1
 }
> # The variables with a p-value <= 0.10 are:
> row.names(univarResults[which(univarResults$pValue < 0.10),])
```

C.1.3 Multivariate Analysis

Using backward stepwise removal, the final multivariate model is determined.

```
> ################################
> ## Multivariate Hazard Ratios ##
> ################################
> # All variables which had a p-value <= 0.10 are tested in the
> # mulitvariate model. Furthermore, age, gender and the four
> # dichotomized main diagnostic categories in this study, are forced
> # into the model.
>
> # So the indices of the variables that should be tested in the
> # multivariate model are:
```

```
> multivarsAll <- unique(c(which(row.names(univarResults) ==
                                 "ZLeeftijd"),
                           which(row.names(univarResults) ==
                                 "gendernum"),
                           which(row.names(univarResults) ==
                                 "agorafobie"),
                           which(row.names(univarResults) ==
                                 "paniekagfob"),
                           which(row.names(univarResults) ==
                                 "socfob"),
                           which(row.names(univarResults) ==
                                 "GAS"),
                           which(univarResults$pValue < 0.10)))
> # The names of these variables have to be pasted in one line so that
> # they can be included in the model:
>
> multivars <- ""
> for(k in multivarsAll)
 {
   multivars <- paste(multivars, row.names(univarResults)[k], sep=
                      " + ")
 }
> multivars
> # + ZLeeftijd + gendernum + agorafobie + paniekagfob + socfob + GAS +
> # ZSFpijcorr + ZSFalgcorr + livingsituation3 + education + occupation
> # + ethnicity + alcohol + ZBAS.1 + ZVOLG + ZCOGVER + ZIDENT + ZAFFLAB
> # + ZBEPRIK + ZPASSAG + ZNINTIM + ZANGSTI + ZGEDRAG + ZWANTR + ZSOCONT
> # + ZONHECH + ZZBESUI + singleanx
>
> # Step 1:
> # This string can be used in the formula of the multivariate model:
> formula <-  as.formula(paste(
 "Surv(time = CorrectedTimes, event = herstelkrap) ~ 1",
                              multivars) )
> multivarPH1 <- coxph(formula, data = data, model = T)
> multivarPH1
> # Substep 1: highest p-value for ZSOCONT, candidate for removal
> multivarPH2verg <- drop1(multivarPH1, ~ ZSOCONT, test =  "none")
> multivarPH2verg
> # AIC decreases, so removal is justified
>
> multivarPH2 <- coxph(Surv(time = CorrectedTimes , event = herstelkrap)
                       ~
```

```
                        1 + ZLeeftijd + gendernum + agorafobie +
                        paniekagfob + socfob + GAS + ZSFpijcorr +
                        ZSFalgcorr + livingsituation3 + education +
                        occupation + ethnicity + alcohol + ZBAS.1 +
                        ZVOLG +
                        ZCOGVER + ZIDENT + ZAFFLAB + ZBEPRIK + ZPASSAG
                        + ZNINTIM + ZANGSTI + ZGEDRAG + ZWANTR +
                        ZONHECH + ZZBESUI + singleanx,
                        data = data)
> multivarPH2
> # Substep 2: biggest p-value for ZONHECH and ZANGSTI, both candidates
> # for removal. Start with ZONHECH:
> multivarPH3verg <- drop1(multivarPH2, ~ ZONHECH, test =  "none")
> multivarPH3verg
> # AIC decreases, justified
> multivarPH3 <- coxph(Surv(time = CorrectedTimes , event = herstelkrap)
                       ~
                        1 + ZLeeftijd + gendernum + agorafobie +
                        paniekagfob + socfob + GAS + ZSFpijcorr +
                        ZSFalgcorr + livingsituation3 + education +
                        occupation + ethnicity + alcohol + ZBAS.1 +
                        ZVOLG + ZCOGVER + ZIDENT + ZAFFLAB + ZBEPRIK +
                        ZPASSAG +
                        ZNINTIM + ZANGSTI + ZGEDRAG + ZWANTR +
                        ZZBESUI + singleanx,
                       data = data)
> multivarPH3
> # Substep 4: biggest p-value for ZANGSTI, candidate for removal
> multivarPH4verg <- drop1(multivarPH3, ~ ZANGSTI, test =  "none")
> multivarPH4verg
> #AIC decreases, justified
> multivarPH4 <- coxph(Surv(time = CorrectedTimes , event = herstelkrap)
                       ~
                        1 + ZLeeftijd + gendernum + agorafobie +
                        paniekagfob + socfob + GAS + ZSFpijcorr +
                        ZSFalgcorr + livingsituation3 + education +
                        occupation + ethnicity + alcohol + ZBAS.1 +
                        ZVOLG + ZCOGVER + ZIDENT + ZAFFLAB + ZBEPRIK +
                        ZPASSAG + ZNINTIM + ZGEDRAG + ZWANTR +
                        ZZBESUI + singleanx,
                       data = data)
> multivarPH4
> # Substep 5: biggest p-value for ZZBESUI, candidate for removal
```

```
> multivarPH5verg <- drop1(multivarPH4, ~ ZZBESUI, test =  "none")
> multivarPH5verg
> # AIC decreases, so justified
> multivarPH5 <- coxph(Surv(time = CorrectedTimes , event = herstelkrap)
                       ~
                         1 + ZLeeftijd + gendernum + agorafobie +
                         paniekagfob + socfob + GAS + ZSFpijcorr +
                         ZSFalgcorr + livingsituation3 + education +
                         occupation + ethnicity + alcohol + ZBAS.1 +
                         ZVOLG + ZCOGVER + ZIDENT + ZAFFLAB + ZBEPRIK +
                         ZPASSAG + ZNINTIM + ZGEDRAG + ZWANTR +
                         singleanx,
                       data = data)
> multivarPH5
> # Substep 6: biggest p-value for ZBEPRIK, candidate for removal
> multivarPH6verg <- drop1(multivarPH5, ~ ZBEPRIK, test =  "none")
> multivarPH6verg
> # AIC decreases, justified
> multivarPH6 <- coxph(Surv(time = CorrectedTimes , event = herstelkrap)
                       ~
                         1 + ZLeeftijd + gendernum + agorafobie +
                         paniekagfob + socfob + GAS + ZSFpijcorr +
                         ZSFalgcorr + livingsituation3 + education +
                         occupation + ethnicity + alcohol + ZBAS.1 +
                         ZVOLG +
                         ZCOGVER + ZIDENT + ZAFFLAB + ZPASSAG +
                         ZNINTIM + ZGEDRAG + ZWANTR +
                         singleanx,
                       data = data)
> multivarPH6
> # Substep 7: biggest p-value for ZCOGVER, candidate for removal
> multivarPH7verg <- drop1(multivarPH6, ~ ZCOGVER, test =  "none")
> multivarPH7verg
> # AIC decreases, justified
> multivarPH7 <- coxph(Surv(time = CorrectedTimes , event = herstelkrap)
                       ~
                         1 + ZLeeftijd + gendernum + agorafobie +
                         paniekagfob + socfob + GAS + ZSFpijcorr +
                         ZSFalgcorr + livingsituation3 + education +
                         occupation + ethnicity + alcohol + ZBAS.1 +
                         ZVOLG +
                         ZIDENT + ZAFFLAB + ZPASSAG +
                         ZNINTIM + ZGEDRAG + ZWANTR +
```

```
                        singleanx,
                      data = data)
> multivarPH7
> # Substep 8: biggest p-value for ZWANTR, candidate for removal
> multivarPH8verg <- drop1(multivarPH7, ~ ZWANTR, test =  "none")
> multivarPH8verg
> # AIC decreases, justified
> multivarPH8 <- coxph(Surv(time = CorrectedTimes , event = herstelkrap)
                      ~
                        1 + ZLeeftijd + gendernum + agorafobie +
                        paniekagfob + socfob + GAS + ZSFpijcorr +
                        ZSFalgcorr + livingsituation3 + education +
                        occupation + ethnicity + alcohol + ZBAS.1 +
                        ZVOLG +
                        ZIDENT + ZAFFLAB + ZPASSAG +
                        ZNINTIM + ZGEDRAG +
                        singleanx,
                      data = data)
> multivarPH8
> # Substep 9: biggest p-value for ZVOLG, candidate for removal
> multivarPH9verg <- drop1(multivarPH8, ~ ZVOLG, test =  "none")
> multivarPH9verg
> # AIC decreases, justified
> multivarPH9 <- coxph(Surv(time = CorrectedTimes , event = herstelkrap)
                      ~
                        1 + ZLeeftijd + gendernum + agorafobie +
                        paniekagfob + socfob + GAS + ZSFpijcorr +
                        ZSFalgcorr + livingsituation3 + education +
                        occupation + ethnicity + alcohol + ZBAS.1 +
                        ZIDENT + ZAFFLAB + ZPASSAG +
                        ZNINTIM + ZGEDRAG +
                        singleanx,
                      data = data)
> multivarPH9
> # Substep 10: biggest p-value for ZSFpijcorr, candidate for removal
> multivarPH10verg <- drop1(multivarPH9, ~ ZSFpijcorr, test =  "none")
> multivarPH10verg
> # AIC decreases, so justified
> multivarPH10 <- coxph(Surv(time = CorrectedTimes, event = herstelkrap)
                       ~
                        1 + ZLeeftijd + gendernum + agorafobie +
                        paniekagfob + socfob + GAS +
                        ZSFalgcorr + livingsituation3 + education +
```

```
                        occupation + ethnicity + alcohol + ZBAS.1 +
                        ZIDENT + ZAFFLAB + ZPASSAG +
                        ZNINTIM + ZGEDRAG +
                        singleanx,
                      data = data)
> multivarPH10
> # Substep 11: biggest p-value for ZIDENT, candidate for removal
> multivarPH11verg <- drop1(multivarPH10, ~ ZIDENT, test =  "none")
> multivarPH11verg
> # AIC decreases, so justified
> multivarPH11 <- coxph(Surv(time = CorrectedTimes, event = herstelkrap)
                        ~
                        1 + ZLeeftijd + gendernum + agorafobie +
                        paniekagfob + socfob + GAS +
                        ZSFalgcorr + livingsituation3 + education +
                        occupation + ethnicity + alcohol + ZBAS.1 +
                        ZAFFLAB + ZPASSAG +
                        ZNINTIM + ZGEDRAG +
                        singleanx,
                      data = data)
> multivarPH11
> # Substep 12: biggest p-value for ZNINTIM, candidate for removal
> multivarPH12verg <- drop1(multivarPH11, ~ ZNINTIM, test =  "none")
> multivarPH12verg
> # AIC decreaes (with 0,6), so justified
> multivarPH12 <- coxph(Surv(time = CorrectedTimes, event = herstelkrap)
                        ~
                        1 + ZLeeftijd + gendernum + agorafobie +
                        paniekagfob + socfob + GAS +
                        ZSFalgcorr + livingsituation3 + education +
                        occupation + ethnicity + alcohol + ZBAS.1 +
                        ZAFFLAB + ZPASSAG +
                        ZGEDRAG +
                        singleanx,
                      data = data)
> multivarPH12
> # Substep 13: biggest p-value for singleanx and ZPASSAG, both
> # candidates for removal. Start with singleanx
> multivarPH13verg <- drop1(multivarPH12, ~ singleanx, test =  "none")
> multivarPH13verg
> # AIC increases -> removal not justified.
>
> # for ZPASSAG:
```

```
> multivarPH13bverg <- drop1(multivarPH12, ~ ZPASSAG, test =  "none")
> multivarPH13bverg
> # AIC increases as well, removal not justified.
>
> #################
> ## Final model ##
> #################
>
> # So multivarPH12 is the final model
> # coxph(formula = Surv(time = CorrectedTimes, event = herstelkrap) ~
> # 1 + ZLeeftijd + gendernum + agorafobie + paniekagfob + socfob +
> #  GAS + ZSFalgcorr + livingsituation3 + education + occupation +
> #  ethnicity + alcohol + ZBAS.1 + ZAFFLAB + ZPASSAG + ZGEDRAG +
> #  singleanx, data = data)
> multivarPH12sum <- summary(multivarPH12)
> FinMultVarModelResults <- data.frame(
   expCoef = multivarPH12sum$coefficients[,2],
   lower.95 = exp(multivarPH12sum$coefficients[,1] -
                  qnorm(0.975, 0, 1, lower.tail=T) *
                  multivarPH12sum$coefficients[,3]),
   upper.95 = exp(multivarPH12sum$coefficients[,1] +
                  qnorm(0.975, 0, 1, lower.tail=T) *
                  multivarPH12sum$coefficients[,3]),
   pValue = multivarPH12sum$coefficients[,5]
 )
```

C.1.4 Plot Product-Limit Curves

Product-Limit Curves are made for the final model.

```
> ################################
> ## Product-Limit curves - Fits ##
> ################################
>
> # Now plot Product-Limit curves for the categorical variables that are
> # significant (as in figure 4 in article)
> # First fit the models for each of the categorical variables:
> fitOccupation <- survfit(Surv(CorrectedTimes, herstelkrap) ~
                           occupation, data = data)
> fitEducation <- survfit(Surv(CorrectedTimes, herstelkrap) ~
                          education, data = data)
> fitEthnicity <- survfit(Surv(CorrectedTimes, herstelkrap) ~
                          ethnicity, data = data)
> fitLivingsituation <- survfit(Surv(CorrectedTimes, herstelkrap) ~
```

```
                                    livingsituation3, data = data)
> data$ZGEDRAG_cat <- cut(data$ZGEDRAG, quantile(data$ZGEDRAG,
                                        probs = c(0, 0.368, 0.668, 1)),
                          include.lowest = T,
                          labels = c("Low", "Medium", "High")
                          )
> fitConductProb <- survfit(Surv(CorrectedTimes, herstelkrap) ~
                              ZGEDRAG_cat, data = data)
> data$ZAFFLAB_cat <- cut(data$ZAFFLAB,
                          quantile(data$ZAFFLAB,
                                   probs = c(0, 0.332, 0.658, 1)),
                          include.lowest = T,
                          labels = c("Low", "Medium", "High")
                          )
> fitAffLab <- survfit(Surv(CorrectedTimes, herstelkrap) ~ ZAFFLAB_cat,
                       data = data)
> ###################################
> ## Product-Limit curves - Plots ##
> ###################################
>
> png("SimpleAnalysisKMCurves.png", width = 900, height = 900)
> par(mfrow = c(2,2), lwd=1, las=1, cex=1)
> ## Daily occupation
> plot(fitOccupation, main = "Daily Occupation",
       xlab = "days", ylab = "Survival Probability",
       col = c("red", "blue"),
       mark.time = FALSE,
       ylim = c(0,1)
       )
> legend(legend = c("daily occupation", "no daily occupation"),
         col = c("red", "blue"),
         lty = 1,
         x = "bottomleft",
         bty = "n"
         )
> ## Education
> plot(fitEducation, main = "Education",
       xlab = "days", ylab = "Survival Probability",
       col = c("red", "blue", "green"),
       mark.time = FALSE,
       ylim = c(0,1)
       )
> legend(legend = c("high", "low", "medium"),
```

```
        col = c("red", "blue", "green"),
        lty = 1,
        x = "bottomleft",
        bty = "n"
        )
> ## Ethnicity
> plot(fitEthnicity, main = "Ethnicity",
      xlab = "days", ylab = "Survival Probability",
      col = c("red", "blue"),
      mark.time = FALSE,
      ylim = c(0,1)
      )
> legend(legend = c("Dutch", "Non-Dutch"),
        col = c("red", "blue"),
        lty = 1,
        x = "bottomleft",
        bty = "n"
        )
> ## DAPP-SF conduct problems
> plot(fitConductProb, main = "Conduct Problems",
      xlab = "days", ylab = "Survival Probability",
      col = c("red", "blue", "green"),
      mark.time = FALSE,
      ylim = c(0,1)
     )
> legend(legend = c("Low", "Medium","High"),
        col = c("red", "blue", "green"),
        lty = 1,
        x = "bottomleft",
        bty = "n"
       )
> dev.off()
```

C.2 R Code for Chapter 4

C.2.1 Load and Transform ROM Data

The required package `Icens` is loaded and the interval boundaries for interval censoring are calculated.

```
> ########################################################
> ## Analysing the ROM dataset with interval censoring ##
> ########################################################
>
```

```
> ##########################
> ## Required Libraries ##
> ##########################
>
> library(Icens)

> ########################################################
> ## Change data representation for the Icens package ##
> ########################################################
> data$IntRInf <- NULL
> # Data preparation for Icens package (no NA's, but Inf's)
> for(i in 1:length(data$IntR))
 {
   if(is.na(data$IntR[i]))
   {
     data$IntRInf[i] <- Inf
   }
   else
     data$IntRInf[i] <- data$IntR[i]
 }
```

C.2.2 Estimate Product-Limit Estimators with `ICens`

The Product-Limit estimators for several of the significant covariates in the time to event model from chapter 3 are estimated.

```
> ############################################################
> ## Survival Function estimation with package Icens - Fits ##
> ############################################################
>
> # Occupation
> EMICMnoOccupation<-with(subset(data,occupation==
              "unemployed retired sickleave"),
                         EMICM(cbind(IntL,IntRInf)))
> EMICMOccupation<-with(subset(data,occupation==
             "employed studying primary caregiver children"),
                       EMICM(cbind(IntL,IntRInf)))
> # Education
> EMICMHighEducation <- with(subset(data,education=="high"),
                             EMICM(cbind(IntL,IntRInf)))
> EMICMMediumEducation <- with(subset(data,education=="medium"),
                               EMICM(cbind(IntL,IntRInf)))
> EMICMLowEducation <- with(subset(data,education=="low"),
                            EMICM(cbind(IntL,IntRInf)))
```

```
> # Ethnicity
> EMICMDutch<-with(subset(data,ethnicity=="dutch"),
                   EMICM(cbind(IntL,IntRInf)))
> EMICMNonDutch<-with(subset(data,ethnicity==
   "nondutch parents or self born outside of the netherlands"),
                     EMICM(cbind(IntL,IntRInf)))
> # Conduct Problems
> EMICMConductHigh <- with(subset(data,ZGEDRAG_cat=="High"),
                           EMICM(cbind(IntL,IntRInf)))
> EMICMConductMedium <- with(subset(data,ZGEDRAG_cat=="Medium"),
                             EMICM(cbind(IntL,IntRInf)))
> EMICMConductLow <- with(subset(data,ZGEDRAG_cat=="Low"),
                          EMICM(cbind(IntL,IntRInf)))
>
```

Plot the Product-Limit survival curves.

```
> ################################################
> ## Cumulative Hazard Curves with package Icens ##
> ################################################
> png("IntervalAnalysisKMCurves.png", width = 900, height = 900)
> par(mfrow = c(2,2), lwd=1, las=1, cex=1)
> # Colours cannot be specified for plot.icsurv, so use different
> # line types
>
> # Occupation
> plot(EMICMnoOccupation, type = "lc", surv=T, xlim = c(0,730),
      main = "Daily Occupation", xlab = "days",
      ylab = "Survival Probability",
      ylim = c(0,1),
      shade = 1
         )
> plot(EMICMOccupation, type = "lc", surv= T, lty = 3, new = FALSE,
      shade =1)
> legend('bottomleft', c('No daily occupation', 'Daily occupation'),
        lty=c(1,3), lwd=1, bty = "n")
> # Education
> plot(EMICMHighEducation, type = "lc", surv=T, xlim = c(0,730),
      main = "Education", xlab = "days",
      ylab = "Survival Probability",
      ylim = c(0,1),
      shade = 1
     )
> plot(EMICMMediumEducation, type = "lc", surv= T, lty = 3, new = FALSE,
```

```
      shade = 1)
> plot(EMICMLowEducation, type = "lc", surv= T, lty = 4, new = FALSE,
      shade = 1)
> legend('bottomleft', c('High', 'Medium', 'Low'),
        lty=c(1,3,4), lwd=1, bty = "n")
> # Ethnicity
> plot(EMICMDutch, type = "lc", surv=T, xlim = c(0,730),
      main = "Ethnicity", xlab = "days",
      ylab = "Survival Probability",
      ylim = c(0,1),
      shade = 1
 )
> plot(EMICMNonDutch, type = "lc", surv= T, lty = 3, new = FALSE,
      shade =1)
> legend('bottomleft', c('Dutch', 'Non-Dutch'),
        lty=c(1,3), lwd=1, bty = "n")
> # Conduct Problems
> plot(EMICMConductHigh, type = "lc", surv=T, xlim = c(0,730),
      main = "Conduct Problems", xlab = "days",
      ylab = "Survival Probability",
      ylim = c(0,1),
      shade = 1
      )
> plot(EMICMConductMedium, type = "lc", surv= T, lty = 3, new = FALSE,
      shade = 1)
> plot(EMICMConductLow, type = "lc", surv= T, lty = 4, new = FALSE,
      shade = 1)
> legend('bottomleft', c('High', 'Medium', 'Low'),
        lty=c(1,3,4), lwd=1, bty = "n")
> dev.off()
```

C.2.3 The Bootstrap Procedure

The functions `intcox.boot` and `intcox.bootstrapping` for the bootstrap procedure are defined as follows.

```
> ######################################################################
> ## Function for bootstrap procedure to find confidence intervals ##
> ## for coefficients found by intcox.                              ##
> ######################################################################
>
>
> # Function to perform one intcox estimation
> intcox.boot <- function(i, formula, data)
```

```
 {
   # input:
   # i = unused integer
   # formula = formula to be implemented in the intcox function,
   #           Surv object (type = interval2) ~ covariates.
   # data = data that includes the left and right boundaries of the
   #        censoring intervals, together with the covariates.
   #
   # output:
   # coef = parameter estimates for the Cox PH for interval censored
   #        data, estimated with the function intcox.
   # term = indicator for the reason of termination,
   #         1 - algorithm converged
   #         2 - no improvement of likelihood possible, the iteration
   #             number is shown
   #         3 - algorithm did not converge - maximum number of
   #             iteration reached
   #         4 - inside precondition(s) are not fulfilled at this
   #             iteration

   # Take bootstrap sample
   boot.sample <- sample(1:nrow(data), size = nrow(data),
                         replace = T)
   data.sample <- data[boot.sample,]

   # Estimate regression parameters for bootstrap sample
   boot.fit <- intcox(formula, data = data.sample, no.warnings = T)

   # Return results
   return(list(coef = coefficients(boot.fit),
               term = boot.fit$termination
               )
          )
 }

> ######################################################################
> ## Function to perform whole bootstrap to estimate CIs for the ##
> ## parameters of the Cox PH model for interval censored data,  ##
> ## using intcox                                                ##
> ######################################################################
>
> intcox.bootstrapping <- function(formula, data, n.iter = 1000,
                                   alpha = 0.05)
 {
```

```
# input:
# formula = formula to be implemented in the intcox function,
#           Surv object (type = interval2) ~ covariates.
# data = data that includes the left and right boundaries of the
#        censoring intervals, together with the covariates.
# n.iter = number of iterations in the bootstrap, default = 1000
# alpha = significance level, default = 0.05, corresponding to a
#         95% CI
#
# output:
# CI.coefficients = upper and lower bound of the confidence
#                   interval of the parameter estimates for each
#                   of the covariates.
# CI.HR = upper and lower bound of the confidence interval of
#         the Hazard Ratios for each of the covariates.
# terminationtypes = vector of indicators for the reason of
#                    termination, for each of the bootstrap
#                    replications.

# Load the intcox-library
library(intcox)

# Apply intcox to n.iter samples from the data, using the
# intcox.boot function. Save output in "boot".
boot <- lapply(1:n.iter, intcox.boot, formula = formula,
               data = data)
cov.names <- names(boot[[1]]$coef)
boot <- matrix(unlist(boot), byrow = T, nrow = n.iter)
colnames(boot) <- c(cov.names, "termination")

# Calculate the CI bounds of the coefficients and the Hazard
# ratios and store them in CICoef and CIHR. Save the termination
# types as well.
CICoef <- data.frame(
  CICoefLower = rep(NA, times = length(cov.names)),
  CICoefUpper = rep(NA, times = length(cov.names))
)

CIHR <- data.frame(
  CIHRLower = rep(NA, times = length(cov.names)),
  CIHRUpper = rep(NA, times = length(cov.names))
)
```

```
  terminationtypes <- boot[,"termination"]

  row.names(CICoef) <- row.names(CIHR) <- cov.names

  # The CI bounds can be found at the following positions
  k <- floor((n.iter + 1)*(alpha/2))
  pos.lower <- k
  pos.upper <- n.iter + 1 - k

  # Find the bounds for each of the covariates in the model
  for(i in 1:length(cov.names))
  {
    cov <- cov.names[i]
    cov.ord <- order(boot[,cov])

    CICoef$CICoefLower[i] <- boot[cov.ord, cov][pos.lower]
    CICoef$CICoefUpper[i] <- boot[cov.ord, cov][pos.upper]
    CIHR$CIHRLower[i] <- exp(CICoef$CICoefLower[i])
    CIHR$CIHRUpper[i] <- exp(CICoef$CICoefUpper[i])
  }

  return(list(CI.coefficients = CICoef,
              CI.HR = CIHR,
              terminationtypes = terminationtypes))
}
```

C.3 R Code for Chapter 5

The implementation of Equation 5.4.3 in R.

```
> survIPCW <- function(Tstart, Tstop, status, weights)
 {
   ord <- order(Tstart)
   Tstart <- Tstart[ord]
   Tstop <- Tstop[ord]
   status <- status[ord]
   weights <- weights[ord]

   tout <- Tstop[status==1]
   tout <- unique(tout)

   nout <- length(tout)
```

```
  di <- Yi <- rep(NA, nout)
  for (i in 1:nout)
  {
    di[i] <- sum((Tstop==tout[i] & status==1)*weights)

    Yi[i] <- sum((Tstop == tout[i])*weights)
  }

  surv <- cumprod(1 - di/Yi)

  return(data.frame(time=tout, surv=surv))
}
```

C.4 R Code for Chapter 6

C.4.1 IPCW on Test Data

Load test data.

```
> library(xtable)
> # Make the test data:
> test <- data.frame(id=1:6,
    time=c(18,23,27,32,57,64),
    status=c(1,1,0,1,0,1),
    cov= c(1,2,1,2,3,2))
> xtable(test)
```

Transformation of the test dataset into the long format (counting data).

```
> # Step 1: Transform test data into long format
> library(survival)
> test$Tstart <- 0
> times <- sort(unique(test$time[test$censored==1 || test$status == 1]))
> test$censored <- 1-test$status
> # Split data over all event en censoring times
> test$Tstart <- 0
> test.long <- survSplit(test, cut=times, end="time", start="Tstart",
                         event="status", id = "id")
> test.long <- test.long[order(test.long$id,test.long$time),]
> test.long.cens <- survSplit(test, cut=times, end="time", start="Tstart",
                              event="censored", id = "id")
> test.long.cens <- test.long.cens[order(test.long.cens$id,
                                         test.long.cens$time),]
> test.long$censored <- test.long.cens$censored
> test.long$id <- as.numeric(test.long$id)
```

```
> # Put columns in better order
> test.long <- test.long[,c(1,5,2,3,6,4)]
> # Give columns correct names:
> colnames(test.long) <- c("id", "Tstart", "Tstop", "status", "censored",
                           "cov")
> xtable(test.long, digits = 0)
```

Fit censoring models with `coxph`, one model without covariates and one with covariates.

```
> C0 <- coxph(Surv(Tstart, Tstop, censored) ~ 1,   data=test.long)
> CZ <- coxph(Surv(Tstart, Tstop, censored) ~ cov, data=test.long)
```

Calculate the survival probabilities for the model without covariates.

```
> C0fit <- summary(survfit(C0), times = test.long$Tstart)
> test.long$K0ti <- C0fit$surv
```

Calculate the survival probabilities for the model including covariates.

```
> test.long$KZti <- NULL
> for(i in 1:nrow(test.long))
 {
   datai <- test.long[i,]

   sfiCZ <- survfit(CZ, newdata = datai)
   ssfiCZ <- summary(sfiCZ, times = datai$Tstart)
   test.long$KZti[i] <- ssfiCZ$surv
 }
```

Calculate unstabilized and stabilized IPCW weights.

```
> test.long$WUnStab <- 1/test.long$KZti
> test.long$WStab <- test.long$K0ti/test.long$KZti
```

Show results for the test data.

```
> xtable(test.long, digits = c(0,0,0,0,0,0,0,2,2,2,2))
```

Estimate the survival model for time to event with the Product-Limit estimator.

```
> # Survival without IPCW weigths:
> res <- survfit(Surv(Tstart,Tstop,status) ~ 1,
                 data=test.long)
> # survival with IPCW stabilized weights:
> resWStab <- survfit(Surv(Tstart,Tstop,status) ~ 1,
                      data=test.long, weights=WStab)
> # survival with IPCW unstabilized weights:
> resWUnStab <- survfit(Surv(Tstart,Tstop,status) ~ 1,
                        data=test.long, weights=WUnStab)
```

Check survival results with the implementation of Equation 5.4.3.

```
> survIPCW(Tstart = test.long$Tstart,
           Tstop = test.long$Tstop,
           status = test.long$status,
           weights = test.long$WStab) # unstabilized weights
> # Same results as on paper and in resWStab:
> cbind(resWStab$time, resWStab$surv)
> survIPCW(Tstart = test.long$Tstart,
           Tstop = test.long$Tstop,
           status = test.long$status,
           weights = test.long$WUnStab) # stabilized weights
> # Same results as on paper and in resW2:
> cbind(resWUnStab$time, resWUnStab$surv)
```

Estimate the survival probabilities with `coxph`.

```
> # Original analysis without IPCW
> CoxW0 <- survfit(coxph(Surv(Tstop,status) ~ cov,
                  data=test.long))
> CoxW0$surv
> # IPCW with stabilized weights
> CoxWStab <- survfit(coxph(Surv(Tstart,Tstop,status) ~ cov,
                           data=test.long, weights = WStab))
> CoxWStab$surv
> # IPCW with unstabilized weights
> CoxWUnStab <- survfit(coxph(Surv(Tstart,Tstop,status) ~ cov,
                             data=test.long, weights = WUnStab))
> CoxWUnStab$surv
```

C.4.2 IPCW on Two Datasets

The test dataset with independent censoring.

```
> library(xtable)
> library(survival)
> library(tcltk)
> test.NoDepCens <- data.frame(id=1:20,
     time=c(18,23,27,32,57,64,65,68,70,73,75,78,80,82,83,84,86,88,91,92),
     status=c(1,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,1,1),
     cov=   c(1,2,1,2,1,1,2,3,2,1,1,2,2,2,1,2,3,1,1,1))
> table(test.NoDepCens$status, test.NoDepCens$cov)
> xtable(test.NoDepCens, digits = 0)
```

The test dataset with dependent censoring.

```
> test.DepCens <- data.frame(id=1:20,
     time=c(18,23,27,32,57,64,65,68,70,73,75,78,80,82,83,84,86,88,91,92),
     status=c(1,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,1,1),
     cov=   c(1,1,2,1,2,3,2,1,2,1,2,1,2,3,2,1,2,1,1,1))
> xtable(test.DepCens, digits = 0)
```

Perform IPCW on the test dataset with independent censoring and plot resulting Product-Limit curve.

```
> library(xtable)
> library(survival)
> library(tcltk)
> # Step 1: Transform test.NoDepCens data into long format
> test.NoDepCens$Tstart <- 0
> times <- sort(unique(test.NoDepCens$time[test.NoDepCens$censored==1 ||
                                             test.NoDepCens$status == 1]))
> test.NoDepCens$censored <- 1-test.NoDepCens$status
> # Split data over all event en censoring times
> test.NoDepCens$Tstart <- 0
> test.NoDepCens.long <- survSplit(test.NoDepCens, cut=times, end="time",
                                   start="Tstart", event="status",
                                   id = "id")
> test.NoDepCens.long <- test.NoDepCens.long[
         order(test.NoDepCens.long$id,
               test.NoDepCens.long$time),]
> test.NoDepCens.long.cens <- survSplit(test.NoDepCens, cut=times,
                                        end="time", start="Tstart",
                                        event="censored", id = "id")
> test.NoDepCens.long.cens <- test.NoDepCens.long.cens[order(
   test.NoDepCens.long.cens$id,test.NoDepCens.long.cens$time),]
> test.NoDepCens.long$censored <- test.NoDepCens.long.cens$censored
> test.NoDepCens.long$id <- as.numeric(test.NoDepCens.long$id)
> # Put columns in better order
> test.NoDepCens.long <- test.NoDepCens.long[,c(1,5,2,3,6,4)]
> # Give columns correct names:
> colnames(test.NoDepCens.long) <- c("id", "Tstart", "Tstop", "status",
                                     "censored", "cov")
> xtable(test.NoDepCens.long, digits = 0)
> ######################################################################
> # Step 2: Fit censoring models C0NoDepCens and CZNoDepCensNoDepCens and
> # calculate the Product-Limit estimators
>
> # Fit models
> C0NoDepCens <- survfit(Surv(Tstart, Tstop, censored) ~ 1,
```

```
                        data=test.NoDepCens.long)
> CZNoDepCens <- coxph(Surv(Tstart, Tstop, censored ) ~ cov,
                       data=test.NoDepCens.long)
> # Calculate Product-Limit estimators for model C0NoDepCens, K0ti:
> C0NoDepCensfit <- summary(C0NoDepCens, times = test.NoDepCens.long$Tstart)
> test.NoDepCens.long$K0ti <- C0NoDepCensfit$surv
> # Calculate the Product-Limit estimators for model CZNoDepCens, KZti
>
> pb <- tkProgressBar(title = "Working hard:", min = 0,
                      max = nrow(test.NoDepCens.long), width = 300)
> test.NoDepCens.long$KZti <- NULL
> for(i in 1:nrow(test.NoDepCens.long))
 {
   setTkProgressBar(pb, i, title = "% ready!")

   datai <- test.NoDepCens.long[i,]

   sfiCZNoDepCens <- survfit(CZNoDepCens, newdata = datai)
   ssfiCZNoDepCens <- summary(sfiCZNoDepCens, times = datai$Tstart)
   test.NoDepCens.long$KZti[i] <- ssfiCZNoDepCens$surv
 }
> close(pb)
> ##########################################################################
> # Step 3: Calculate IPCW weights:
>
> test.NoDepCens.long$WUnStab <- 1/test.NoDepCens.long$KZti
> test.NoDepCens.long$WStab <- test.NoDepCens.long$K0ti/
                                 test.NoDepCens.long$KZti
> ##########################################################################
> # Step 4: Fit the survival model for time to event in the absence of
> # censoring
>
> # Product-Limit estimator:
> # Survival without IPCW weights:
> resNoDepCens <- survfit(Surv(Tstart,Tstop,status) ~ 1,
                          data=test.NoDepCens.long)
> # survival with IPCW stabilized weights:
> resNoDepCensWStab <- survfit(Surv(Tstart,Tstop,status) ~ 1,
                               data=test.NoDepCens.long, weights=WStab)
> # survival with IPCW unstabilized weights:
> resNoDepCensWUnStab <- survfit(Surv(Tstart,Tstop,status) ~ 1,
                                 data=test.NoDepCens.long, weights=WUnStab)
> # Plot the curves in one figure:
```

```
> png("SurvivalPlotsNoDepCens.png", width = 400, height = 400)
> plot(resNoDepCens$time, resNoDepCens$surv, col="black",
      lty=1, type = "s", ylim = c(0,1), xlab = "Time (days)",
      ylab = "Survival Probability", lwd = 2, cex.lab = 1
      )
> lines(resNoDepCensWStab$time, resNoDepCensWStab$surv, col="red",lty=1,
      new = FALSE,
      type = "s", lwd = 2)
> legend(legend = c("No IPCW", "IPCW (Un)stabilized Weights"),
        col = c("black", "red"),
        lty = 1,
        x = "bottomleft",
        cex = 1,
        bty = "n",
        lwd = 1
 )
> dev.off()
```

Perform IPCW on the test dataset with dependent censoring and plot resulting Product-Limit curve.

```
> library(xtable)
> library(survival)
> library(tcltk)
> # Step 1: Transform test.DepCens data into long format
> library(survival)
> test.DepCens$Tstart <- 0
> times <- sort(unique(test.DepCens$time[test.DepCens$censored==1 ||
                                          test.DepCens$status == 1]))
> test.DepCens$censored <- 1-test.DepCens$status
> # Split data over all event en censoring times
> test.DepCens$Tstart <- 0
> test.DepCens.long <- survSplit(test.DepCens, cut=times, end="time",
                                 start="Tstart", event="status",
                                 id = "id")
> test.DepCens.long <- test.DepCens.long[order(test.DepCens.long$id,
                                               test.DepCens.long$time),]
> test.DepCens.long.cens <- survSplit(test.DepCens, cut=times, end="time",
                                      start="Tstart", event="censored",
                                      id = "id")
> test.DepCens.long.cens <- test.DepCens.long.cens[order(
   test.DepCens.long.cens$id,test.DepCens.long.cens$time),]
> test.DepCens.long$censored <- test.DepCens.long.cens$censored
> test.DepCens.long$id <- as.numeric(test.DepCens.long$id)
```

```
> # Put columns in better order
> test.DepCens.long <- test.DepCens.long[,c(1,5,2,3,6,4)]
> # Give columns correct names:
> colnames(test.DepCens.long) <- c("id", "Tstart", "Tstop", "status",
                                   "censored", "cov")
> xtable(test.DepCens.long, digits = 0)
> ######################################################################
> # Step 2: Fit censoring models C0DepCens and CZ and calculate the Product-
> # Limit estimators
>
> # Fit models
> C0DepCens <- survfit(Surv(Tstart, Tstop, censored) ~ 1,
                       data=test.DepCens.long)
> CZDepCens <- coxph(Surv(Tstart, Tstop, censored ) ~ cov,
                     data=test.DepCens.long)
> # Calculate Product-Limit estimators for model C0, K0ti:
> C0DepCensfit <- summary(C0DepCens, times = test.DepCens.long$Tstart)
> test.DepCens.long$K0ti <- C0DepCensfit$surv
> # Calculate the Product-Limit estimators for model CZ, KZti
>
> library(tcltk)
> pb <- tkProgressBar(title = "Working hard:", min = 0,
                      max = nrow(test.DepCens.long), width = 300)
> test.DepCens.long$KZti <- NULL
> for(i in 1:nrow(test.DepCens.long))
 {
   setTkProgressBar(pb, i, title = "% ready!")

   datai <- test.DepCens.long[i,]

   sfiCZDepCens <- survfit(CZDepCens, newdata = datai)
   ssfiCZDepCens <- summary(sfiCZDepCens, times = datai$Tstart)
   test.DepCens.long$KZti[i] <- ssfiCZDepCens$surv
 }
> close(pb)
> ######################################################################
> # Step 3: Calculate IPCW weights:
>
> test.DepCens.long$WUnStab <- 1/test.DepCens.long$KZti
> test.DepCens.long$WStab <- test.DepCens.long$K0ti/
                             test.DepCens.long$KZti
> ######################################################################
> # Step 4: Fit the survival model for time to event in the absence of
```

```
> # censoring
>
> # Product-Limit estimator:
> # Survival without IPCW weights:
> resDepCens <- survfit(Surv(Tstart,Tstop,status) ~ 1,
                        data=test.DepCens.long)
> # survival with IPCW stabilized weights:
> resDepCensWStab <- survfit(Surv(Tstart,Tstop,status) ~ 1,
                             data=test.DepCens.long, weights=WStab)
> # survival with IPCW unstabilized weights:
> resDepCensWUnStab <- survfit(Surv(Tstart,Tstop,status) ~ 1,
                               data=test.DepCens.long, weights=WUnStab)
> # Plot the curves in one figure:
> png("SurvivalPlotsDepCens.png", width = 400, height = 400)
> plot(resDepCens$time, resDepCens$surv, col="black",
      lty=1, type = "s", ylim = c(0,1), xlab = "Time (days)",
      ylab = "Survival Probability", cex.lab = 1,
      lwd = 2)
> lines(resDepCensWStab$time, resDepCensWStab$surv, col="red",lty=1,
       type = "s",
       lwd = 2)
> legend(legend = c("No IPCW", "IPCW (Un)stabilized Weights"),
        col = c("black", "red"),
        lty = 1,
        x = "bottomleft",
        cex = 1,
        bty = "n",
        lwd = 2
 )
> dev.off()
```

C.4.3 IPCW on ROM Data

Fit time to censoring model

```
> # Make a variable which indicates whether a subject was censored or not:
> data$censored <- as.numeric(!data$herstelkrap)
> # Estimate survival curve (Product-Limit) for time to
> # censoring
> survCens <- survfit(Surv(time = CorrectedTimes , event = censored) ~ 1,
                      data = data)
```

Plot result.

```
> png("KMCurveCens.png", width = 400, height = 300)
> plot(survCens, conf.int= FALSE,
```

```
      xlab = "days", ylab = "Survival Probability",
      mark.time = FALSE,
      ylim = c(0,1)
 )
> dev.off()
```

Transform Data to long format.

```
> # Event Times (event and censoring times)
> eTimes <- sort(unique(data$CorrectedTimes))
> # Measurement times of BAS and BSI:
> mTimesCol <- grep("ObsTIME.",names(data), fixed = T)
> mTimes <- NULL
> for(i in mTimesCol)
 {
   mTimes <- c(mTimes, data[,i])
 }
> mTimes <- sort(unique(mTimes))
> # All censoring times and measurementTimes are:
> allTimes <- sort(unique(c(eTimes, mTimes)))
> # Identify the columns in which the BAS and BSI-12 measurements are
> # stored:
> BASmCol <- grep("BAS.",names(data), fixed = T)[-1]
> BSImCol <- grep("BSI_12.",names(data), fixed = T)[-16]
> # For each subject, make splitted time intervals
> data.long <- survSplit(data, cut=allTimes, start="Tstart",
                         end="CorrectedTimes", event="censored")
> data.long.herstel <- survSplit(data, cut=allTimes, start="Tstart",
                                 end="CorrectedTimes", event="herstelkrap")
> # sort the data by subject and times:
> data.long <- data.long[order(data.long$ID,data.long$Tstart),]
> data.long.herstel <- data.long.herstel[order(data.long.herstel$ID,
                                               data.long.herstel$Tstart),]
> # Put in the correct herstelkrap indicators:
> data.long$herstelkrap <- data.long.herstel$herstelkrap
> ptm <- proc.time()
> # From which visit should the measurements be taken?
> data.long$Visit <- apply(data.long, 1, function(x) {
   sum(as.numeric(as.numeric(x["Tstart"]) >= as.numeric(x[mTimesCol])),
       na.rm = T)
 }
 )
> data.long$BAStd <- apply(data.long, 1, function(x) {
   as.numeric(x[BASmCol[as.numeric(x["Visit"])]])
```

```
 })
> data.long$BSItd <- apply(data.long, 1, function(x) {
   as.numeric(x[BSImCol[as.numeric(x["Visit"])]])
 })
> proc.time() - ptm
> ########################################
>
> # Calculate Z-values for time-dependent ZBAS en ZBSI-12.
> # Use the mean and sd from the baseline BAS and BSI-12 measurements:
>
> data.long$ZBSItd <- (data.long$BSItd - mean(data$BSI_12.1))/
   sd(data$BSI_12.1)
> data.long$ZBAStd <- (data.long$BAStd - mean(data$BAS.1))/
   sd(data$BAS.1)
> # Remove short format:
> remove(setdiff(ls(), "data.long"))
> save(data.long,

      file = ".\\DataLong.Rdata"
 )
```

Load data (included sucht that the dataset does not have to be transformed in each session.)

```
> load("./DataLong.Rdata")
```

Fit censoring model without covariates and calculate the survival probabilities.

```
> ###############################################
> ## Fit survival model for time to censoring ##
> ###############################################
> C0 <- coxph(Surv(Tstart, CorrectedTimes, censored) ~ 1, data=data.long)
> ################################################################
> ## Calculate Product-Limit estimators for model C0, the K0ti ##
> ################################################################
> C0fit <- summary(survfit(C0), times = data.long$Tstart)
> data.long$K0ti <- C0fit$surv
> save(data.long,

      file = ".\\DataLongIPCW.Rdata"
 )
```

Model 1: Time Dependent Covariates

Fit censoring model.

```
> # Application of IPCW with censoring model which includes only time
> # dependent covariates (ZBAStd and ZBSItd)
>
> library(survival)
> library(tcltk) # for the progress bar
> ########################################
> ## Loading and transforming the data ##
> ########################################
>
> # The data in long format, together with the Product-Limit estimators for
> # the time to censoring model without covariates are available in the
> # following file:
> load("C:\\Users\\Sanne\\Documents\\School\\LU - MA2\\Scriptie\\Data\\DataLongIPCW.Rd
> ######################################################################
> ## Fit the censoring model for Model 1 with only time dependent   ##
> ## covariates                                                     ##
> ######################################################################
>
> # Model type 1 includes all variables that are time dependent
> set.seed(123)
> censModTDep1 <- coxph(Surv(time = Tstart, time2 = CorrectedTimes ,
                             event = censored) ~ ZBAStd + ZBSItd,
                        data = data.long)
> # Geen significante effecten.
>
> censModTDep1Sum <- summary(censModTDep1)
> censModTDep1Res <- data.frame(
   expCoef = censModTDep1Sum$coefficients[,2],
   lower.95 = exp(censModTDep1Sum$coefficients[,1] -
                    qnorm(0.975, 0, 1, lower.tail=T) *
                    censModTDep1Sum$coefficients[,3]),
   upper.95 = exp(censModTDep1Sum$coefficients[,1] +
                    qnorm(0.975, 0, 1, lower.tail=T) *
                    censModTDep1Sum$coefficients[,3]),
   pValue = censModTDep1Sum$coefficients[,5]
 )
> library(xtable)
> xtable(censModTDep1Res, digits = 3)
```

Calculate IPCW weights.

```
> ##############################################
> ## Application of IPCW method to ROM dataset ##
> ##############################################
```

```
> data.long.TDEP.1 <- data.long
> # Calculate the survival estimates for time to censoring for the CZ model:
> # models:
> CZ <- censModTDep1
> # Seperate data for each ID, then calculate KZti per subject
> # at each timepoint for that subject:
> # calculate the estimates for each of the individuals:
> IDs <- unique(data.long.TDEP.1$ID)
> pb <- tkProgressBar(title = "Working hard:", min = 0,
                      max = nrow(data.long.TDEP.1), width = 300)
>  K1ti <- NULL
> for(i in IDs)
 {
   setTkProgressBar(pb, i, title = "% ready!")

   datai <- subset(data.long.TDEP.1, ID == i)

   pbsmall <- tkProgressBar(title = "% ready with subject i", min = 0,
                       max = nrow(datai), width = 300)
   for(j in 1:nrow(datai))
   {
     setTkProgressBar(pbsmall, j, title = "% ready with subject i!")

     # Fit C1
     sfiCZ <- survfit(CZ, newdata = datai[j,])
     ssfiCZ <- summary(sfiCZ, times = datai[j,]$Tstart)
     KZti <- c(KZti, ssfiCZ$surv)
   }

 }
> close(pb)
> data.long.TDEP.1$KZti <- KZti
> ################################################################
>
> # Calculate weights:
> data.long.TDEP.1$WUnStab <- 1/data.long.TDEP.1$K1ti
> data.long.TDEP.1$WStab <- data.long.TDEP.1$K0ti/data.long.TDEP.1$K1ti
> save(data.long.TDEP.1,
      file = ".\\DataLongTDEP1.Rdata")
```

Load data result

```
> # Calculations were done by the computer of the LUMC. Results are
> # stored in:
> load(".\\DataLongTDEP1.Rdata")
```

Plot Product-Limit curves.

```
> ##########################################################################
> # Now compare survival curves:
> survBaselineNoIPCW <- survfit(Surv(Tstart,CorrectedTimes,herstelkrap) ~ 1,
                                data=data.long.TDEP.1)
> survIPCWTINDEP1Stab <- survfit(Surv(Tstart,CorrectedTimes,herstelkrap) ~ 1,
                                 data=data.long.TDEP.1, weights=WStab)
> survIPCWTINDEP1UnStab <- survfit(Surv(Tstart,CorrectedTimes,herstelkrap) ~ 1,
                                   data=data.long.TDEP.1, weights=WUnStab)
> png("KMCurvesMod1TimeDepIPCW.png", width = 900, height = 600)
> plot(survBaselineNoIPCW$time, survBaselineNoIPCW$surv, col="black",
      lty=1, type = "l", ylim = c(0,1), xlab = "Time (days)",
      ylab = "Survival Probability")
> lines(survIPCWTINDEP1Stab$time, survIPCWTINDEP1Stab$surv, col="red",lty=1)
> legend(legend = c("No IPCW, S(t)", "IPCW (Un)stabilized Weights, S[IPCW]"),
         col = c("black", "red"),
         lty = 1,
         x = "bottomleft",
         cex = 1.5,
         bty = "n"
 )
> dev.off()
```

Fit Cox models.

```
> ##########################################################################
> # Now Cox model
>
> CoxIPCWTDEP1WStabTrunc <- coxph(Surv(Tstart, CorrectedTimes,
                                       event = herstelkrap) ~
                                    1 + ZLeeftijd + gendernum + agorafobie +
                                  paniekagfob + socfob + GAS + ZSFalgcorr +
                                  livingsituation3 + education + occupation +
                                  ethnicity + alcohol + ZBAS.1 + ZAFFLAB +
                                  ZPASSAG + ZGEDRAG + singleanx,
                                 data = data.long.TDEP.1,
                                  weights = WStabTrunc)
> CoxIPCWTDEP1WStabTruncSUM <- summary(CoxIPCWTDEP1WStabTrunc)
> CoxIPCWTDEP1WStabTruncRES <- data.frame(
   HR = CoxIPCWTDEP1WStabTruncSUM$coefficients[,2],
   lower.95 = exp(CoxIPCWTDEP1WStabTruncSUM$coefficients[,1] -
                    qnorm(0.975, 0, 1, lower.tail=T) *
                     CoxIPCWTDEP1WStabTruncSUM$coefficients[,3]),
   upper.95 = exp(CoxIPCWTDEP1WStabTruncSUM$coefficients[,1] +
```

```
                    qnorm(0.975, 0, 1, lower.tail=T) *
                     CoxIPCWTDEP1WStabTruncSUM$coefficients[,3])
 )
> # Unstabilized weights
> CoxIPCWTDEP1WUnStab <- coxph(Surv(Tstart, CorrectedTimes,
                                    event = herstelkrap) ~
                                   1 + ZLeeftijd + gendernum + agorafobie +
                                 paniekagfob + socfob +
                                   GAS + ZSFalgcorr + livingsituation3 +
                                 education + occupation +
                                   ethnicity + alcohol + ZBAS.1 + ZAFFLAB +
                                 ZPASSAG + ZGEDRAG +
                                   singleanx, data = data.long.TDEP.1,
                                 weights = WUnStab)
> CoxIPCWTDEPWUnStabSUM <- summary(CoxIPCWTDEP1WUnStab)
> CoxIPCWTDEP1WUnStabRES <- data.frame(
   HR = CoxIPCWTDEPWUnStabSUM$coefficients[,2],
   lower.95 = exp(CoxIPCWTDEPWUnStabSUM$coefficients[,1] -
                    qnorm(0.975, 0, 1, lower.tail=T) *
                    CoxIPCWTDEPWUnStabSUM$coefficients[,3]),
   upper.95 = exp(CoxIPCWTDEPWUnStabSUM$coefficients[,1] +
                    qnorm(0.975, 0, 1, lower.tail=T) *
                    CoxIPCWTDEPWUnStabSUM$coefficients[,3])
 )
```

Model 2: Baseline Covariates with a Significant Influence on the Time to Failure

Fit censoring model.

```
> library(survival)
> library(tcltk) # for the progress bar
> ########################################
> ## Loading and transforming the data ##
> ########################################
>
> # The data in long format, together with the Product-Limit estimators for
> # the time to censoring model without covariates are available in the
> # following file:
> load("C:\\Users\\Sanne\\Documents\\School\\LU - MA2\\Scriptie\\Data\\DataLongIPCW.Rdata")
> ######################################################################
> ## Fit the censoring model for Model 1 with only time independent ##
> ## covariates                                                       ##
> ######################################################################
```

```
>
> # Model type 2 includes all variables that are included in the time-to-
> # failure model.
> set.seed(123)
> censModTIndep1 <- coxph(Surv(time = Tstart, time2 = CorrectedTimes ,
                               event = censored) ~
                          1 + ZLeeftijd + gendernum + agorafobie +
                          paniekagfob + socfob + GAS +
                          ZSFalgcorr + livingsituation3 + education +
                          occupation + ethnicity + alcohol + ZBAS.1 +
                          ZAFFLAB + ZPASSAG +
                          ZGEDRAG +
                          singleanx,
                        data = data.long)
> censModTIndep1Sum <- summary(censModTIndep1)
> censModTIndep1Res <- data.frame(
   expCoef = censModTIndep1Sum$coefficients[,2],
   lower.95 = exp(censModTIndep1Sum$coefficients[,1] -
                  qnorm(0.975, 0, 1, lower.tail=T) *
                  censModTIndep1Sum$coefficients[,3]),
   upper.95 = exp(censModTIndep1Sum$coefficients[,1] +
                  qnorm(0.975, 0, 1, lower.tail=T) *
                  censModTIndep1Sum$coefficients[,3]),
   pValue = censModTIndep1Sum$coefficients[,5]
 )
> library(xtable)
> xtable(censModTIndep1Res, digits = 3)
```

Calculate IPCW weights.

```
> ################################################################
> ## Fit censoring model CZ and calculate the Product- Limit ##
> ## estimators, the KZti.                                      ##
> ################################################################
> data.long.TINDEP.1 <- data.long
> # The CZ model:
> CZ <- censModTIndep1
> # Calculate the Product-Limit estimators for model CZ, the KZti
> # calculate the estimates for each of the individuals. Since the model
> # only includes time independent variables, the survival curve does not
> # have to be fit for every line, but can be fit for each subject.
> IDs <- unique(data.long.TINDEP.1$ID)
> pb <- tkProgressBar(title = "Working hard:", min = 0, max = max(IDs),
                      width = 300)
```

```
> KZti <- NULL
> for(i in IDs)
 {
   setTkProgressBar(pb, i, title = "% ready!")

   datai <- subset(data.long.TINDEP.1, ID == i)

   # Fit CZ
   sfiCZ <- survfit(CZ, newdata = datai[1,])
   ssfiCZ <- summary(sfiCZ, times = datai$Tstart)
   KZti <- c(KZti, ssfiCZ$surv)

   save(KZti, i,

        file = "./DataLongTINDEP1Tussendoor.Rdata"
   )
 }
> close(pb)
> data.long.TINDEP.1$K1ti <- KZti
> data.long.TINDEP.1$WUnStab <- 1/data.long.TINDEP.1$KZti
> data.long.TINDEP.1$WStab <- data.long.TINDEP.1$K0ti/
                              data.long.TINDEP.1$KZti
> save(data.long.TINDEP.1,

        file = "./DataLongTINDEP1.Rdata"
   )
```

Load result.

```
> load("./DataLongTINDEP1.Rdata")
```

Plot Product-Limit Curves.

```
> ##########################################################################
> # Now compare survival curves:
> survBaselineNoIPCW <- survfit(Surv(Tstart,CorrectedTimes,herstelkrap) ~ 1,
                                data=data.long.TINDEP.1)
> survIPCWTINDEP1Stab <- survfit(Surv(Tstart,CorrectedTimes,herstelkrap) ~ 1,
                            data=data.long.TINDEP.1, weights=WStab)
> survIPCWTINDEP1UnStab <- survfit(Surv(Tstart,CorrectedTimes,herstelkrap) ~ 1,
                                data=data.long.TINDEP.1, weights=WUnStab)
> png("KMCurvesMod1TimeIndepIPCW.png", width = 900, height = 600)
> plot(survBaselineNoIPCW$time, survBaselineNoIPCW$surv, col="black",
      lty=1, type = "l", ylim = c(0,1), xlab = "Time (days)",
      ylab = "Survival Probability")
```

```
> lines(survIPCWTINDEP1Stab$time, survIPCWTINDEP1Stab$surv, col="red",lty=1)
> legend(legend = c("No IPCW", "IPCW (Un)stabilized Weights"),
        col = c("black", "red"),
        lty = 1,
        x = "bottomleft",
        cex = 1.5,
        bty = "n"
 )
> dev.off()
```

Fit Cox models.

```
> ######################################################################
> # With Cox PH model for survival:
> # no weights:
> CoxNoIPCW <- coxph(Surv(Tstart, CorrectedTimes, event = herstelkrap) ~
 1 + ZLeeftijd + gendernum + agorafobie + paniekagfob + socfob +
 GAS + ZSFalgcorr + livingsituation3 + education + occupation +
 ethnicity + alcohol + ZBAS.1 + ZAFFLAB + ZPASSAG + ZGEDRAG +
 singleanx, data = data.long.TINDEP.1)
> survCoxNoIPCWSUM <- summary(CoxNoIPCW)
> CoxNoIPCWRES <- data.frame(
   HR = survCoxNoIPCWSUM$coefficients[,2],
   lower.95 = exp(survCoxNoIPCWSUM$coefficients[,1] -
                    qnorm(0.975, 0, 1, lower.tail=T) *
                     survCoxNoIPCWSUM$coefficients[,3]),
   upper.95 = exp(survCoxNoIPCWSUM$coefficients[,1] +
                    qnorm(0.975, 0, 1, lower.tail=T) *
                     survCoxNoIPCWSUM$coefficients[,3])
                          )

> # Stabilized weights - does not work since some weights are 0
> #CoxIPCWTINDEP1WStab <- coxph(Surv(Tstart, CorrectedTimes,
> # event = herstelkrap) ~
> #                         1 + ZLeeftijd + gendernum + agorafobie +
> #                         paniekagfob + socfob +
> #                         GAS + ZSFalgcorr + livingsituation3 + education +
> #                         occupation +
> #                         ethnicity + alcohol + ZBAS.1 + ZAFFLAB + ZPASSAG +
> #                         ZGEDRAG +
> #                         singleanx, data = data.long.TINDEP.1,
> #                       weights = WStab)
>
> # Stabilized weights truncated
```

```
> CoxIPCWTINDEP1WStabTrunc <- coxph(Surv(Tstart, CorrectedTimes,
                                         event = herstelkrap) ~
                        1 + ZLeeftijd + gendernum + agorafobie + paniekagfob +
                                  socfob +
                        GAS + ZSFalgcorr + livingsituation3 + education +
                                  occupation +
                        ethnicity + alcohol + ZBAS.1 + ZAFFLAB + ZPASSAG +
                                  ZGEDRAG +
                        singleanx, data = data.long.TINDEP.1,
                       weights = WStabTrunc)
> CoxIPCWTINDEP1WStabTruncSUM <- summary(CoxIPCWTINDEP1WStabTrunc)
> CoxIPCWTINDEP1WStabTruncRES <- data.frame(
   HR = CoxIPCWTINDEP1WStabTruncSUM$coefficients[,2],
   lower.95 = exp(CoxIPCWTINDEP1WStabTruncSUM$coefficients[,1] -
                    qnorm(0.975, 0, 1, lower.tail=T) *
                     CoxIPCWTINDEP1WStabTruncSUM$coefficients[,3]),
   upper.95 = exp(CoxIPCWTINDEP1WStabTruncSUM$coefficients[,1] +
                    qnorm(0.975, 0, 1, lower.tail=T) *
                     CoxIPCWTINDEP1WStabTruncSUM$coefficients[,3])
                         )
> # Unstabilized weights
> CoxIPCWTINDEP1WUnStab <- coxph(Surv(Tstart, CorrectedTimes,
                                         event = herstelkrap) ~
                        1 + ZLeeftijd + gendernum + agorafobie + paniekagfob +
                                 socfob +
                        GAS + ZSFalgcorr + livingsituation3 + education +
                                 occupation +
                        ethnicity + alcohol + ZBAS.1 + ZAFFLAB + ZPASSAG +
                                 ZGEDRAG +
                        singleanx, data = data.long.TINDEP.1,
                       weights = WUnStab)
> CoxIPCWTINDEP1WUnStabSUM <- summary(CoxIPCWTINDEP1WUnStab)
> CoxIPCWTINDEP1WUnStabRES <- data.frame(
   HR = CoxIPCWTINDEP1WUnStabSUM$coefficients[,2],
   lower.95 = exp(CoxIPCWTINDEP1WUnStabSUM$coefficients[,1] -
                    qnorm(0.975, 0, 1, lower.tail=T) *
                    CoxIPCWTINDEP1WUnStabSUM$coefficients[,3]),
   upper.95 = exp(CoxIPCWTINDEP1WUnStabSUM$coefficients[,1] +
                    qnorm(0.975, 0, 1, lower.tail=T) *
                    CoxIPCWTINDEP1WUnStabSUM$coefficients[,3])
     )
```

Model 3: Baseline Covariates with a Significant Influence on Censoring Time

Find covariates that are significant for time to censoring.

```
> ##########################################################
> ## Fit model type 3 for time to censoring with Cox PH ##
> ##########################################################
> library(survival)
> library(tcltk) # for the progress bar
> ############################
> ## Loading and the data ##
> ############################
>
> # The data in long format, together with the Product-Limit estimators for
> # the time to censoring model without covariates are available in the
> # following file:
> load("C:\\Users\\Sanne\\Documents\\School\\LU - MA2\\Scriptie\\Data\\DataLongIPCW.Rd
> data.long.TINDEP.2 <- data.long
> ########################################################################
> ## Fit the censoring model for Model 1 with only time independent ##
> ## covariates significant for time to censoring.                    ##
> ########################################################################
>
> ###############################
> ## Univariate Hazard Ratios ##
> ###############################
>
> # The first step in the analysis in the article is to calculate the
> # univariate Hazard Ratios of censoring, to see which variables are
> # significant by themselves ( alpha = 0.10).
>
> # The names of all the variables are:
>
> univars <- names(data.long.TINDEP.2)[1:48][-c(which(
                        names(data.long.TINDEP.2) == "ID"),
                           which(names(data.long.TINDEP.2) == "herstelkrap")
                       )]
> univars <- c(univars, "ZBSI_12.1")
> # The results of the univariate Hazard Ratio's will be stored in the
> # following two dataframes. Their rows are given the name of the
> # corresponding variable.
> univarResultsCens <- data.frame(
                   coef = rep(NA, times = length(univars)),
                   expCoef = rep(NA, times = length(univars)),
```

```
                    pValue = rep(NA, times = length(univars))
                )
> row.names(univarResultsCens) <- univars
> # Now, for each of these variables, the univariate Proportional Hazard
> # will be fitted and the results will be stored in the dataframe defined
> # above, the package survival is needed for that.
> j <- 1
> pb <- tkProgressBar(title = "Working hard:", min = 0, max = length(univars),
                      width = 300)
> for (i in univars)
 {
   setTkProgressBar(pb, j, title = "% ready!")

   command1 <- paste( "univarPH <- coxph(Surv(time = Tstart, time2 = CorrectedTimes, event =
                       censored) ~ 1 +", i, " , data = data.long.TINDEP.2)")

   eval(parse(text=command1))

   univarResultsCens[j,] = c(univarPH$coefficients[1],
                          exp(univarPH$coefficients[1]),
                          anova(univarPH)[2,4]
                        )

   # update j:
   j <- j + 1
 }
> close(pb)
> # The variables with a p-value <= 0.10 are:
> row.names(univarResultsCens[which(univarResultsCens$pValue < 0.10),])
> # Results:
> # [1] "D_DEPDYS"    "ZSFggzcorr"  "ZMADRS.1"    "agorafobie"  "paniekagfob"
> # [6] "ZLeeftijd"
>
> ################################
> ## Multivariate Hazard Ratios ##
> ################################
>
> # All variables which had a p-value <= 0.10 are tested in the mulitvariate
> # model. Furthermore, age, gender and the four dichotomized main
> # diagnostic categories in this study, are forced into the model.
>
> # So the indices of the variables that should be tested in the
> # multivariate model are:
```

```
> multivarsAllCensCoxM3 <- unique(c(which(row.names(univarResultsCens) ==
                                          "ZLeeftijd"),
                            which(row.names(univarResultsCens) == "gendernum"),
                            which(row.names(univarResultsCens) == "agorafobie"),
                            which(row.names(univarResultsCens) == "paniekagfob"),
                            which(row.names(univarResultsCens) == "socfob"),
                            which(row.names(univarResultsCens) == "GAS"),
                            which(univarResultsCens$pValue < 0.10)))
> # The names of these variables have to be pasted in one line so that they
> # can be included in the model:
>
> multivarsCensCoxM3 <- ""
> for(k in multivarsAllCensCoxM3)
 {
   multivarsCensCoxM3 <- paste(multivarsCensCoxM3,
                               row.names(univarResultsCens)[k],
                               sep= " + ")
 }
> multivarsCensCoxM3
> # + ZLeeftijd + gendernum + agorafobie + paniekagfob + socfob + GAS +
> # + D_DEPDYS + ZSFggzcorr + ZMADRS.1
>
> # Step 1:
> # Baseline multivariate model
> set.seed(123)
> multiCensCoxMod3.1 <- coxph(Surv(time = Tstart, time2 = CorrectedTimes,
                                   event = censored) ~
                              ZLeeftijd + gendernum + agorafobie +
                              paniekagfob + socfob + GAS + D_DEPDYS +
                              ZSFggzcorr + ZMADRS.1,
                              data = data.long.TINDEP.2, model = T)
> multiCensCoxMod3.1
> # Substep 1: highest p-value for ZMADRS.1, candidate for removal
> multiCensCoxMod3.1verg <- drop1(multiCensCoxMod3.1, ~ ZMADRS.1 ,
                                  test =  "none")
> multiCensCoxMod3.1verg
> # AIC decreases, so removal is justified
>
> # Second multivariate model
> set.seed(123)
> multiCensCoxMod3.2 <- coxph(Surv(time = Tstart, time2 = CorrectedTimes,
                                   event = censored) ~
                              ZLeeftijd + gendernum + agorafobie +
```

```
                          paniekagfob + socfob + GAS + D_DEPDYS +
                          ZSFggzcorr,
                        data = data.long.TINDEP.2, model = T)
> multiCensCoxMod3.2
> # biggest p-value for D_DEPDYS
> multiCensCoxMod3.2verg <- drop1(multiCensCoxMod3.2, ~ D_DEPDYS,
                                  test =  "none")
> multiCensCoxMod3.2verg
> # AIC decreases, justified
>
> # Third multivariate model
> set.seed(123)
> multiCensCoxMod3.3 <- coxph(Surv(time = Tstart, time2 = CorrectedTimes,
                                   event = censored) ~
                          ZLeeftijd + gendernum + agorafobie +
                          paniekagfob + socfob + GAS +
                          ZSFggzcorr,
                        data = data.long.TINDEP.2, model = T)
> multiCensCoxMod3.3
> # No more unsignificant variables (except those who are forced in the
> # model.
>
> ##################
> ## Final model ##
> ##################
>
> # So multivarPH3Cens is the final model
> # coxph(formula = Surv(time = CorrectedTimes, event = herstelkrap) ~
> #  + ZLeeftijd + gendernum + agorafobie + paniekagfob + socfob + GAS +
> # + ZSFggzcorr,
> # data = data)
> multiCensCoxMod3.3sum <- summary(multiCensCoxMod3.3)
> censModCoxMod3res <- data.frame(
   Coef = multiCensCoxMod3.3sum$coefficients[,1],
   expCoef = multiCensCoxMod3.3sum$coefficients[,2],
   lower.95 = exp(multiCensCoxMod3.3sum$coefficients[,1] -
                    qnorm(0.975, 0, 1, lower.tail=T) *
                    multiCensCoxMod3.3sum$coefficients[,3]),
   upper.95 = exp(multiCensCoxMod3.3sum$coefficients[,1] +
                    qnorm(0.975, 0, 1, lower.tail=T) *
                    multiCensCoxMod3.3sum$coefficients[,3]),
   pValue = multiCensCoxMod3.3sum$coefficients[,5]
 )
```

```
> library(xtable)
> xtable(censModCoxMod3res, digits = 3)
```

Calculate weights.

```
> ###############################################
> ## Application of IPCW method to ROM dataset ##
> ###############################################
>
> # Calculate the survival estimates for time to censoring for both the
> # C0 model and the C1 model:
> # models:
> CZ <- multiCensCoxMod3.3
> # Calculate the Product-Limit estimators for model CZ, the KZti
> # calculate the estimates for each of the individuals. Since the model
> # only includes time independent variables, the survival curve does not
> # have to be fit for every line, but can be fit for each subject.
> IDs <- unique(data.long.TINDEP.2$ID)
> pb <- tkProgressBar(title = "Working hard:", min = 0, max = max(IDs),
                      width = 300)
> KZti <- NULL
> for(i in IDs)
 {
   setTkProgressBar(pb, i, title = "% ready!")

   datai <- subset(data.long.TINDEP.2, ID == i)

   # Fit CZ
   sfiCZ <- survfit(CZ, newdata = datai[1,])
   ssfiCZ <- summary(sfiCZ, times = datai$Tstart)
   KZti <- c(KZti, ssfiCZ$surv)

   save(KZti, i,

        file = "C:\\Users\\Sanne\\Documents\\School\\LU - MA2\\Scriptie\\Data\\DataLo
   )
 }
> close(pb)
> data.long.TINDEP.2$KZti <- KZti
> data.long.TINDEP.2$WUnStab <- 1/data.long.TINDEP.2$KZti
> data.long.TINDEP.2$WStab <- data.long.TINDEP.2$K0ti/
                                    data.long.TINDEP.2$KZti
> save(data.long.TINDEP.2,
```

```
        file = "./DataLongTINDEP2.Rdata"
   )
```

Load result.

```
> load("./DataLongTINDEP2.Rdata")
```

Plot Product-Limit curves.

```
> # Now compare survival curves, not including any covariates:
> survBaselineNoIPCW <- survfit(Surv(Tstart,CorrectedTimes,herstelkrap) ~ 1,
                                data=data.long.TINDEP.2)
> survIPCWTINDEP2Stab <- survfit(Surv(Tstart,CorrectedTimes,herstelkrap) ~ 1,
                                 data=data.long.TINDEP.2, weights=WStab)
> survIPCWTINDEP2UnStab <- survfit(Surv(Tstart,CorrectedTimes,herstelkrap) ~ 1,
                                   data=data.long.TINDEP.2, weights=WUnStab)
> png("KMCurvesMod2TimeIndepIPCW.png", width = 900, height = 600)
> plot(survBaselineNoIPCW$time, survBaselineNoIPCW$surv, col="black",
      lty=1, type = "l", ylim = c(0,1), xlab = "Time (days)",
      ylab = "Survival Probability")
> lines(survIPCWTINDEP2Stab$time, survIPCWTINDEP2Stab$surv, col="red",
       lty=1)
> legend(legend = c("No IPCW", "IPCW (Un)stabilized Weights"),
        col = c("black", "red"),
        lty = 1,
        x = "bottomleft",
        cex = 1.5,
        bty = "n"
 )
> dev.off()
```

Fit Cox models.

```
> # Stabilized weights truncated
> CoxIPCWTINDEP2WStabTrunc <- coxph(Surv(Tstart, CorrectedTimes,
                                         event = herstelkrap) ~
                          1 + ZLeeftijd + gendernum + agorafobie + paniekagfob +
                                     socfob +
                          GAS + ZSFalgcorr + livingsituation3 + education +
                                     occupation +
                          ethnicity + alcohol + ZBAS.1 + ZAFFLAB + ZPASSAG +
                                     ZGEDRAG +
                          singleanx, data = data.long.TINDEP.2,
                        weights = WStabTrunc)
> CoxIPCWTINDEP2WStabTruncSUM <- summary(CoxIPCWTINDEP2WStabTrunc)
> CoxIPCWTINDEP2WStabTruncRES <- data.frame(
```

```
   HR = CoxIPCWTINDEP2WStabTruncSUM$coefficients[,2],
   lower.95 = exp(CoxIPCWTINDEP2WStabTruncSUM$coefficients[,1] -
                qnorm(0.975, 0, 1, lower.tail=T) *
                 CoxIPCWTINDEP2WStabTruncSUM$coefficients[,3]),
   upper.95 = exp(CoxIPCWTINDEP2WStabTruncSUM$coefficients[,1] +
                qnorm(0.975, 0, 1, lower.tail=T) *
                 CoxIPCWTINDEP2WStabTruncSUM$coefficients[,3])
                             )
> # Unstabilized weights
> CoxIPCWTINDEP2WUnStab <- coxph(Surv(Tstart, CorrectedTimes,
                                    event = herstelkrap) ~
                          1 + ZLeeftijd + gendernum + agorafobie +
                                   paniekagfob +
                                   socfob +
                          GAS + ZSFalgcorr + livingsituation3 + education +
                                   occupation +
                          ethnicity + alcohol + ZBAS.1 + ZAFFLAB + ZPASSAG +
                                   ZGEDRAG +
                          singleanx, data = data.long.TINDEP.2,
                         weights = WUnStab)
> CoxIPCWTINDEP2WUnStabSUM <- summary(CoxIPCWTINDEP2WUnStab)
> CoxIPCWTINDEP2WUnStabRES <- data.frame(
   HR = CoxIPCWTINDEP2WUnStabSUM$coefficients[,2],
   lower.95 = exp(CoxIPCWTINDEP2WUnStabSUM$coefficients[,1] -
                   qnorm(0.975, 0, 1, lower.tail=T) *
                   CoxIPCWTINDEP2WUnStabSUM$coefficients[,3]),
   upper.95 = exp(CoxIPCWTINDEP2WUnStabSUM$coefficients[,1] +
                   qnorm(0.975, 0, 1, lower.tail=T) *
                   CoxIPCWTINDEP2WUnStabSUM$coefficients[,3])
     )
```

Compare Regression Coefficients

Plot confidence intervals of significant covariates.

```
> ########################################################################
> # Plot coefficient estimates with CI
> library(gplots)
> xnames <- c("Age",
               "Gender: male",
               "Agoraphobia",
               "Panic",
               "Social",
               "GAD",
```

```
                "General Health",
                "Livingsituation: alone",
                "Livingsituation: with family",
                "Education: medium",
                "Education: low",
                "Occupation: no",
                "Ethnicity: non-Dutch",
                "Alcohol abuse",
                "BAS",
                "Affective Liability",
                "Oppositionality",
                "Conduct",
                "Multiple Anxiety Dis.")
> ## Unstabilized weights
> png("CompareRegrCoefIPCWUnStab.png", width = 900, height = 600)
> par(mar = c(12,5,4,2) + 0.1)
> plotCI(x = 1:nrow(CoxNoIPCWRES)-0.20,
        y = CoxNoIPCWRES$HR,
        uiw = NULL,
        liw = NULL,
        ui = CoxNoIPCWRES$upper.95,
        li = CoxNoIPCWRES$lower.95,
        err = "y",
        type = "p",
        pch=16,
        cex = 1,
        gap =0,
        lty = "dashed",
        sfrac = 0.0025,
        ylab = "HR and 95% CI",
        xlab = "",
        xaxt = "n",
        ylim = c(0.25,2.25),
        lwd = 2,
        cex.lab = 1.5
        )
> axis(1, at=1:nrow(CoxNoIPCWRES),
      labels=xnames,
      padj = "0",
      las=2,
      cex.axis = 1.5)
> plotCI(x = 1:nrow(CoxIPCWTDEP1WUnStabRES)-0.08,
        y = CoxIPCWTDEP1WUnStabRES$HR,
```

```
        uiw = NULL,
        liw = NULL,
        ui = CoxIPCWTDEP1WUnStabRES$upper.95,
        li = CoxIPCWTDEP1WUnStabRES$lower.95,
        err = "y",
        type = "p",
        pch=16,
        cex = 1,
        gap =0,
        lty = "dashed",
        sfrac = 0.0025,
        xaxt = "n",
        add = T,
        col = "forestgreen",
        lwd = 2)
> plotCI(x = 1:nrow(CoxIPCWTINDEP1WUnStabRES)+0.04,
        y = CoxIPCWTINDEP1WUnStabRES$HR,
        uiw = NULL,
        liw = NULL,
        ui = CoxIPCWTINDEP1WUnStabRES$upper.95,
        li = CoxIPCWTINDEP1WUnStabRES$lower.95,
        err = "y",
        type = "p",
        pch=16,
        cex = 1,
        gap =0,
        lty = "dashed",
        sfrac = 0.0025,
        xaxt = "n",
        add = T,
        col = "red",
        lwd = 2)
> plotCI(x = 1:nrow(CoxIPCWTINDEP2WUnStabRES)+0.16,
        y = CoxIPCWTINDEP2WUnStabRES$HR,
        uiw = NULL,
        liw = NULL,
        ui = CoxIPCWTINDEP2WUnStabRES$upper.95,
        li = CoxIPCWTINDEP2WUnStabRES$lower.95,
        err = "y",
        type = "p",
        pch=16,
        cex = 1,
        gap =0,
```

```
        lty = "dashed",
        sfrac = 0.0025,
        xaxt = "n",
        add = T,
        col = "blue",
        lwd = 2)
> abline(h=1,  col = "grey")
> dev.off()
> ## Stabilized truncated weights
> png("CompareRegrCoefIPCWStabTrunc.png", width = 900, height = 600)
> par(mar = c(12,5,4,2) + 0.1)
> plotCI(x = 1:nrow(CoxNoIPCWRES)-0.20,
        y = CoxNoIPCWRES$HR,
        uiw = NULL,
        liw = NULL,
        ui = CoxNoIPCWRES$upper.95,
        li = CoxNoIPCWRES$lower.95,
        err = "y",
        type = "p",
        pch=16,
        cex = 0.5,
        gap =0,
        lty = "dashed",
        sfrac = 0.0025,
        ylab = "HR and 95% CI",
        xlab = "",
        xaxt = "n",
        ylim = c(0.25,2.25),
        lwd = 2,
        cex.lab = 1.5
        )
> axis(1, at=1:nrow(CoxNoIPCWRES),
      labels=xnames,
      padj = "0",
      las=2,
      cex.axis = 1.5)
> plotCI(x = 1:nrow(CoxIPCWTDEP1WStabTruncRES)-0.08,
        y = CoxIPCWTDEP1WStabTruncRES$HR,
        uiw = NULL,
        liw = NULL,
        ui = CoxIPCWTDEP1WStabTruncRES$upper.95,
        li = CoxIPCWTDEP1WStabTruncRES$lower.95,
        err = "y",
```

```
        type = "p",
        pch=16,
        cex = 0.5,
        gap =0,
        lty = "dashed",
        sfrac = 0.0025,
        xaxt = "n",
        add = T,
        col = "forestgreen",
        lwd = 2)
> plotCI(x = 1:nrow(CoxIPCWTINDEP1WStabTruncRES)+0.04,
        y = CoxIPCWTINDEP1WStabTruncRES$HR,
        uiw = NULL,
        liw = NULL,
        ui = CoxIPCWTINDEP1WStabTruncRES$upper.95,
        li = CoxIPCWTINDEP1WStabTruncRES$lower.95,
        err = "y",
        type = "p",
        pch=16,
        cex = 0.5,
        gap =0,
        lty = "dashed",
        sfrac = 0.0025,
        xaxt = "n",
        add = T,
        col = "red",
        lwd = 2)
> plotCI(x = 1:nrow(CoxIPCWTINDEP2WStabTruncRES)+0.16,
        y = CoxIPCWTINDEP2WStabTruncRES$HR,
        uiw = NULL,
        liw = NULL,
        ui = CoxIPCWTINDEP2WStabTruncRES$upper.95,
        li = CoxIPCWTINDEP2WStabTruncRES$lower.95,
        err = "y",
        type = "p",
        pch=16,
        cex = 0.5,
        gap =0,
        lty = "dashed",
        sfrac = 0.0025,
        xaxt = "n",
        add = T,
        col = "blue",
```

```
       lwd = 2)
> abline(h=1,  col = "grey")
> dev.off()
```

Load results.

```
> # Fit survival curves for one individual whose difference between
> # B_IPCW * X - B_0 * X is biggest
>
> # required libraries:
> library(survival)
> # Load original data and with weights for IPCW models:
> load("C:\\Users\\Sanne\\Documents\\School\\LU - MA2\\Scriptie\\Data\\DataCorrected.Rdata"
 )
> load("C:\\Users\\Sanne\\Documents\\School\\LU - MA2\\Scriptie\\Data\\DataLongTINDEP1.Rdata
 )
> load("C:\\Users\\Sanne\\Documents\\School\\LU - MA2\\Scriptie\\Data\\DataLongTINDEP2.Rdata
 )
```

Biggest Difference, person 1.

```
> ######################################################################
> # Estimate Cox-model without using weights:
> CoxNoIPCW <- coxph(Surv(Tstart, CorrectedTimes, event = herstelkrap) ~

                      1 + ZLeeftijd + gendernum + agorafobie + paniekagfob +
                      socfob +
                      GAS + ZSFalgcorr + livingsituation3 + education +
                      occupation +
                      ethnicity + alcohol + ZBAS.1 + ZAFFLAB + ZPASSAG +
                      ZGEDRAG +
                      singleanx, data = data.long.TINDEP.1)
> # Estimate Cox model (time independent covariates 1) with unstabilized
> # weights:
> CoxIPCWTINDEP1WUnStab <- coxph(Surv(Tstart, CorrectedTimes,
                                      event = herstelkrap) ~
                                  1 + ZLeeftijd + gendernum + agorafobie +
                                  paniekagfob + socfob +
                                  GAS + ZSFalgcorr + livingsituation3 +
                                  education + occupation +
                                  ethnicity + alcohol + ZBAS.1 + ZAFFLAB +
                                  ZPASSAG + ZGEDRAG +
                                  singleanx, data = data.long.TINDEP.1,
                                weights = WUnStab)
> # Now find the individual with the biggest difference in
```

```
> # beta_IPCW * x and beta_noIPCW *x
> # use the original data, where all individuals are included just once:
> individual.max.diff.TINDEP.1 <- which.max(abs(
   predict(CoxIPCWTINDEP1WUnStab, newdata = data, type = "lp") -
     predict(CoxNoIPCW, newdata = data, type = "lp")))
> IndivSurvNoIPCW.TINDEP.1 <- survfit(CoxNoIPCW,
                            newdata = data[individual.max.diff.TINDEP.1,])
> IndivSurvWithIPCW.TINDEP.1 <- survfit(CoxIPCWTINDEP1WUnStab,
                              newdata = data[individual.max.diff.TINDEP.1,])
> ############################################################
> # Now do the same for the third Cox model
> ############################################################################
> # Estimate Cox-model without using weights:
> CoxNoIPCW <- coxph(Surv(Tstart, CorrectedTimes, event = herstelkrap) ~
                      1 + ZLeeftijd + gendernum + agorafobie + paniekagfob +
                      socfob +
                      GAS + ZSFalgcorr + livingsituation3 + education +
                      occupation +
                      ethnicity + alcohol + ZBAS.1 + ZAFFLAB + ZPASSAG +
                      ZGEDRAG +
                      singleanx, data = data.long.TINDEP.2)
> # Estimate Cox model (time independent covariates 1) with unstabilized
> # weights:
> CoxIPCWTINDEP2WUnStab <- coxph(Surv(Tstart, CorrectedTimes,
                                      event = herstelkrap) ~
                                   1 + ZLeeftijd + gendernum + agorafobie +
                                   paniekagfob + socfob +
                                   GAS + ZSFalgcorr + livingsituation3 +
                                   education + occupation +
                                   ethnicity + alcohol + ZBAS.1 + ZAFFLAB +
                                   ZPASSAG + ZGEDRAG +
                                   singleanx, data = data.long.TINDEP.2,
                                 weights = WUnStab)
> # Now find the individual with the biggest difference in
> # beta_IPCW * x and beta_noIPCW *x
> # use the original data, where all individuals are included just once:
> individual.max.diff.TINDEP.2 <- which.max(abs(
   predict(CoxIPCWTINDEP2WUnStab, newdata = data, type = "lp") -
     predict(CoxNoIPCW, newdata = data, type = "lp")))
> IndivSurvNoIPCW.TINDEP.2 <- survfit(CoxNoIPCW,
           newdata = data[individual.max.diff.TINDEP.2,])
> IndivSurvWithIPCW.TINDEP.2 <- survfit(CoxIPCWTINDEP1WUnStab,
           newdata = data[individual.max.diff.TINDEP.2,])
```

Biggest Difference, person 2.

```
> ######################################################################
> # Estimate Cox-model without using weights:
> CoxNoIPCW <- coxph(Surv(Tstart, CorrectedTimes, event = herstelkrap) ~
                     1 + ZLeeftijd + gendernum + agorafobie + paniekagfob +
                     socfob +
                     GAS + ZSFalgcorr + livingsituation3 + education +
                     occupation +
                     ethnicity + alcohol + ZBAS.1 + ZAFFLAB + ZPASSAG +
                     ZGEDRAG +
                     singleanx, data = data.long.TINDEP.1)
> # Estimate Cox model (time independent covariates 1) with unstabilized
> # weights:
> CoxIPCWTINDEP1WUnStab <- coxph(Surv(Tstart, CorrectedTimes,
                                      event = herstelkrap) ~
                                  1 + ZLeeftijd + gendernum + agorafobie +
                                  paniekagfob + socfob +
                                  GAS + ZSFalgcorr + livingsituation3 +
                                  education + occupation +
                                  ethnicity + alcohol + ZBAS.1 + ZAFFLAB +
                                  ZPASSAG + ZGEDRAG +
                                  singleanx, data = data.long.TINDEP.1,
                                weights = WUnStab)
> # Now find the individual with the biggest difference in
> # beta_IPCW * x and beta_noIPCW *x
> # use the original data, where all individuals are included just once:
> indivicual.max.diff.TINDEP.1 <- which.max(abs(
   predict(CoxIPCWTINDEP1WUnStab, newdata = data, type = "lp") -
     predict(CoxNoIPCW, newdata = data, type = "lp")))
> IndivSurvNoIPCW.TINDEP.1 <- survfit(CoxNoIPCW,
                             newdata = data[indivicual.max.diff.TINDEP.1,])
> IndivSurvWithIPCW.TINDEP.1 <- survfit(CoxIPCWTINDEP1WUnStab,
                               newdata = data[indivicual.max.diff.TINDEP.1,])
> ##############################################################
> # Now do the same for the second Cox model
> ######################################################################
> # Estimate Cox-model without using weights:
> CoxNoIPCW <- coxph(Surv(Tstart, CorrectedTimes, event = herstelkrap) ~
                     1 + ZLeeftijd + gendernum + agorafobie +
                     paniekagfob +
                     socfob +
                     GAS + ZSFalgcorr + livingsituation3 + education +
                     occupation +
```

```
                    ethnicity + alcohol + ZBAS.1 + ZAFFLAB + ZPASSAG +
                    ZGEDRAG +
                    singleanx, data = data.long.TINDEP.2)
> # Estimate Cox model (time independent covariates 1) with unstabilized
> # weights:
> CoxIPCWTINDEP2WUnStab <- coxph(Surv(Tstart, CorrectedTimes,
                                    event = herstelkrap) ~
                                  1 + ZLeeftijd + gendernum + agorafobie +
                                  paniekagfob + socfob +
                                  GAS + ZSFalgcorr + livingsituation3 +
                                  education + occupation +
                                  ethnicity + alcohol + ZBAS.1 + ZAFFLAB +
                                  ZPASSAG + ZGEDRAG +
                                  singleanx, data = data.long.TINDEP.2,
                                weights = WUnStab)
> # Now find the individual with the biggest difference in
> # beta_IPCW * x and beta_noIPCW *x
> # use the original data, where all individuals are included just once:
> indivicual.max.diff.TINDEP.2 <- which.max(abs(
   predict(CoxIPCWTINDEP2WUnStab, newdata = data, type = "lp") -
     predict(CoxNoIPCW, newdata = data, type = "lp")))
> IndivSurvNoIPCW.TINDEP.2 <- survfit(CoxNoIPCW,
                     newdata = data[indivicual.max.diff.TINDEP.2,])
> IndivSurvWithIPCW.TINDEP.2 <- survfit(CoxIPCWTINDEP1WUnStab,
                     newdata = data[indivicual.max.diff.TINDEP.2,])
```

Plot results.

```
> png("BiggestDifferenceIndividualTINDEP1.png", width = 600, height = 400)
> plot(IndivSurvNoIPCW.TINDEP.1$time, IndivSurvNoIPCW.TINDEP.1$surv,
      col="black",
      lty=1, type = "s", ylim = c(0,1), xlab = "Time (days)",
      ylab = "Survival Probability",
      cex.lab= 1.5,
      lwd = 2)
> lines(IndivSurvWithIPCW.TINDEP.1$time, IndivSurvWithIPCW.TINDEP.1$surv,
       col="red",lty=1,
       type = "s",
       lwd = 2)
> legend(legend = c("Estimated survival curve for standard Cox model",
                    "Estimated survival curve for IPCW Cox model 2"),
        col = c("black", "red"),
        lty = 1,
        x = "bottomleft",
```

```
        cex = 1.5,
        bty = "n"
 )
> dev.off()
> png("BiggestDifferenceIndividualTINDEP2.png", width = 600, height = 400)
> plot(IndivSurvNoIPCW.TINDEP.2$time, IndivSurvNoIPCW.TINDEP.2$surv,
      col="black",
      lty=1, type = "s", ylim = c(0,1), xlab = "Time (days)",
      ylab = "Survival Probability",
      lwd = 2,
      cex.lab = 1.5)
> lines(IndivSurvWithIPCW.TINDEP.2$time, IndivSurvWithIPCW.TINDEP.2$surv,
       col="red",lty=1,
       type = "s",
       lwd = 2)
> legend(legend = c("Estimated survival curve for standard Cox model",
                    "Estimated survival curve for IPCW Cox model 3"),
        col = c("black", "red"),
        lty = 1,
        x = "bottomleft",
        cex = 1.5,
        bty = "n"
 )
> dev.off()
```

C.5 R Code for Chapter 7

C.5.1 Functions Defined for Simulations

Many functions were defined to for the simulation study. The R code is given below. Function to estimate the hazard rates for time to event and time to censoring, which depent on the covariates and follow an exponential distribution.

```
> get.hazards <- function(data, beta, lambda0X, phi, lambda0C)
 {
   data$lambdaX <- lambda0X * exp(cbind(data$ZAge, data$Gender) %*%
                                  beta)
   data$lambdaC <- lambda0C * exp(cbind(data$ZAge, data$Gender) %*%
                                  phi)
   return(data)
 }
```

Function to generate observed data (t_i, d_i) for each patient in the dataset.

```
> generateXC <- function(data)
 {
   # Generate u1 and u2 from uniform distribution for each of the n patients:
   data$u1 <- runif(n)
   data$u2 <- runif(n)

   # For each patient, generate ti and ci with the lambda's and u's:
   # (use theorem of probability integral transform)
   data$xi <- qexp(p = data$u1, rate = data$lambdaX)
   data$ci <- qexp(p = data$u2, rate = data$lambdaC)

   # Calculate ti and di:
   data$ti <- pmin(data$xi, data$ci)
   data$di <- as.numeric(data$xi <= data$ci)

   return(data)
 }
```

Function to estimate the survival probabilities at time points tt. This function can be used to estimate the real survival curve and to apply the standard method.

```
> calc.surv <- function(times, status, tt, data)
 {
   # Fit model
   #cox <- coxpSurv(times, status) ~ ZAge + Gender, data = data)
   surv <- survfit(Surv(times, status) ~ 1, data = data)

   # Estimate survival probabilities at time points tt
   ssf <- summary(surv, times = tt, extend = TRUE)

   # Return results:
   return(ssf$surv)

 }
```

Function to fit a Cox model.

```
> calc.beta <- function(times, status, data)
 {
   # Fit model:
   Cox <- coxph(Surv(times, status) ~ ZAge + Gender, data = data)

   # Return result:
   return(summary(Cox)$coef[,1])
```

```
 }
>
```

Function to transform data into long format.

```
> transform.data <- function(data)
 {
   # Define Tstart and the indicator "censored":
   data$Tstart <- 0
   data$censored <- 1 - data$di

   # Times at which to split the intervals:
   cut.times <- data$ti

   # Split data with event = di (event):
   data.long <- survSplit(data = data,
                          cut = cut.times,
                          end = "ti",
                          start = "Tstart",
                          event = "di")
   data.long <- data.long[order(data.long$ID,
                                data.long$ti),]

   # Split data with event = censored (censoring):
   data.long.cens <- survSplit(data,
                               cut=cut.times,
                               end="ti",
                               start="Tstart",
                               event="censored")
   data.long.cens <- data.long.cens[order(data.long.cens$ID,
                                          data.long.cens$ti),]

   # Add "censored" indicator to long data format:
   data.long$censored <- data.long.cens$censored
   data.long$ID <- as.numeric(data.long$ID)

   # Return long data format:
   return(data.long)
 }
```

Function to calculate the IPCW stabilized and unstabilized weights.

```
> calc.IPCW <- function(C0, CZ, data.long)
 {
   # Calculate survival estimates for model C0, K0ti:
```

```
  C0fit <- survfit(C0)
  C0sum <- summary(C0fit, times = data.long$Tstart)
  data.long$K0ti <- C0sum$surv

  # Calculate the survival estimates for model CZ, KZti:
  # Since the model only includes time independent variables, the survival
  # curve does not have to be fit for every line, but can be fit once for
  # each subject.
  IDs <- unique(data.long$ID)

  pbsub <- tkProgressBar(title = "Estimator for time to censoring",
                         min = 0, max = max(IDs)+1, width = 300)
  KZti <- NULL

  for(i in IDs)
  {
    setTkProgressBar(pbsub, i, title = "IPCW Weights: % ready!")

    # Take the data for subject i:
    datai <- subset(data.long, ID == i)

    # Fit CZ on the baseline covariates of subject i:
    sfiCZ <- survfit(CZ, newdata = datai[1,])
    ssfiCZ <- summary(sfiCZ, times = datai$Tstart)
    KZti <- c(KZti, ssfiCZ$surv)

  }
  close(pbsub)


  data.long$KZti <- KZti

  # Calculate Unstabilized and Stabilized weights:
  data.long$WUnStab <- 1/data.long$KZti
  data.long$WStab <- data.long$K0ti/data.long$KZti

  # Return result:
  return(data.long)
}
```

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

```
> calc.surv.IPCW <- function(Tstart, Tstop, status, tt, IPCW.weights,
```

```
                          data.long)
{
  # Fit the Product-Limit estimator with weighted subjects
  surv.IPCW <- survfit(Surv(Tstart, Tstop, status) ~ 1, data = data.long,
                       weights = IPCW.weights)

  # Estimate survival probabilities at time points tt
  ssurv.IPCW <- summary(surv.IPCW, times=tt, extend = TRUE)

  # Return resulting survival probabilities:
  return(ssurv.IPCW$surv)
}
```

Function to fit the Cox model for time to event with weighted subjects.

```
> calc.beta.IPCW <- function(Tstart, Tstop, status, IPCW.weights, data.long)
{
  # Fit model:
  Cox <- coxph(Surv(Tstart, Tstop, status,
                    type = "counting") ~ ZAge + Gender,
               data = data.long,
               weights = IPCW.weights)

  # Return results:
  return(summary(Cox)$coef[,1])
}
```

C.5.2 Code to Repeat Simulations

Load libraries and required functions.

```
> # Load libraries
> library(survival)
> library(tcltk)
> # Load functions used for the Monte Carlo Simulations from this file:
> source("C:\\Users\\Sanne\\Documents\\School\\LU - MA2\\Scriptie\\SimulatiesDef\\KMsimulati
```

Choose parameters.

```
> n <- 100 # number of subjects, varied to create different situations
> M <- 25 # number of Monte Carlo repetitions
> # Population parameters:
> meanAge <- 50
> sdAge <- 10
> # Parameters for hazard functions time to event and time to censoring:
```

```
> beta <- c(0.5, 1.5)
> lambda0X <- 0.1
> phi <- c(1.5, 5) # Varied to create different situations
> lambda0C <- 0.00557 # Varied to create different situations
> # Times at which survival probabilities should be calculated:
> tt <- c(seq(0,5,by=0.1),
          seq(5.25,10,by=0.25),
          seq(10.5,20,by=0.5),
          seq(21,50,by=1))
```

Define vectors to store results.

```
> # Save survival curve estimates in:
> surv.real <-  matrix(nrow = M, ncol = length(tt))
> surv.cens <-  matrix(nrow = M, ncol = length(tt))
> surv.IPCW.UnStab <- matrix(nrow = M, ncol = length(tt))
> surv.IPCW.Stab <- matrix(nrow = M, ncol = length(tt))
> # Save parameter estimates for time to event model in:
> beta.real <- beta.cens <- beta.IPCW.StabTrunc <- beta.IPCW.UnStab <-
    beta.IPCW.Stab <-
    matrix(nrow = M, ncol = 2)
> # Save parameter estimates for time to censoring model in:
> phi.IPCW <- matrix(nrow = M, ncol = 2)
> # Save censoring percentage in:
> cens.perc <- vector(mode = "numeric", length = M)
> # Save correlation between X and C in:
> correlationXC <- vector(mode = "numeric", length = M)
```

For-loop to repeat estimations M times.

```
> pb <- tkProgressBar(title = "Working hard:", min = 0,
                      max = M, width = 300)
> #set.seed(123)
> for(j in 1:M)
 {

    ####################
    ## Simulate data ##
    ####################
    data.sim <- data.frame(ID = 1:n,
                           Age = rnorm(n, mean = meanAge, sd = sdAge),
                           Gender = sample(0:1, size = n, replace = TRUE,
                                           prob = c(0.5,0.5))
                           )
```

```
data.sim$ZAge <- (data.sim$Age - meanAge)/sdAge

##########################################
## Estimate hazard rates for each subject ##
##########################################

data.sim <- get.hazards(data = data.sim, beta, lambda0X, phi, lambda0C)

################################
## Simulate ti, ci, xi and di ##
################################
data.sim <- generateXC(data.sim)

#############################################################################
## Calculate the censoring percentage and correlation between X and C ##
#############################################################################
cens.perc[j] <- 1 - sum(data.sim$di)/n
correlationXC[j] <- cor(data.sim$xi, data.sim$ci)

#####################################################################
## Calculate the survival curves and parameter estimates for ##
## the real model                                                ##
#####################################################################

# Real survival curves for test persons:
data.sim$one <- 1


surv.real[j,] <- calc.surv(times = data.sim$xi,  # event times
                           status = data.sim$one, # only events
                           tt = tt,
                           data = data.sim
                           )

# Real parameter estimates:
beta.real[j,] <- calc.beta(times = data.sim$xi, # event times
                           status = data.sim$one, # only events
                           data = data.sim)

#####################################################################
## Calculate the survival curves and parameter estimates with ##
## the standard method that assumes independent censoring     ##
#####################################################################
```

```
surv.cens[j,] <- calc.surv(times = data.sim$ti, # observed times
                           status = data.sim$di, # event indicator
                           tt = tt,
                           data = data.sim)

# Censoring parameter estimates:
beta.cens[j,] <- calc.beta(times = data.sim$ti, # observed times
                           status = data.sim$di, # event indicator
                           data = data.sim)



#####################################################################
## Calculate the survival curves and parameter estimates with ##
## the IPCW method that incorporates dependent censoring      ##
#####################################################################

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

# Fit time to censoring models, with and without covariates:
C0 <- coxph(Surv(time = Tstart,
                 time2 = ti,
                 event = censored,
                 type = "counting") ~ 1,
            data=data.sim.long)
CZ <- coxph(Surv(time = Tstart,
                 time2 = ti,
                 event = censored,
                 type = "counting") ~ ZAge + Gender,
            data=data.sim.long)

# Save parameter estimates for the time to censoring model
phi.IPCW[j,] <- summary(CZ)$coef[,1]

# Calculate the IPCW weights:
data.sim.long <- calc.IPCW(C0, CZ, data.sim.long)


########################
## Stabilized Weights ##
########################
```

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

   # Save the parameter estimates for the time to event model
   beta.IPCW.Stab[j,] <- calc.beta.IPCW(Tstart = data.sim.long$Tstart,
                                       Tstop = data.sim.long$ti,
                                       status = data.sim.long$di,
                                       IPCW.weights = data.sim.long$WStab,
                                       data.long = data.sim.long)

   ############################
   ## Unstabilized Weights ##
   ############################

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

    # Save the parameter estimates for the time to event model
   beta.IPCW.UnStab[j,] <- calc.beta.IPCW(Tstart = data.sim.long$Tstart,
                                Tstop = data.sim.long$ti,
                                status = data.sim.long$di,
                                IPCW.weights = data.sim.long$WUnStab,
                                data.long = data.sim.long)

   setTkProgressBar(pb, j, title = "M repeats: % ready!")

 }
> close(pb)
```

Save results.

```
> # Save results:
> save(surv.real, beta.real,
```

```
      surv.cens,  beta.cens,
      surv.IPCW.Stab, beta.IPCW.Stab,
      surv.IPCW.UnStab, beta.IPCW.UnStab,
      phi.IPCW,
      cens.perc, correlationXC,
      file = "C:\\Users\\Sanne\\Documents\\School\\LU - MA2\\Scriptie\\SimulatiesDef\\
 )
```

Summarize results.

```
> surv.real.MC <- colMeans(surv.real)
> surv.cens.MC <- colMeans(surv.cens)
> surv.IPCW.UnStab.MC <- colMeans(surv.IPCW.UnStab)
> surv.IPCW.Stab.MC <- colMeans(surv.IPCW.Stab)
```

Plot survival curves.

```
> # Code used to plot simulation results
> tt <- c(seq(0,5,by=0.1),seq(5.25,10,by=0.25),seq(10.5,20,by=0.5),seq(21,50,by=1))
> png("C:\\Users\\Sanne\\Documents\\School\\LU - MA2\\Scriptie\\SimulatiesDef\\KMsimul
     width = 400, height = 300)
> plot(tt, surv.real.MC, main = "",
      xlab = "time", ylab = "Survival Probability",
      type="l", xlim=c(0,50), ylim=c(0,1), col = "green",
      cex = 1.2,
      cex.main = 1.2,
      cex.axis = 1.2,
      cex.lab=1.2
 )
> lines(tt, surv.cens.MC, type = "s", col = "red", lwd = 1.5)
> lines(tt, surv.IPCW.Stab.MC, type = "s", col = "blue", lwd = 1.5)
> legend(legend = c("Real",
                    "Standard method",
                    "IPCW"),
        col = c("green", "red", "blue"),
        lty = 1,
        x = "topright",
        cex = 1.2,
        bty = "n"
 )
> dev.off()
```

Summarize model parameters by taking means.

```
> beta.real.MC <- colMeans(beta.real)
> beta.cens.MC <- colMeans(beta.cens)
```

```
> beta.IPCW.StabTrunc.MC <- colMeans(beta.IPCW.StabTrunc)
> beta.IPCW.UnStab.MC <- colMeans(beta.IPCW.UnStab)
> beta.IPCW.Stab.MC <- colMeans(beta.IPCW.Stab)
> phi.IPCW.MC <- colMeans(phi.IPCW)
```

Save results.

```
> save(surv.real.MC, beta.real.MC,
      surv.cens.MC,  beta.real.MC,
      surv.IPCW.UnStab.MC, beta.IPCW.UnStab.MC,
      surv.IPCW.Stab.MC, beta.IPCW.Stab.MC,
      phi.IPCW.MC,
      file = "C:\\Users\\Sanne\\Documents\\School\\LU - MA2\\Scriptie\\SimulatiesDef\\KMsimu
 )
```

Code used to plot the boxplot.

```
> xnames <- c("35%",
              "50%",
              "65%")
> png("Beta1VaryPercentage.png", width = 500, height = 400)
> par(mar = c(7,5,4,2))
> boxplot(betaStab65[,1],
          xlim = c(0,4),
          ylim = c(-0.5,1.5),
          lwd = 1,
          outline = FALSE,
          cex.axis = 1.5,
          yaxt="n",
          at = 3.75,
          col = "darkblue"


 )
> axis(1, at=c(0.5, 2, 3.5),
      labels=xnames,
      padj = "0",
      las=1,
      cex.axis = 1.5)
> axis(2,
      cex.axis = 1.5)
> legend(legend = c("Unstabilized Weights",
                    "Stabilized Weights"),
        fill = c("lightblue", "darkblue"),
        lty = 1,
```

```
        x = "bottomleft",
        cex = 1,
        bty = "n",
        seg.len = 0
 )
> boxplot(betaUnStab65[,1],
         lwd = 1,
         cex.lab = 1.5,
         add = TRUE,
         at = 3.25,
         outline = FALSE,
         yaxt="n",
         col = "lightblue"
 )
> boxplot(betaStab50[,1],
         lwd = 1,
         cex.lab = 1.5,
         add = TRUE,
         at = 2.25,
         outline = FALSE,
         yaxt="n",
         col = "darkblue"
 )
> boxplot(betaUnStab50[,1],
         lwd = 1,
         cex.lab = 1.5,
         add = TRUE,
         at = 1.75,
         outline = FALSE,
         yaxt="n",
         col = "lightblue"
 )
> boxplot(betaStab35[,1],
         lwd = 1,
         cex.lab = 1.5,
         add = TRUE,
         at = 0.75,
         outline = FALSE,
         yaxt="n",
         col = "darkblue"
 )
> boxplot(betaUnStab35[,1],
         lwd = 1,
```

```
         cex.lab = 1.5,
         add = TRUE,
         at = 0.25,
         outline = FALSE,
         yaxt="n",
         col = "lightblue"
 )
> dev.off()
> png("Phi2VaryN.png", width = 600, height = 400)
> par(mar = c(7,5,4,2))
> boxplot(phin100[,2],
         xlim = c(0,5),
         lwd = 1,
         outline = FALSE,
         cex.axis = 1.5,
         yaxt="n"

 )
> axis(1, at=1:4,
      labels=xnames,
      padj = "0",
      las=2,
      cex.axis = 1.5)
> axis(2,
      cex.axis = 1.5)
> boxplot(phin250[,2],
         xlim = c(0,5),
         lwd = 1,
         cex.lab = 1.5,
         add = TRUE,
         at = 2,
         outline = FALSE,
         yaxt="n"


 )
> boxplot(phin500[,2],
         xlim = c(0,5),
         lwd = 1,
         cex.lab = 1.5,
         add = TRUE,
         at = 3,
         outline = FALSE,
```

```
          yaxt="n"


 )
> boxplot(phin2500[,2],
          xlim = c(0,5),
          lwd = 1,
          cex.lab = 1.5,
          add = TRUE,
          at = 4,
          outline = FALSE,
          yaxt="n"


 )
> dev.off()
```

Bibliography

[1] O. O. Aalen. Nonparametric inference for a family of counting processes. *Annals of Statistics*, 6:701–726, 1978.

[2] P. K. Andersen and R. D. Gill. Cox's regression model for counting processes: A large sample study. *The Annals of Statistics*, 10(4):pp. 1100–1120, 1982.

[3] Ralf Bender, Thomas Augustin, and Maria Blettner. Generating survival times to simulate cox proportional hazards models. *Statistics in Medicine*, 24(11):1713–1723, 2005.

[4] Roel Braekers and Noël Veraverbeke. Cox's regression model under partially informative censoring. *Communications in Statistics - Theory and Methods*, 34(8): 1793–1811, 2005.

[5] D. Burr. A comparison of certain bootstrap confidence intervals in the cox model. *Journal of the American Statistical Association*, 89:1290–1302, 1994.

[6] D. R. Cox. Regression models and life tables (with discussion). *Journal of the Royal Statistical Society B*, 34:187–220, 1972.

[7] E. de Beurs and F. G. Zitman. Routine outcome monitoring: Het meten van therapie-effect in de klinische praktijk met webbased software. *Maandblad Geestelijke Volksgezondheid*, 62:13–28, 2007.

[8] Bradley Efron and Robert J. Tibshirani. *An Introduction to the Bootstrap*. Monographs on statistics and applied probability. Chapman & Hall, 1993.

[9] R. Gentleman and Alain Vandal. *Icens: NPMLE for Censored and Truncated Data*. R package version 1.26.0.

[10] G. Gómez, M. Luz Calle, R. Oller, and K. Langohr. Tutorial on methods for interval-censored data and their implementation in r. *Statistical Modelling*, 9:259–297, 2009.

[11] P. Groeneboom and J. A. Wellner. *Information bounds and nonparametric maximum likelihood estimation*. Birkhäuser Verlag, 1992.

[12] V. Henschel, C. Heiß, and U. Mansmann. intcox: Compendium to apply the iterative convex minorant algorithm to interval censored data, June 2009.

[13] Volkmar Henschel and Ulrich Mansmann. *intcox: Iterated Convex Minorant Algorithm for interval censored event data*, 2013. URL `http://CRAN.R-project.org/package=intcox`. R package version 0.9.3.

[14] Xuelin Huang and Robert A. Wolfe. A frailty model for informative censoring. *Biometrics*, 58(3):510–520, September 2002.

[15] E. L. Kaplan and P. Meier. Nonparametric estimation from incomplete observations. *Journal of the American Statistical Association*, 53:457–481, 1958.

[16] John P. Klein and Melvin L. Moeschberger. *Survival Analysis: Techniques for Censored and Truncated Data*. Statistics for Biology and Health. Springer, second edition edition, 2003.

[17] David G. Kleinbaum and Mitchel Klein. *Survival Analysis: A Self-Learning Text*. Statistics for Biology and Health. Springer, third edition edition, 2012.

[18] W. Nelson. Theory and applications of hazard plotting for censored failure data. *Technometrics*, 14:945–965, 1972.

[19] W. Pan. Extending the iterative convex minorant algorithm to the cox model for interval censored data. *Journal of Computational and Graphical Statistics*, 8:109–120, 1999.

[20] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2008. URL `http://www.R-project.org`.

[21] J. M. Robins. Information recovery and bias adjustment in proportional hazards regression analysis of randomized trials using surrogate markers. *Proceedings of the Biopharmaceutical Section, American Statistical Association*, pages 24–33, 1993.

[22] J. M. Robins and D. M. Finkelstein. Correcting for noncompliance and dependent censoring in an aids clinical trial with inverse probability of censoring weighted (ipcw) log-rank tests. *Biometrics*, 56:779–788, 2000.

[23] James M. Robins and Andrea Rotnitzky. Recovery of information and adjustment for dependent censoring using surrogate markers. In *AIDS Epidemiology*, pages 297–331. Birkhäuser Boston, 1992.

[24] A. Schat, M.S. van Noorden, M.J. Noom, E.J. Giltay, N.J.A. van der Wee, R.R.J.M. Vermeiren, and F.G. Zitman. Predictors of outcome in outpatients with anxiety disorders: The leiden routine outcome monitoring study. *Journal of Psychiatric Research*, 47(12):1876 – 1885, 2013.

[25] Shaun R Seaman and Ian R White. Review of inverse probability weighting for dealing with missing data. *Statistical Methods in Medical Research*, 22(3):278–295, 2013.

[26] Terry M. Therneau and Patricia M. Grambsch. *Modeling Survival Data: Extending the Cox Model*. Springer, New York, 2000.

[27] Terry M Therneau. *A Package for Survival Analysis in S*, 2013. URL `http://CRAN.R-project.org/package=survival`. R package version 2.37-4.

[28] B. Turnbull. The empirical distribution function with arbitrarily grouped, censored and truncated data. *Journal of the Royal Statistical Society, Series B*, 38:290–295, 1976.

[29] M. S. van Noorden, E. M. van Fenema, N. J. A. van der Wee, Y. R. van Rood, I. V. E. Calier, F. G. Zitman, and E. J. Giltay. Predicting outcomes of mood, anxiety and somatoform disorders: The leiden routine outcome monitoring study. *Journal of Affective Disorders*, 142:122–131, 2012.

[30] J. A. Wellner and Y. Zahn. A hybrid algorithm for computation of the nonparametric maximum likelihood estimator from censored data. *Journal of the American Statistical Society*, 92:945–959, 1997.

[31] H.U. Wittchen, F. Jacobi, J. Rehm, A. Gustavsson, M. Svensson, B. Jönsson, J. Olesen, C. Allgulander, J. Alonso, C. Faravelli, L. Fratiglioni, P. Jennum, R. Lieb, A. Maercker, J. van Os, M. Preisig, L. Salvador-Carulla, R. Simon, and H.-C. Steinhausen. The size and burden of mental disorders and other disorders of the brain in europe 2010. *European Neuropsychopharmacology*, 21(9):655 – 679, 2011.

[32] Ming Zheng and Klein J. P. A self-consistent estimator of marginal survival functions based on dependent competing risk data and an assumed copula. *Communications in statistics. Theory and methods*, 23(8):2299–2311, 1994.

Index