Estimation from Interval-censored Time-to-event Data:

Method Comparison by Simulation based on GALLIUM Study for Follicular Lymphoma

---

Master Thesis in Biostatistics (STA495)

by

Monika Hebeisen
00-910-240

supervised by

Dr. Kaspar Rufibach, Roche Biostatistics Basel
Prof. Torsten Hothorn, University of Zurich

Zurich, October 2014

Acknowledgements

First of all, I would like to thank Dr. Kaspar Rufibach for his kind and thorough support during the whole project and his in-depth introduction into the daily life of a statistician at Roche. I could learn a lot from him, also related to the way of thinking of a statistician. Then I would like to thank Prof. Hothorn for his agreement to be my official supervisor from University of Zurich. A special thanks goes to Dr. Eva Furrer, who supported my wish to do an internship at Roche and always had an open ear for all kinds of questions. Thank you Jipcy Amador, for the very helpful discussions about some topics of my thesis. I would like to express my gratitude also to Hans Ulrich Burger and Ulrich Beyer from Roche for giving me the opportunity to do my internship at Roche Clinical Biostatistics in Basel. My office mates at Roche, Ludger Banken, Giuseppe Palermo, Markus Niggli, Simona Rossomanno and Paul Delmar, get a special "thank you" for their pleasant company and the good working environment.
A sincere thank goes to David Dejardin for all the enriching discussions about survival analysis and his function to calculate intervalcensored hazard ratio, to Shamil Sadhikov for the interesting general discussions about statistics and to Volkmar Henschel, who supported me with the usage of his software package. I am grateful as well to Giuseppe Palermo for his help with running simulations on the unix server and to Emanuela Pozzi. She gave me the opportunity to apply my functions to real data.
Then I would like to thank Lisa Squassante for her refreshing company in the early mornings, and... I could add many more names. Finally: thanks to all the other statisticians from Clinical Biostatistics in Basel for the friendly talks and the refreshing company in the breaks.

Contents

Chapter 1

Introduction

1.1 Oncology clinical trials

1.1.1 Cancer treatment

Cancer is a class of diseases characterized by the uncontrolled growth of a single cell. The many resulting cancer cells form a tumor and might invade neighbouring tissue and spread to other parts of the body. To kill all cancer cells is the aim of cancer treatment. Traditionally, this is tried to be achieved by surgery, radiotherapy and/or chemotherapy, depending on the cell type affected [8, Chapter 9.5]. But these treatments have some restrictions. They often do not reach all cancer cells and/or do also harm normal cells. To improve therapy specificity, target-specific drugs have been developed [13]. These drugs can either be antibodies or small molecules. They are normally given in addition to traditional treatments, but combinations of them might replace chemotherapy in the future.

To compare different treatments and to monitor treatment success, in many indications doctors look at tumor size changes. Tumor size is often measured by imaging techniques, such as computer tomography (CT) or positron emission tomography (PET). Depending on the size change of tumor lesions, patients are then assigned to response categories that are relevant for determination of treatment success and further treatments. These categories are for most solid tumors and lymphoma *complete response (CR)* when the lesions disappear completely, *partial response (PR)* when their size remarkably decreases, *stable disease (SD)* when they stay the same size, and *progressive disease (PD)* when the lesions grow or new lesions appear [7, 4].

1.1.2 Study endpoints

Phase III oncology clinical trials are normally randomized and controlled, and ideally the endpoint is overall survival. However, when treatment is effective and for indolent (slowly progressing) cancer types (eg. Follicular Lymphoma), overall survival might be very long. In consequence, this leads to large studies with either very long study time or huge study size.

Therefore other, intermediate endpoints, called *surrogate endpoints*, have been established as primary endpoints. In cancer, surrogate endpoints are often based on tumor

assessments and the treatment response criteria that were described in Section 1.1.1.

Two common surrogate endpoints based on tumor progression are time to progression (TTP) and progression-free survival (PFS). TTP is defined as the time from randomization to tumor progression. If a patient in a TTP study dies before progression, he is right-censored at the day of death. PFS is a *composite endpoint* made up from TTP and overall survival (OS), that means death is considered an event [15]. Composite endpoints have some advantages over single endpoints. As they capture both intermediate events and death, they increase study power. Additionally, they eliminate the problem of the competing risk of death by including death events in the analysis [5, p. 66]. The Food and drug administration of USA (FDA) prefers the PFS endpoint to the TTP endpoint [15].

Surrogate endpoints such as TTP and PFS have one disadvantage compared to OS endpoints. The event is based on tumor progression. Often, progression might be asymptomatic, that means not accompanied by any symptoms that could be reported by the patient, and it can only be detected in a hospital visit via a tumor assessment. Therefore the event date cannot be determined exactly, but is just known to have happened between two assessment dates. This type of event is called *interval-censored*, that means, the event of interest is just known to lie inside a certain time interval.

1.1.3 Censoring of observations

Progression events are one example of censoring in oncology clinical trials with PFS endpoint, but there are some more.

A few patients drop out from the study and are therefore *right-censored*. These patients decide to not continue with treatment and/or assessment visits and are then censored at the last assessment date. By doing this we use the information that they did not have an event until that date and might have one anytime in the future.

All the patients that did not have an event yet when the study is analysed are *right-censored* at the last assessment date before the analysis timepoint. So we know that their event time is larger than that date. This type of censoring is called administrative censoring.

As a generalization of the progression events, any kind of events due to tumor assessments are *interval-censored* between two assessment dates. We only know that this kind of event has happened in between the two assessment visits, but not when exactly it has happened.

If time-to-event data was not censored, it would be straightforward to analyze it with standard statistical procedures for continuous variables. But there are always censored events in time-to-event clinical trials. This is due to withdrawal of consent and because we cannot wait until all patients in the study have an event. Therefore more special survival analysis procedures have to be used.

A general assumption in analysis of censored data is that the censoring times and event times are independent (also called noninformative censoring) [21]. The methods used in this thesis all imply this assumption. For interval-censored data in addition the interval-generating process has to be independent of event and rightcensoring times [3, Chapter 1]. In practice the independence assumption of censoring and event times can hardly be proven and might not always be correct.

The most established survival analysis methods consider just right-censored data. If one wants to consider interval-censored data, one has to use more general methods.

1.2 Project aim

1.2.1 Standard analysis of a time-to-event study

Typically, oncology clinical trials are analyzed with standard survival analysis methods such as Kaplan-Meier estimates, log-rank test, and Cox regression [21]. These methods are well-established and used for decades, but they only consider right-censoring, no interval-censoring. Therefore all events are assumed to have happened at an exact date. For progression events, the assessment date where the event was first detected is used as exact date. Even if this approach generally overestimates endpoints such as PFS (we impute the right end of the censoring interval, which is the latest possible event timepoint), this procedure is generally accepted, also from health authorities [3, Chapter 10].

But some bias to longer progression times might be introduced here. This bias tends to be bigger the longer the intra-assessment periods are [3, Chapter 10]. With long intraassessment and study periods this bias might be substantial.

When we consider only plausible cases this bias can be reduced by imputing the middle of the intra-assessment period as an exact event instead of imputing the end [3, Chapter 10]. This is done sometimes, but the result is only unbiased if the assumption of a constant event probability (density, not hazard) in that interval holds. This assumption implies a linear survival function, which is not very plausible.

It would be best to use the appropriate methods that also consider interval-censoring. Some of these methods, like the Turnbull estimator for survival functions [39], are already available for many years. In recent years many more methods for interval-censored data have been established [34] and implemented in statistical software. The methods for interval-censored data that correspond to the above-mentioned methods for right-censored data are the Turnbull estimator [39], generalized log-rank test [3, Chapter 14] and "generalized Cox regression" [18].

Methods for interval-censored data are still not used regularly in the pharmaceutical industry. Nevertheless, there have been some first requests from health authorities to provide interval-censored analysis as a sensitivity analysis [41, slide 6]. Interval-censored analysis has also been recommended as a sensitivity analysis of PFS endpoints by the PhRMA working group that is sponsored by the American Pharmaceutical Research and Manufacturers Association [31]. So this type of analysis might be used more often in clinical studies in the future.

1.2.2 Aim of thesis

Roche would like to improve its knowledge of interval-censoring methods in general and of their behaviour compared to the well-known right-censoring methods in particular. The aim of this thesis is to establish a framework of survival analysis procedures for interval-censored data in the software `R` [29] and to test it with some realistically simulated datasets. With these simulations effects of right- and interval-censoring are compared for different scenarios and their behaviour is better understood.

Finally, realistic data generation for simulation of other studies is established in `R`, ready to use by other statisticians at Roche. The established framework should also be usable with clinical trial data to be ready for further health authority requests.

1.2.3 Project outline

This aim is achieved in a three-step procedure.

First, some survival data is simulated and right- and interval-censorings applied to it. Simulation parameters are chosen appropriately to mimic a clinical trial in lymphoma.

Then the estimations of the survival functions, of the p-values of the log-rank tests and of the hazard ratios of the Cox regression are done with both right- and interval-censoring methods. The survival function is estimated with both non-parametric and parametric procedures (by Weibull distribution).

Finally, the estimates are compared to the true values simulated from. Survival function estimates are compared at certain times and quantiles by *mean squared error (MSE)* and *bias*. Log-rank tests are compared by *power* and *Type I-error* and hazard ratios by *MSE* and *bias* as well.

This whole simulation procedure is repeated with some specific trial parameters changed and then results are compared. In this way we can find out with which type of trial we can expect big differences between right-censoring and interval-censoring analysis.

Also simulation scenarios are chosen that do not fulfil the uninformative censoring criterion (see 1.1.3) because their behaviour differs systematically across arms. Some bias might be introduced when these scenarios are analyzed [3, Chapter 10]. By testing these scenarios the robustness of the right- and intervalcensoring methods with regards to informative censoring is compared. The final goal of all these tests is to assess which method is more suitable for the analysis of what kind of data.

1.3 Example of clinical trial: GALLIUM

We would like to simulate data that is close to real clinical trial data. So we use the sample size calculation parameters of a real clinical trial called GALLIUM [1] as a basis of our simulations. The GALLIUM trial is explained more thoroughly in the next section.

1.3.1 Key features

The trial is a phase III trial designed for approval of the drug in this new indication. It is a two-arm, 1:1 randomized, open-label study that compares the control treatment chemotherapy plus Rituximab (an established antibody drug) to the new treatment chemotherapy plus Gazyva (a new antibody drug). The study started in 2011 and is expected to complete in 2017. It includes a total of 1400 patients in about 200 sites in 70 countries worldwide. The patients have untreated, advanced, indolent Non-Hodgkin's lymphoma, about 1200 of them the subtype follicular lymphoma, the other 200 patients have the subtype marginal zone lymphoma. The primary endpoint of the study is PFS of the follicular lymphoma patients.

1.3.2 Disease

Lymphoma is the cancer of the lymph cells. These are the cells of the immune system. There are diagnosed about 20 new lymphoma cases per 100000 people per year [22], about 7 - 10% of all cancer cases.

Follicular lymphoma is a cancer of B-cells in follicles of lymph nodes. B-cells are the antibody-producing cells of the immune system. Follicular lymphoma accounts for about 20% of all lymphoma cases. In marginal zone lymphoma another subtype of B-cells is affected, that is present in lymph nodes, spleen, stomach and other organs. 10% of lymphoma cases are of this subtype [22].

Follicular lymphoma is a disease of the elderly. The median age at diagnosis is 60 years [40]. Mostly it is accompanied by a prominent genetic change, overexpression of the oncogene Bcl-2. It is an indolent disease, which means slowly progressing. When patients have no symptoms and the cancer lesions are small, patients are sometimes not treated immediately but kept in "watch and wait" state, sometimes for years. Patients with advanced disease are treated with chemotherapy plus Rituximab, which is the control treatment of this study. Median PFS with this treatment is six years [1]. This is quite long and typical for indolent diseases. In summary, follicular lymphoma is highly treatable, but ultimately uncurable [40].

1.3.3 Treatment regimen

Follicular lymphoma is regularly treated with different chemotherapeutic agents. Therefore three different agents, Bendamustine, CHOP and CVP can be used in the GALLIUM study. Sites can choose the one they prefer. These chemotherapies are given intravenously in the hospital. They kill most proliferating cells in the body, not only cancer cells. So they are not that specific, but still effective.

The chemotherapy is combined with an anti-CD20-antibody that binds to antigens on the surface of B-cells and induces cell death. All B-cells are killed here, not just the cancerous ones. The antibodies are given intravenously, together with the chemotherapy. In the study, the standard antibody Rituximab (marketed as MabThera in Europe) that is a first generation antibody drug and on the market for 16 years is used in the control group. The new antibody marketed as Gazyva (with product name Obinutuzumab), a glycoengineered, humanized antibody of the second generation is tested in the treatment group.

Treatment response is measured by imaging with CT or PET scans. Six tumor lesions (mostly in lymph nodes) are assessed regularly by size measurements and the response categories described in Section 1.1.1 are applied [4].

1.3.4 Study phases

The GALLIUM study has three phases.

The first one is the induction treatment phase that lasts for about 6 months and contains 6 to 8 cycles of chemotherapy plus antibody and 3 tumor assessments, one of them being a baseline assessment before start of treatment.

The second phase is the maintenance treatment or observation phase. Patients with complete or partial response after induction will receive maintenance treatment, which is

12 more cycles of antibody during 2 years. All patients (also those with stable disease who get into observation phase at this time) will have 5 tumor assessments in this period.

The third phase is the follow-up phase that lasts for a maximum of 5 years. In the first 3 years there are assessments every 6 months, followed by yearly assessments.

Exact treatment and assessment dates are shown for CVP chemotherapy in Figure 1.1. In the first plot, one can also see that in the first of the eight three-weekly chemotherapy cycles, there are two more Gazyva infusions, on day 8 and 15, due to results from pharmacokinetic studies. Rituximab patients do not need an infusion on these days. And because there are no placebo infusions done for ethical reasons, this study cannot be blinded.

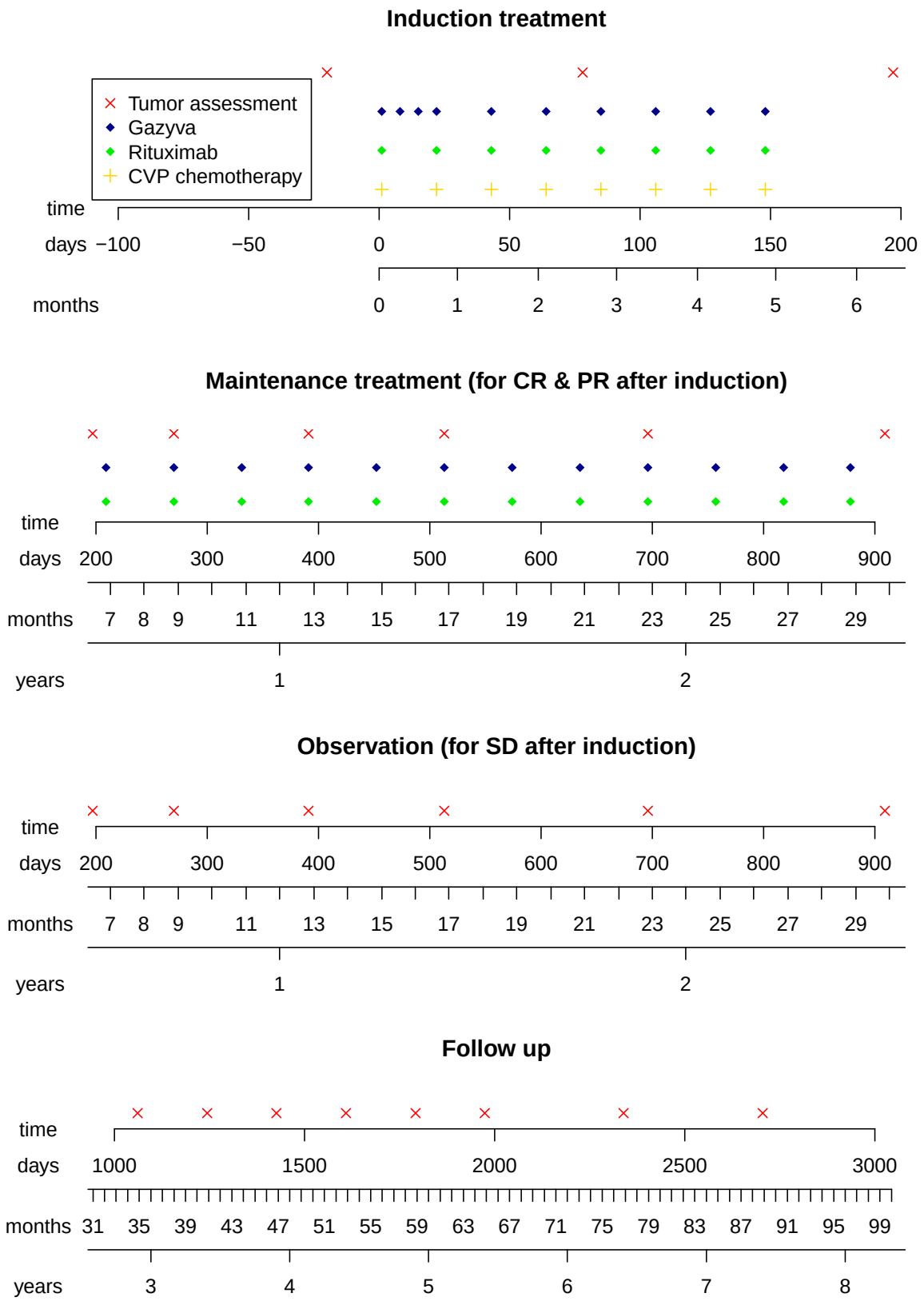

Figure 1.1: Three phases of GALLIUM study: Induction, maintenance and follow-up. Example of treatment with CVP chemotherapy and either Rituximab or Gazyva antibody. Tumor assessment dates are indicated with red crosses

Chapter 2

Methodology

2.1 Survival analysis

Statistical analysis of time-to-event data is called survival analysis, also when the event is not death. It has some characteristics that differ from standard statistical analysis for continuous variables, in addition to the different terminology.

We already got to know some special characteristics of survival analysis, like the consideration of censored observations in Section 1.1.3 and the treatment of composite endpoints in Section 1.1.2 of the introductory chapter. There are some more that are explained here.

First, event times are continuous random variables that can only be non-negative ($T \geq 0$). Any distribution function that is used for description of event times has to consider this.

Secondly, in survival analysis the standard representation of a distribution is not the cumulative distribution function (cdf) $F(t) = P(T \leq t)$, and the corresponding density function $f(t) = \partial F(t)/\partial t$. Instead, the survival function $S(t) = 1 - F(T) = P(T > t)$ and the hazard function $h(t) = f(t)/S(t)$ are used for analysis and display of time-to-event data. This for historical reasons and because they are interpretable in an intuitive way as the probability of an event after a certain time (for S) and the instantaneous risk of an event (for h) [21, Chapter 2].

2.2 Estimation of survival functions

Survival functions of the study arms are a very important result of oncology clinical trials, especially the timepoints of the median survival. Therefore they are estimated first in my simulation study, and they are estimated in two ways. In addition to nonparametric estimation, also the parametric case is considered.

2.2.1 Parametric estimation: Weibull

Parametric models have the advantage of being easily applicable to both right- and interval-censored data. Maximum likelihood estimation results in parameter estimates and thus estimated distributions including standard errors based on Maximum likelihood theory [17] that can be used for any kind of further inference. The disadvantage is the

strong assumption on the distribution of the data that is taken when a parametric model is used. If that assumption does not hold, any further inference is biased.

Weibull distributon

As a realistic representation of a survival function for the GALLIUM study, the Weibull distribution was chosen. It has two parameters and can be represented in the following way [17, appendix]:

$$T \sim Wb(\alpha, \mu), \qquad \alpha, \mu > 0$$

$$F(t) = 1 - \exp\left[-\left(\frac{1}{\mu} \cdot t\right)^{\alpha}\right], \quad t \geq 0 \qquad\qquad S(t) = 1 - F(t) = \exp\left[-\left(\frac{1}{\mu} \cdot t\right)^{\alpha}\right]$$

$$f(t) = \frac{1}{\mu} \cdot \alpha \cdot \left(\frac{1}{\mu} \cdot t\right)^{\alpha-1} \cdot \exp\left[-\left(\frac{1}{\mu} \cdot t\right)^{\alpha}\right] \qquad h(t) = \frac{1}{\mu} \cdot \alpha \cdot \left(\frac{1}{\mu} \cdot t\right)^{\alpha-1} \quad .$$

The Weibull distribution has the advantage of being adaptable to many different shapes and at the same time fulfilling the proportional hazard assumption. This assumption is convenient, because it allows calculation of a hazard ratio that is constant over time.

Likelihood functions

Parametric estimation is done easily by applying the Maximum Likelihood (ML) framework to the chosen distribution. So the likelihood is the product of the densities at the observed event times:

$$\mathbf{L} = \prod_{i=1}^{n} f(T_i) \quad .$$

The likelihood depends on the parameters of the distribution, and we want to estimate these parameters in likelihood inference, but for the sake of simplicity, a symbol for these parameters was not included in the equation above and in the next equations. The above equation is only valid if all events are completely observed, that means if there are no censored observations. Censored observations have to be considered in a different way.

In the **right-censoring** framework, one can observe exact events and right-censorings. They are displayed with two variables [21, Chapter 3]:

time $\qquad \mathbf{T_{RC}}$

indicator $\qquad \boldsymbol{\delta} = \begin{cases} 1 & \text{if time is event,} \\ 0 & \text{if time is censoring} \end{cases} \quad .$

For right-censored observations the event could happen anytime after the censoring. To find the corresponding likelihood, the probabilities of all possible event times are integrated, that means all times bigger than the censoring time. This integrated probability corresponds to the survival probability at the censoring time.

To find the likelihood of the whole sample, again the product of the likelihoods of the observations has to be formed. To choose the appropriate likelihood for each observation the censoring indicator δ comes now into place. The likelihood formula for right-censored and exact observations is [21, Chapter 3]:

$$\mathbf{L} = \prod_{i=1}^{n} f(T_{RC\,i})^{\delta_i} \cdot S(T_{RC\,i})^{1-\delta_i} \quad .$$

It is only valid if the censoring and event-generating mechanisms are independent.

Interval-censored observations are represented by two times, T_L for the left border of the censoring interval and T_R for the right border. Exact events and right-censorings are special cases in this framework and therefore can be represented by two times as well. For exact events $T_L = T_R = T_{RC}$ and for right-censorings $T_L = T_{RC}, \quad T_R = \infty$.

The likelihood of interval-censored observations can be explained in a similar way as the one of the right-censored observations. One should find the probability of all possible event times, which for an interval-censored event corresponds to the value of the survival function at the left border minus the value of the survival function at the right border. This probability can be found via $P(T_L \leq T \leq T_R) = F(T_R) - F(T_L) = S(T_L) - S(T_R)$. It is only valid if the censoring and event generating process are independent.

In this version of the likelihood function also exact events and right-censored observations can be inserted. The resulting likelihood function is [21, Chapter 3]:

$$\mathbf{L} = \prod_{i=1}^{n} \left[S(T_{L\,i}) - S(T_{R\,i})\right] \quad .$$

Insertion of the corresponding Weibull distribution functions into the likelihood functions results in the likelihood functions that we used in this thesis. These are:

for right-censoring:

$$\begin{aligned}\mathbf{L}(\alpha, \mu) \quad = \prod_{i=1}^{n} \quad & \left\{ \frac{1}{\mu} \cdot \alpha \cdot \left(\frac{1}{\mu} \cdot T_{RC\,i} \right)^{\alpha-1} \cdot \exp\left[- \left(\frac{1}{\mu} \cdot T_{RC\,i} \right)^{\alpha} \right] \right\}^{\delta_i} \\ & \cdot \left\{ \exp\left[- \left(\frac{1}{\mu} \cdot T_{RC\,i} \right)^{\alpha} \right] \right\}^{1-\delta_i} \quad ,\end{aligned}$$

for interval-censoring:

$$\mathbf{L}(\alpha, \mu) = \prod_{i=1}^{n} \left\{ \exp\left[- \left(\frac{1}{\mu} \cdot T_{L\,i} \right)^{\alpha} \right] - \exp\left[- \left(\frac{1}{\mu} \cdot T_{R\,i} \right)^{\alpha} \right] \right\} \quad .$$

These likelihood functions are maximized with the observed times inserted, what results in ML estimates of the Weibull parameters, $\hat{\alpha}_{ML}$ and $\hat{\mu}_{ML}$ for these observations. The corresponding survival function of the Weibull distribution is taken to find estimates at certain times and quantiles.

Implementation in R [29]

Parametric estimation of distributions for time-to-event data is available in `R` packages `survival` [37, 38] and `fitdistrplus` [6]. With `survival` package, the data is put in a `Surv` object that is called in the `survreg` function. In `fitdistrplus` package the `fitdistcens` function is used.

The two functions give sufficiently close results. They can both be used for right- and interval-censored data, but in the `fitdistcens` function the data can only be inserted in the interval-censoring representation with two times.

In the simulations in this thesis the `survreg` function is used for estimation of parametric survival functions. Estimation of survival quantiles and times is summarized in the here-defined `SurvWeibUncens`, `SurvWeibRightcens`, `SurvWeibIntervalcens`, `TimeWeibUncens`, `TimeWeibRightcens` and `TimeWeibIntervalcens` functions that are shown in the Appendix 8.2.2.

2.2.2 Nonparametric estimation: NPMLE

Generalizations of the empirical cumulative distribution function (ECDF) that also consider censorings are normally used as nonparametric estimators in the survival analysis context. These nonparametric estimators are quite popular in survival analysis because they do not make any assumptions on the shape of the function. But their disadvantage is that they are step functions and inference for them is less immediately available compared to parametric functions.

One can also show that these estimators are a special type of Maximum Likelihood (ML) estimators. So they are also called nonparametric maximum likelihood estimators (NPMLE) [34, Chapter 3].

For right-censored data: Kaplan-Meier

The nonparametric estimator used for right-censored data is called Kaplan-Meier estimator or product-limit estimator [19, 20]. This estimator can be derived from probability ratios that are used to formulate a likelihood function whose maximization results in ML estimates in the form of counts. The usual definition of the Kaplan-Meier estimator is with these counts.

Theory To start the explanations we have to sort the observed event times ($T_{RC\,i},\ \delta_i = 1,\ i \in \{1, 2, ..., n\}$) in increasing order and get unique event times $t_j,\ j = \{1, 2, ..., s\}$ such that

$$t_1 < t_2 < t_3 \quad ... < t_s \quad .$$

There can be more than one observation with the same time. Then these observations are tied and their number per timepoint is recorded in the variable d_j with $j = 1, 2, 3, ..., s$.

The event times t_j are considered to be fixed. They define the location of the steps of the step function. That means the survival function does only decrease at these event times and is constant in between **??**. The value of the survival function at one of the event times t_j is:

$$
\begin{aligned}
S(t_j) &= P(T > t_j) = P(T \geq t_{j+1}) \\
&= \frac{P(T \geq t_{j+1})}{P(T \geq t_j)} \cdot \frac{P(T \geq t_j)}{P(T \geq t_{j-1})} \cdot ... \cdot \frac{P(T \geq t_2)}{P(T \geq t_1)} \cdot P(T \geq t_1) \\
&= \left[1 - \frac{P(T = t_j)}{P(T \geq t_j)}\right] \cdot \left[1 - \frac{P(T = t_{j-1})}{P(T \geq t_{j-1})}\right] \cdot ... \cdot \left[1 - \frac{P(T = t_1)}{P(T \geq t_1)}\right] \\
&= \prod_{l=1}^{j} \left[1 - \frac{P(T = t_l)}{P(T \geq t_l)}\right] =: \prod_{l=1}^{j} (1 - \lambda_l) \quad . \qquad (2.1)
\end{aligned}
$$

In a next step the censoring times of the censored observations ($T_{RC\,i},\ \delta_i = 0,\ i \in \{1, 2, ..., n\}$) are sorted, per event time interval $[t_j, t_{j+1})$:

$$
t_{jk},\ k = \{1, 2, 3,\ ..., c_j\} \quad ,
$$

where c_j is the number of censored observations in the j-th time-interval. The survival probability at an individual censoring time is:

$$
P(T > t_{jk}) = S(t_{jk}) = S(t_j) \quad . \qquad (2.2)
$$

This because of the constant survival function at non-event times.

At the same time the likelihood contribution of an event time can be written in terms of the survival function:

$$
f(t_j) = P(T = t_j) = P(T \geq t_j) - P(T > t_j) = S(t_{j-1}) - S(t_j) \quad . \qquad (2.3)
$$

With knowledge of (2.1), (2.2) and (2.3) one can reformulate the likelihood function in terms of event times as follows:

$$
\begin{aligned}
\mathbf{L}(\lambda_1, ..., \lambda_s) &= \prod_{j=1}^{s} \left[S(t_{j-1}) - S(t_j)\right]^{d_j} \cdot S(t_j)^{c_j} \\
&= \prod_{j=1}^{s} \left[\prod_{l=1}^{j-1}(1 - \lambda_l) - \prod_{l=1}^{j}(1 - \lambda_l)\right]^{d_j} \cdot \left[\prod_{l=1}^{j}(1 - \lambda_l)\right]^{c_j} \\
&= \prod_{j=1}^{s} \left[\lambda_j^{d_j} (1 - \lambda_j)^{\left[(\sum_{i=j}^{s} d_i + c_i) - d_j\right]}\right] \quad .
\end{aligned}
$$

Maximizing the likelihood leads to the following ML estimate of λ_j (see Appendix 8.1 for details):

$$
\hat{\lambda}_{j;ML} = \frac{d_j}{\sum_{i=j}^{s} d_i + c_i} = \frac{d_j}{r_j} =: \frac{\#\text{ of events at } t_j}{\#\text{ at risk just before } t_j} \quad .
$$

d_j and r_j are two numbers that can be calculated at every event time t_j of the dataset. They represent the number of events (d_j) and the risk set (r_j) and can be summarized for any right-censored dataset with times $T_{RC\,i} = t_i,\ i = \{1, 2, 3, ..., n\}$ as follows:

$$d_j = \sum_{i=1}^{n} \begin{cases} 1 & \text{if } t_i = t_j \text{ and } \delta_i = 1, \\ 0 & \text{otherwise} \end{cases}$$

$$r_j = \sum_{i=1}^{n} \begin{cases} 1 & \text{if } t_i \geq t_j, \\ 0 & \text{otherwise} \end{cases} .$$

The resulting Kaplan-Meier estimator for any set of t is defined as [21, Chapter 4]:

$$\widehat{S(t)} = \begin{cases} 1 & \text{for } t < t_1, \\ \prod_{t_j \leq t} \left(1 - \frac{d_j}{r_j}\right) & \text{for } t \geq t_1 \\ \text{not defined} & \text{for } t > t_{\max} \end{cases} .$$

Example Typically to compute Kaplan-Meier survival function estimation a so-called risk table is generated. This lists the time (t_j), number at risk (r_j), number of events (d_j) and Kaplan-Meier estimator ($\widehat{S(t_j)}$) at all numbered event times (j).

We show this risk table in Table 2.1 for an example dataset with 15 patients which is the same dataset as the one used in Figure 3.1 in the simulation setup section.

The corresponding Kaplan-Meier estimate is shown in Figure 2.1. In this dataset the last patient was censored. That is the reason why the curve does end above zero survival probability. For all timepoints after the last time the function is not defined. Therefore no survival probabilities below the value of the last time are estimated.

j	t_j	r_j	d_j	$S(t_j)$
1	71	15	1	0.933
2	74	14	1	0.867
3	169	13	1	0.800
4	344	12	1	0.733
5	353	11	1	0.667
6	382	10	1	0.600
7	504	8	1	0.525
8	579	7	1	0.450
9	645	6	1	0.375
10	754	5	1	0.300
11	829	4	1	0.225
12	971	2	1	0.113

Table 2.1: Risk table to compute a Kaplan-Meier estimate

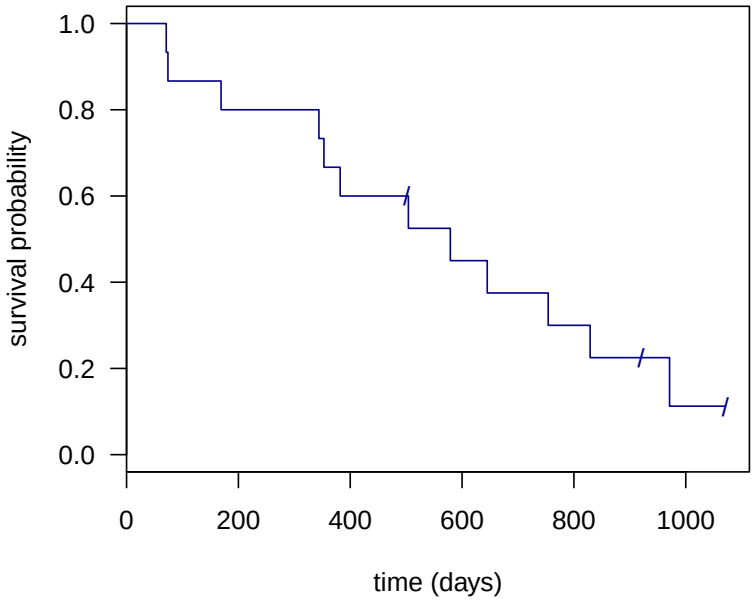

Figure 2.1: Kaplan-Meier NPMLE for right-censored data. The timepoints where censorings occurred are marked with /-signs on the curve.

Implementation in R [29] Kaplan-Meier estimation is available in the `R` package `survival` [37, 38]. The data is put in a `Surv` object that is called in the `survfit` function.

Estimation of survival quantiles and times is summarized in the here-defined `SurvStepUncens`, `SurvStepRightcens`, `TimeStepUncens` and `TimeStepRightcens` functions that are shown in the Appendix 8.2.2.

For interval-censored data: Turnbull

The nonparametric estimator used for interval-censored data is called Turnbull estimator [19, Chapter 3], [39]. It is uniquely defined up to certain areas of identical likelihood. Unfortunately, its ML estimate cannot be derived analytically but only iteratively by some optimization algorithms.

Theory Again, we have to sort the observed times in increasing order, now both the $T_{L\,i}$ and $T_{R\,i},\ i = \{1, 2, ..., n\}$ together and get $t_{b\,k}$ with $b \in \{R, L\}$ and $k = \{1, 2, ..., 2n\}$ (if there are no ties) such that

$$t_{b\,1} < t_{b\,2} < t_{b\,3} \quad ... < t_{b\,2n} \quad .$$

If we have exact events with $T_{L\,i} = T_{R\,i}$, we convert them to very small intervals to be able to apply the above sorting. For ties of the form $T_{L\,i} = T_{R\,j},\ i \neq j$ we increase $T_{L\,i}$ a little bit to avoid them. This is done formally by defining intervals that are left-open and right-closed, as described in the next paragraph. If whole intervals are tied, that means $T_{L\,i} = T_{L\,j}$ and $T_{R\,i} = T_{R\,j}$, one interval has to be put inside the other, that means $T_{L\,i} < T_{L\,j} < T_{R\,j} < T_{R\,i}$, by slightly shifting the times.

In a second step we have to identify all intervals $(t_{L\,k}, t_{R\,k+1}]$ of adjacent left and right interval borders, called Turnbull intervals, and redefine them in increasing order as

$$(t_{L\,j}, t_{R\,j}] = t_j, \quad j = \{1, 2, ..., s\} \quad .$$

It can be shown that any NPMLE of interval-censored data must concentrate all mass on these s Turnbull intervals [19]. That means the NPMLE is only decreasing inside these intervals and constant outside of them. At the same time the NPMLE is unidentifiable (up to upper and lower boundaries) inside the Turnbull intervals. This gives the Turnbull estimate survival function a special shape with horizontal stretches in non-Turnbull intervals and rectangular boxes indicating areas of equal likelihood in Turnbull intervals.

The levels of the horizontal stretches, i.e. the survival probabilities at certain times, are found by the following procedure. We again consider the t_j to be fixed and then only consider the mass inside the Turnbull intervals $f(t_j) = S(t_{L\,j}) - S(t_{R\,j})$ to form the likelihood function:

$$\mathbf{L}\left(f(t_1), ..., f(t_s)\right) = \prod_{i=1}^{n} \left(\sum_{j=1}^{s} \alpha_{ij} \cdot f(t_j) \right) \quad ,$$

where $\alpha_{ij} = 1\{(t_{L\,j}, t_{R\,j}] \subset (T_{L\,i}, T_{R\,i}]\}$ indicates whether the interval j is inside the observation i, that means the observation could have occurred in the Turnbull interval.

Maximizing the likelihood is done numerically in an interative way. Turnbull [39] uses the self-consistency algorithm which estimates $f(t_j)$ repetitively by applying:

$$\widehat{f(t_j)} = \sum_{i=1}^{n} \frac{\alpha_{ij} \cdot \widehat{f(t_j)}}{\sum_{l=1}^{s} \alpha_{il} \cdot \widehat{f(t_l)}} \quad .$$

This procedure is a special form of the expectation-maximization (EM) algorithm [19]. There exist other algorithms that are faster. The fastest one that is implemented in an `R` package was chosen for this thesis, see below.

Example We show the generation of the Turnbull estimate with the same example dataset as the one used in Figure 2.1 in the Kaplan-Meier function section that is coming from Figure 3.1 in the simulation setup section. The boundaries of the Turnbull intervals ($(t_{L,j}, t_{R,j}]$), that are the basis of the function, are shown in Table 2.2.

The corresponding Turnbull survival estimate is shown in Figure 2.2. The Turnbull intervals are marked with rectangles. The Turnbull estimate is not defined inside them and could take any decreasing kind of shape there. Consequently, at some timepoints and most quantiles the Turnbull survival function is only defined up to a certain range. This is a disadvantage for estimations of points on the curve that we would like to do later. Estimations are much easier if we choose one line type that is crossing through the rectangles.

In later estimations we choose three ways of crossing the Turnbull intervals. These are the lower boudary (dashed line in Figure 2.2), the upper boundary (solid line in Figure 2.2) and the linear approximation through the rectangles, which crosses all rectangles diagonally from the upper left to the lower right corner (dotted line in Figure 2.2). In this thesis the linear approximation (dotted line) is chosen for the simulation study.

The last rectangle in Figure 2.2 extends to infinite time and therefore does not contain any linear approximation. Consequently, the linear approximation does end above zero survival percentage.

The interpretation of the last rectangle is the same as for the rightcensoring. For all timepoints after the last non-infinite time the function is not defined. Therefore no survival probabilities beyond the value of the last time are possible.

j	$t_{L,j}$	$t_{R,j}$	$S(t_j)$
1	1	71	0.867
2	76	169	0.800
3	288	344	0.600
4	501	504	0.525
5	579	579	0.375
6	667	754	0.300
7	829	829	0.225
8	920	971	0.113
9	1071	Inf	0.000

Table 2.2: Table with Turnbull intervals, used to generate the Turnbull estimate

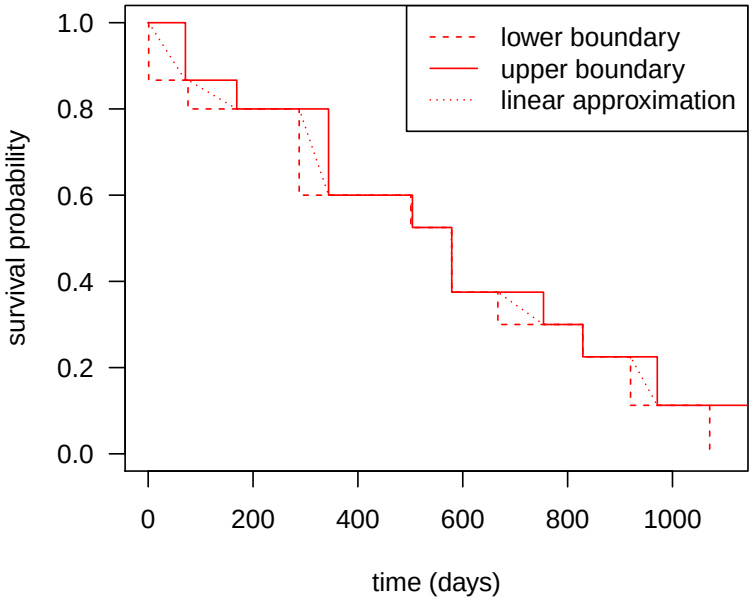

Figure 2.2: Turnbull NPMLE for interval-censored data. Both the upper and the lower boundaries of the Turnbull intervals are shown in the plot, giving rise to rectangular boxes inside which the function could take any shape. The linear approximation through the boxes that is taken for simulation studies in this thesis is indicated with the dotted line.

Implementation in R [29] Turnbull estimation is available in four different `R` packages that use different maximization algorithms. These are:

- EM algorithm in packages:
  - `survival` [37, 38] (functions: `Surv`, `survfit`)
  - `interval` [9] (function: `icfit`)
- EM-ICM (iterative-convex-minorant) algorithm in package:
  - `Icens` [12, 42] (function: `EMICM`)
- ISDM algorithm in package:
  - `Icens` [12, 23] (function: `ISDM`)
- projected gradient algorithm in package:
  - `Icens` [12, 43] (function: `PGM`)
- vertex exchange algorithm in package:
  - `Icens` [12, 2] (function: `VEM`)
- height map algorithm for reduction step and combination of sequential quadratic programming and support reduction algorithm for optimization step in package:

– `MLEcens` [25] (function: `computeMLE`)

Performance of these functions was compared by application to datasets generated in this thesis. ISDM and projected gradient algorithms did not converge and therefore were not considered further. Vertex exchange algorithm converged when bigger tolerance than default was accepted and was considered further. All remaining procedures led to the same result, but the speed of convergence was quite different. The speed was measured by R profiling and the resulting times shown in Table 2.3. The VEM and EM-ICM algorithms were the slowest, the EM algorithm variant in the `survival` package was quite a bit faster, but the most complex algorithm used in the `MLEcens` package was by far the fastest.

Estimation of survival quantiles and times is summarized in the here-defined `SurvStepIntervalcens` and `TimeStepIntervalcens` functions that are shown in the Appendix 8.2.2.

survival	Icens EMICM	Icens VEM	MLEcens	interval
1.94	3.60	5.56	0.16	3.58

Table 2.3: Time needed for calculation of Turnbull NPMLE, in seconds. Determined from one run with the `Rprof` function.

The `MLEcens` package with the `computeMLE` function is used in simulations of this thesis.

2.3 Test for identity

It is common in oncology clinical studies to compare the survival functions of the treatment groups by a statistical test. Simple tests for equality of survival functions are nonparametric rank tests [19].

2.3.1 For right-censored data: Logrank test

Theory

A nonparametric rank test for rightcensored data is the logrank test [19]. It tests for the null hypothesis of equal hazard. We use the two-sample version where the hazard of two groups is compared ($H_0 : h_1 = h_2$, formula of test statistic in [21, Chapter 7]). This is the same as testing if the hazard ratio between the two groups is equal to one (HR = 1).

The logrank test has highest power when treatment groups have proportional hazard [19]. It is equivalent to the score statistic of the semiparametric Cox regression model. The Cox regression model is a linear transformation model with Extreme Value probability distribution function [24], the same model type that is used for derivation of the generalized logrank test for intervalcensored data in the next section. The asymptotic properties of the logrank test can be derived by counting process theory [19].

Implementation in R [29]

The logrank test for rightcensored data is available in the `R` package `survival` [37, 38]. The data is put in a `Surv` object that is called in the `survdiff` function.

Estimation of p-values is summarized in the here-defined `LogRankTestUncens` and `LogRankTestRightcens` functions that are shown in Appendix 8.2.3.

2.3.2 For interval-censored data: Generalized logrank test

Theory

There are different ways to generalize the logrank test for rightcensoring data to the intervalcensoring case. In this thesis we focus on the ones mentioned in the `interval R` package that are nicely described in [3, Chapter 13] and also used in our estimations.

In Chapter 13 of [3] the derivation of two types of generalized logrank statistics is done for linear (or parametric) transformation models and their score statistics. Linear transformation models are defined by an unknown increasing function of event time, $g(.)$, and a known probability function, $F(.)$, such that

$$P(T \leq t|x) = F\left(g(t) - x^T \cdot \beta\right) \quad .$$

With this we can write the survival function at t for a patient with treatment indicator x_i as

$$S(t;\ x_i^T\beta, S_0) = 1 - F\left(F^{-1}\left(1 - S_0(t)\right) - x^T\beta\right) \quad , \tag{2.4}$$

with $S_0(t)$ being an unspecified survival function.

Now to get a score statistic we would like to maximize the likelihood as described for the general case in Section 2.2.1,

$$\mathbf{L}(\beta, S_0) = \prod_{i=1}^{n} \left[S(T_{L\,i};\ x_i^T\beta, S_0) - S(T_{R\,i};\ x_i^T\beta, S_0)\right] \quad ,$$

under the null hypothesis of $\beta = 0$, what implies $S(t;\ 0, S_0) = S_0$ and results in the likelihood function

$$\mathbf{L}(0, S_0) = \prod_{i=1}^{n} \left[S_0(T_{L\,i}) - S_0(T_{R\,i})\right] \quad .$$

The maximization can be done with the nonparametric maximum likelihood estimator for intervalcensored data, the Turnbull NPMLE, that is described in Section 2.2.2. Now we have a ML estimate for S_0, $\widehat{S_{0\,ML}}$.

Finally the score statistic can be written as

$$\mathbf{Z} = \left[\frac{\partial \log\left(\mathbf{L}(\beta, S_0)\right)}{\partial\beta}\right]_{\beta=0, S_0=\hat{S_0}} = \sum_{i=1}^{n} x_i \cdot \frac{\hat{S}'(T_{L\,i}) - \hat{S}'(T_{R\,i})}{\hat{S}_0(T_{L\,i}) - \hat{S}_0(T_{R\,i})} \quad , \tag{2.5}$$

where

$$\hat{S}'(T) = \left[\frac{\partial \log\left(S(T;\ \eta = x_i^T\beta, S_0)\right)}{\partial\eta}\right]_{\eta=0, S_0=\hat{S_0}} \quad . \tag{2.6}$$

This is the general form of the score statistic. Now we can define different variants of it by using different assumptions to get a simplified version of $\hat{S}'(t)$.

One assumption, that is made by Finkelstein [11] and used in our estimations, is the proportional hazard assumption. It is reached by setting the probability distribution F to an Extreme value distribution [3, Chapter 13]. With this assumption we can simplify (2.4) to

$$S(t;\ x_i^T\beta, S_0) = S_0(t)^{\exp\left(-x_i^T\beta\right)} \quad ,$$

and (2.6) to

$$\hat{S}'(t) = S_0(t) \cdot \log\left(S_0(t)\right) \quad .$$

This results in the generalized logrank test statistic proposed by Finkelstein [11],

$$\mathbf{Z} = \sum_{i=1}^{n} x_i \cdot \frac{\hat{S}_0(T_{L\,i}) \log\left(\hat{S}_0(T_{L\,i})\right) - \hat{S}_0(T_{R\,i}) \log\left(\hat{S}_0(T_{R\,i})\right)}{\hat{S}_0(T_{L\,i}) - \hat{S}_0(T_{R\,i})} \quad , \tag{2.7}$$

that is used for all generalized tests in this thesis.

The variant by Sun [33] that can be chosen in the `LogRankTestIntervalcens` function, has a slightly different definition of $\hat{S}'(t)$ resulting in the statistic

$$\mathbf{Z} = \sum_{i=1}^{n} x_i \cdot \frac{\hat{S}_0(T_{L\,i}) \log\left(\tilde{S}_0(T_{L\,i})\right) - \hat{S}_0(T_{R\,i}) \log\left(\tilde{S}_0(T_{R\,i})\right)}{\hat{S}_0(T_{L\,i}) - \hat{S}_0(T_{R\,i})} \quad , \tag{2.8}$$

with $\tilde{S}(t)$ being a function of a Nelson-Aalen type of estimate of $\hat{S}_0$.

The intervalcensoring methods for the logrank test cannot be related to the counting process theory and therefore it is difficult to derive their asymptotic properties [3, Page 13]. Because the asymptotic properties of these methods under the alternative hypothesis are unknown, they cannot be used for sample size calculations.

Implementation in R [29]

Different types of nonparametric tests for intervalcensored data are available in the `R` package `interval` [9]. They are all implemented in the `icfit` function with different arguments. Two types of generalizations of the logrank test, the one described by Finkelstein in [11] and Sun in [33], were examined further in two estimation process variants each, a permutation and a score variant. For our datasets it was observed that the four variants gave very similar results. In general the permutation variant was faster than the score estimation process.

To compute the logrank test statistic, the Turnbull NPMLE has to be computed. This is done with the EM algorithm of the `icfit` function that has been described in Section 2.2.2 as being quite slow. Therefore the estimation speed is much slower than for the survival function estimates of Section 2.2. One could implement a faster algorithm if one would mimick the structure of the `icfit` output. This was not done due to time limitations.

Estimation of p-values with all four versions is implemented in the here-defined `LogRankTestIntervalcens` function that is shown in Appendix 8.2.3. The Finkelstein method in permutation form is the default analysis method in that function and is also used for all the analysis in this thesis.

Chapter 3

Simulation setup

3.1 Overview

Our goal is to compare analysis methods for right- and interval-censored data in a specific study setting. For the best comparison we should relate the estimates obtained by analysis of censored datasets to the true values. The true values are only known for simulated datasets. Therefore we have to simulate datasets and cannot use observed data for our comparison. To stay close to the real world we try to simulate datasets that are very close to the GALLIUM clinical study of lymphoma. With the PFS primary endpoint the simulated datasets contain the following five main groups of patients:

1. patients that dropout from the study before an event due to refusal of consent or toxicity.
   → They are *right-censored* at dropout.

2. patients that die before progression is detected.
   There are two possible ways of treatment in the literature [36].
   - if we expect these patients having died from the disease: There must have been progression before death, we just did not detect it. They have an *interval-censored* event between the last assessment visit and the death date
   - if we expect these patients having died from another cause:
     → They have an *exact* event at the day of death.

   We choose the second option in our simulations.

3. patients that progress during the study.
   → They have an *interval-censored* event between two planned tumor assessment visits.

4. patients with no event (or dropout) until the last planned assessment visit.
   → They are *right-censored* at the last assessment visit.

5. patients with no event until study analysis. (Analysis might be earlier than the last planned visit. Typically analysis is done after a pre-specified number of events.)

→ They are *right-censored* at their last assessment visit.

These five patient groups are considered in a sequential manner in the data simulation process for this thesis.

Two types of censored data is generated, containing all the patient groups. The data types are displayed in more detail in Table 3.1.

a)	**Patient type**	Data type	**Intervalcensoring**	
No	Description		left time (T_L)	right time (T_R)
1.	dropout	rightcensored	time of dropout	infinity
2.	death event	exact event	time of death	time of death
3.	progression event	intervalcensored	assess. before	assess. after
4.	no event last assess.	rightcensored	last assess.	infinity
5.	no event analysis	rightcensored	assess. before	infinity
b)	**Patient type**	Data type	**Rightcensoring**	
No	Description		event/censoring time (T_{RC})	event indicator (δ)
1.	dropout	rightcensored	time of dropout	0 (censored)
2.	death event	exact event	time of death	1 (event)
3.	progression event	exact event	assess. after	1 (event)
4.	no event last assess.	rightcensored	last assess.	0 (censored)
5.	no event analysis	rightcensored	assess. before	0 (censored)

Table 3.1: Different patient groups and data types, applied to interval-censoring (a) and right-censoring (b) analysis.

3.2 Setup step by step

In this section, the data simulation process is explained step by step. The explanations are accompanied by a simulation of a small dataset and its visualization in Figures 3.1 and 3.2. The simulation parameters are all listed with the same names as the ones used in the simulation functions `WeibPFSSim1` and `WeibPFSSim2` that are shown in Appendix 8.2.1. The simulation functions exactly map the procedures that are explained below.

3.2.1 Sampling of event times

Event times (T_E) are sampled from the Weibull distribution,

$$T_E \sim Wb(\alpha = \texttt{shape}, \mu = \texttt{scale}) \quad , \quad S(t) = 1 - F(t) = \exp\left[-\left(\frac{1}{\mu} \cdot t\right)^{\alpha}\right] \quad ,$$
$$t \geq 0 \quad ; \quad \alpha, \mu > 0 \quad .$$

This step is visualized in Figure 3.1 a) and b). In Figure 3.1 a) the Weibull survival function that is sampled from is shown and the random sample drawn from it by "x"-es in the bottom. In Figure 3.1 b) the sampled event times are displayed in increasing order, such that the resulting plot looks similar to the 1 - ECDF of that sample.

The `shape` and `scale` parameters of the Weibull distribution as well as the number of patients `n` (reflected in the sample size) form simulation parameters that have to be defined before the simulation can be started.

3.2.2 Sampling of death events

Indicators of death events are sampled from a binomial distribution with π-parameter corresponding to the probability of a death event,

$$\texttt{death.ind} \sim Bin(1, \pi = \texttt{deathprob}) \quad .$$

If then $\texttt{death.ind} = 1$, this event was a death event, and

$$T_E = T_{RC} = T_L = T_R \quad ,$$

see also Section Overview 3.1, Point 2 for an alternative way to sample death events.

This step is visualized in Figure 3.1 c). The death events are the ones where both the right-censoring event time in blue and the interval-censoring time borders in red are displayed.

The `deathprob` parameter is the relevant simulation parameter for this sampling process.

3.2.3 Sampling of censoring times for dropout

One of the prerequisites for most analysis methods of censored data is that the censoring mechanism has to be independent from the mechanism of event generation (see also Section 1.1.3). In simulations one considers this by separately sampling from a censoring distribution, comparing the censoring and event times and choosing the one time that comes first.

We sample the censoring times (T_C) from the survival function of an exponential distribution,

$$T_C \sim Exp(\lambda = 1/\mu = 1/\texttt{censscale}) \quad , \quad S(t) = 1 - F(t) = \exp\left(-\frac{1}{\mu} \cdot t\right) \quad ,$$
$$t \geq 0 \quad ; \quad \mu > 0 \quad .$$

Then we choose the minimum of T_E and T_C as the rightcensoring time (T_{RC}) and either 0 or 1 as the censoring indicator (δ):

$$T_{RC} = \min\{T_E, T_C\}$$
$$\delta = \begin{cases} 1 & \text{if } T_E \leq T_C, \\ 0 & \text{otherwise} \end{cases} \quad .$$

For all the patients that are right-censored ($\delta = 0$), we apply the following interval-censoring times (T_L and T_R):

$$T_L = T_{RC}, \qquad T_R = \infty \quad .$$

This step is visualized in Figure 3.1 d) and e). In Figure 3.1 d) the Weibull survival function used for sampling the event times (in black) and the Exponential survival function used for sampling the censoring times (in red) are shown. The Exponential function decreases more slowly, so the probability to sample a smaller censoring time is quite low. This was also the case in our small sample shown in Figure 3.1 e), where only two event times are censored. The censoring times are displayed with green dots and red bars showing the left boundary of the interval-censoring interval.

The `censscale` parameter is the relevant simulation parameter that has to be defined for this process.

3.2.4 Sampling of assessment times for progression events

To simulate progression events, the tumor assessment schedule (`assess.vec`) has to be defined first.

In a first example with no deviations, the assessment times exactly match the `assess.vec`. The assessment time before the event ($T_{\text{A.before}}$) is taken as the left boundary of the interval-censoring interval and the assessment time after the event ($T_{\text{A.after}}$) is taken as the right boundary of the interval-censoring interval and as the right-censoring event time:

$$T_{\text{A.before}} = T_L, \qquad T_{\text{A.after}} = T_{RC} = T_R \quad ,$$

as explained in Section 1.2.1. All the event times that had not been assigned to death or censoring are mapped to that schedule. The result of this procedure is visualized in Figure 3.1 f), where assessment times are shown in orange lines, Interval-censoring intervals are shown in red and right-censoring events are displayed in blue.

An assessment schedule with no deviations for all patients is quite unrealistic in practice. To make it more realistic it was allowed in the simulation to introduce some deviations in the assessment schedule.

This was done by sampling the intrassessment interval widths (a vector) from some normal distribution with its mean corresponding to the target interval width and its standard deviation (sd) introduced as a simulation parameter

$$\begin{aligned}
&\text{width}_v = \texttt{assess.vec}_v - \texttt{assess.vec}_{v-1} \quad , \\
&v = \{2, ..., m\},\ m = \text{number of assessments} = \text{length of } \texttt{assess.vec} \quad , \\
&\texttt{width.jit} \sim N(\mu = \text{width}, \sigma = \texttt{sd}) \quad .
\end{aligned}$$

The resulting vector of widths is different for every patient and converted back to a patient-specific assessment schedule (assess.vec.jit)

$$\texttt{assess.vec.jit}_1 = \texttt{width.jit}_1, \quad \texttt{assess.vec.jit}_v = \sum_{w=1}^{v} \texttt{width.jit}_w \quad .$$

The matching of the assessment times to the event time is done in the same way as for the first example with no deviations and displayed in Figure 3.1 g).
An additional procedure has to be followed for the event or censoring times that are

larger than the last assessment time ($T_{\text{A.last}}$). They have to be right-censored at the last assessment, because their event or censoring can only be detected until that timepoint. The corresponding mathematical formulae are:

$$\begin{aligned} \text{If} \quad & T_{RC} > T_{\text{A.last}} \ : \\ & T_{\text{A.last}} = T_{RC} = T_L, \quad T_R = \infty \quad , \end{aligned}$$

and this is done for patient 1 in Figure 3.1 h).
The `assess.vec` vector and the `sd` parameter are the relevant simulation parameters for this sampling process.

3.2.5 Sampling of recruitment rate to calculate analysis timepoint

In clinical trials with time-to-event endpoint the analysis timepoint is chosen after a certain number of events, because this number, together with the estimated hazard ratio, determines the study's power. The number of events are defined for the whole study, therefore the data of all the study arms has to be put together for this last simulation step. The previous steps have been done with the `WeibPFSSim1` function, individually for every study arm, and this step is done with the `WeibPFSSim2` function, see also Appendix 8.2.1.
The number of events has to be assessed on the level of calendar time to find a calendar timepoint where the necessary number of events have been observed. To find the calendar event time from the simulated event time (which is the time since randomization), arrival times of patients to the study have to be sampled. We did this with the parameter `recr` that defines the patients recruited per month for that study arm. The sampling of arrival times (T_{arrival}) was done from an uniform distribution of the width one month. `recr` arrival times were sampled in the first month, the same amount in the second month, etc. until all patients had their arrival time,

$$\begin{aligned} \text{For} \quad & i \leq \texttt{recr} \ : \quad T_{\text{arrival}\,i} \sim U(0, 30) \quad , \\ \text{For} \quad & \texttt{recr} < i \leq 2 \cdot \texttt{recr} \ : \quad T_{\text{arrival}\,i} \sim U(30, 60) \quad , \\ & \text{etc.} \\ & i = \{1, ..., n\}, \quad n = \text{number of patients in this study arm} \quad . \end{aligned}$$

Then the calendar rightcensoring time (T_{calendar}) was calculated as the sum of the arrival time and the rightcensoring time,

$$T_{\text{calendar}} = T_{\text{arrival}} + T_{RC} \quad .$$

We then put the calendar times of the events (with event indicators $\delta = 1$) in increasing order,

$$T_{\text{calendar}\,(\delta=1)\,\mathbf{1}} \leq T_{\text{calendar}\,(\delta=1)\,\mathbf{2}} \leq T_{\text{calendar}\,(\delta=1)\,\mathbf{3}} \leq \cdots \quad ,$$

and pick out the event time of the `cutoff`-th patient. The cutoff time (T_{cutoff}) is then found just after this event time,

$$T_{\text{cutoff}} = T_{\text{calendar}\,(\delta=1)\,\mathbf{cutoff}} + \Delta \quad ,$$

where Δ is a very small positive number.

All patients with calendar times that are larger than the cutoff time are rightcensored at the last assessment before the cutoff time

$$\begin{aligned} \text{If} \quad & T_{\text{calendar}} > T_{\text{cutoff}} \ : \\ & T_{\text{A.before.cutoff}} = T_{RC} = T_L, \quad T_R = \infty \quad . \end{aligned}$$

This process is visualized in Figure 3.2 a) and b) and the final dataset shown in c). a) shows the calendar time per patient and in b) the cutoff time is applied as a dotted pink line and all events beyond that line are censored. An extract of the final dataset in c) is shown in Table 3.2 with all the time values and the description of the censoring processes applied to these patients. The `recr` parameter per study arm and the `cutoff` parameter for the whole study are the relevant simulation parameters for this sampling process.

no	Description	left.time	right.time	Data.type	rightcens.time	indic.
14	dropout	501	Inf	right	501	0
13	progression	753	835	interval	835	1
12	death	829	829	exact	829	1
11	progression	903	971	interval	971	1
10	death	1006	1006	exact	1006	1
9	dropout	920	Inf	right	920	0
8	analysis	1071	Inf	right	1071	0
7	assess. + analysis	1036	Inf	right	1036	0
6	assess. + analysis	1031	Inf	right	1031	0
5	assess. + analysis	1071	Inf	right	1071	0

Table 3.2: Extract of dataset used to construct plot c) in Figure 3.2.

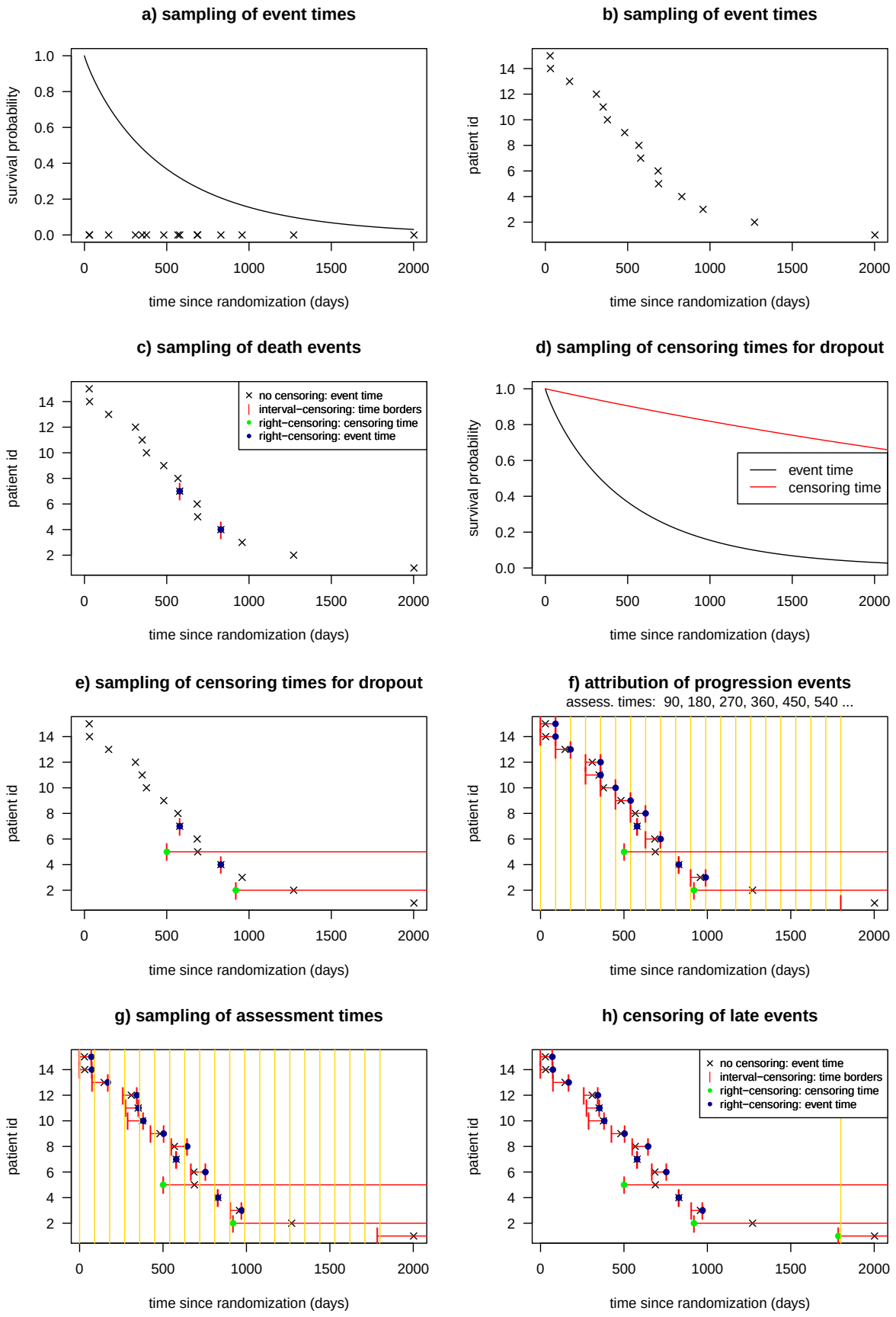

Figure 3.1: First part of simulation with one study arm: Visualization of the different steps with a small dataset. The data of the last plot h) can be reproduced by using this function: `WeibPFSSim1(1, 40, 0.9, 500, 0.25, 5000, (1:20)*90, 10, 10, 200)$sim.data[[1]][21:35, ]`. See Sections 3.2.1 to 3.2.4 for details of the process.

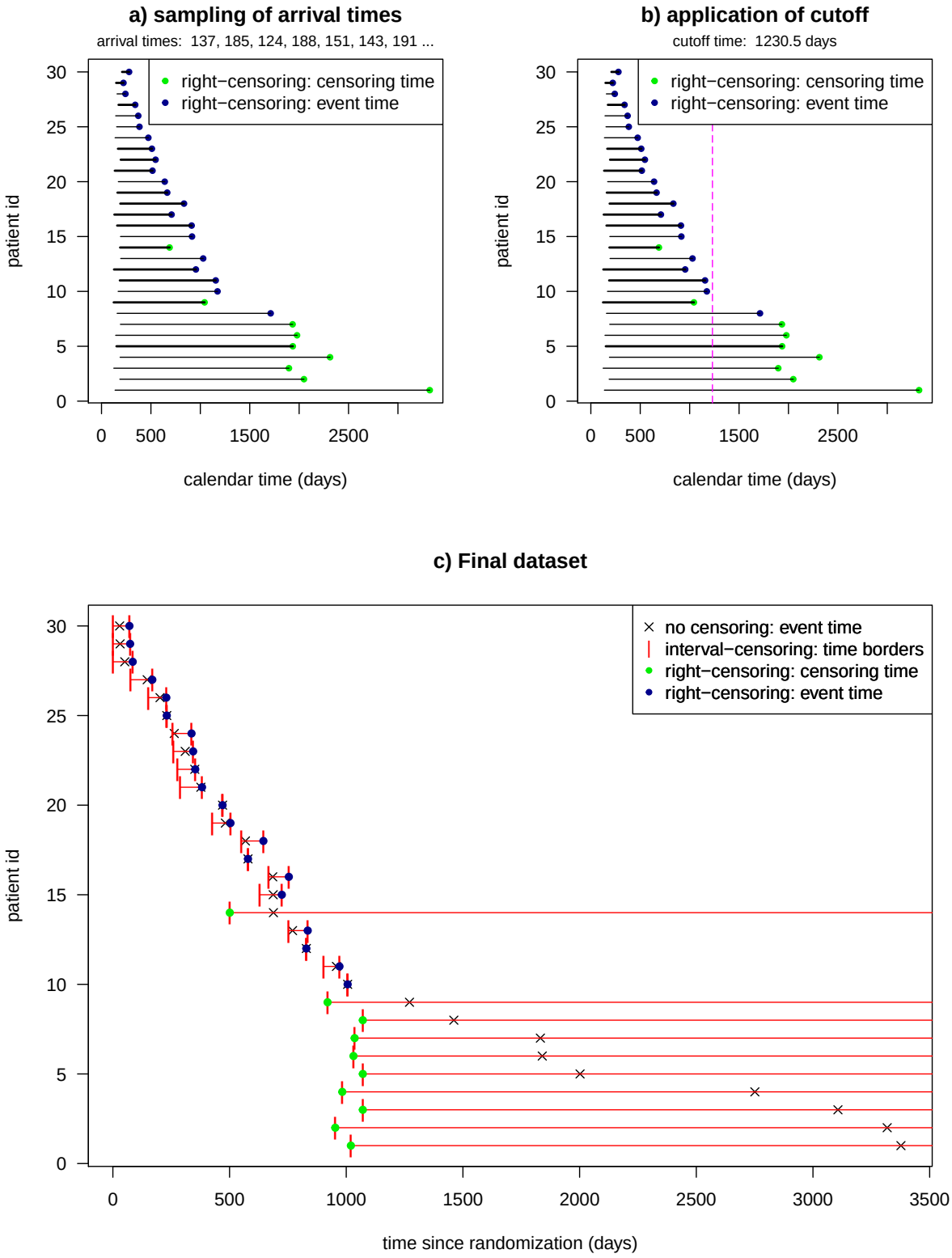

Figure 3.2: Second part of simulation with both study arms: For calculation of cutoff time the whole dataset and the arrival times have to be considered. The patients from the study arm whose data is shown in Figure 3.1 are indicated with thicker lines in a) and b). The final dataset in c) can be received with the following function:
`WeibPFSSim2(WeibPFSSim1(1, 40, 0.9, 500, 0.25, 5000, (1:20)*90, 10, 10, 200), WeibPFSSim1(1, 40, 0.9, 700, 0.25, 5000, (1:20)*90, 10, 10, 20000), 5, 5, 60, 2, 200)$sim.data[[1]][c(21:35, 61:75), ].`

3.3 Simulation scenarios

To perform a data simulation as it is described in the previous section, one needs to specify many parameters. The name and nature of these parameters is also described in the previous section, but not the parameter values that were used. Dependent on the parameter values many different kinds of datasets can be simulated.

In our simulation study we first define parameter values that closely match an existing study, the GALLIUM study of lymphoma. The set of these parameters is called the baseline scenario. This scenario should be quite realistic. Its composition and properties are described in the next section.

In the subsequent sections some parameters of the baseline scenario are varied to get scenarios that deviate in some aspects from the baseline scenario. With these other scenarios the robustness of the estimation methods with regards to deviations from the baseline scenario is tested. The final goal of this process and the whole thesis is to find differences between rightcensoring and intervalcensoring methods.

The parameter values that were used for the different scenarios are summarized in Table 3.3. In addition, for all the scenarios the same simulation seed was used. It was 200 for `WeibPFSSim1` for the control arm, 20000 for `WeibPFSSim1` for the treatment arm and 200 for `WeibPFSSim2`.

3.3.1 GALLIUM study: baseline Scenario 1

To define the parameter values of this scenario we use the values that were used for sample size calculations of the GALLIUM study. The only deviation from these parameters is to use a Weibull distribution as the basis of the simulations instead of the Exponential distribution that was used in sample size calculations. This because the Weibull distribution is more flexible in capturing different shapes for the survival function.

Parameters of Section 3.2.1: Sampling of event times

The `shape` ($= \alpha$) parameter of the Weibull distribution (in the form of Section 2.2.1) was chosen to be 0.9. With this parameter the function has a similar shape as the Kaplan-Meier estimate of an earlier study on Follicular Lympoma [26].

With this `shape` parameter and the knowledge of the observed median survival of 6 years for Rituximab treatment and the assumed median survival of the Gazyva treatment of 8.1 years the `scale` parameter could be calculated from the quantile function of the Weibull distribution,

$$F^{-1}(t) = Q(p) = \texttt{scale} \cdot (-\log(1-p))^{\frac{1}{\texttt{shape}}} \quad ,$$

by finding the `scale` parameter of the only suitable Weibull function that crosses the median at the derived timepoint. This is done in practice by setting the quantile function equal to the derived timepoint t and then solving the resulting equation for `scale` to get

$$\texttt{scale} = \frac{t}{(-\log(1-p))^{\frac{1}{\texttt{shape}}}} \quad .$$

By inserting $t = 6 \cdot 365.24 = 2191$ for the control arm and $t = 8.1 \cdot 365.24 = 2958$ for the treatment arm, and $p = 0.5$ and `shape` $= 0.6$ we get the `scale` parameter for the control arm to be 3293, and for the treatment arm to be 4446.

The number of patients per study arm (parameter `n`) is 600. This is calculated from the 1200 Follicular Lymphoma patients needed for the whole study and considering the 1:1 randomization to the study arms. Only the Follicular Lymphoma patients from GALLIUM are included because this population is used for the primary endpoint of the study.

Parameters of Section 3.2.2: Sampling of death events

The probability for a death event (parameter `deathprob`) is 0.25. This probability is known empirically from previous studies in Follicular Lymphoma and from the current GALLIUM data.

Parameters of Section 3.2.3: Sampling of censoring times for dropout

The annual dropout rate is assumed to be 2.5%. We have to recalculate this value to a daily dropout rate with a kind of compound interest calculation and then convert it to the scale parameter of the Exponential distribution of which the censoring times are sampled from. This is done with the formula

$$\texttt{censscale} = \frac{1}{1 - \exp\left(\log(1 - \texttt{dropout.year})/365.24\right)} \quad .$$

Inserting 0.025 for the annual rate results in `censscale` $= 14427$.

Parameters of Section 3.2.4: Sampling of assessment times for progression events

The assessment days are calculated according to the GALLIUM study protocol and visualized in Figure 1.1. They are 78, 197, 270, 391, 513, 696, 909, 1061, 1244, 1426, 1609, 1792, 1974, 2339 and 2705 days after randomization.

Parameters of Section 3.2.5: Sampling of recruitment to calculate analysis timepoint

According to the sample size calculation of GALLIUM, the recruitment number per month is smaller until the first futility interim analysis, and higher later on. In our simulations we just model the final analysis timepoint and therefore a simpler model could be chosen with just one recruitment number per month per arm (parameter `recr`), corresponding to the later recruitment number of GALLIUM.

The advantage to define the recruitment number per arm is that one can cover different randomization patterns by setting the `recr` parameter to different values across arms. However, in GALLIUM there is 1:1 randomization and the `recr` parameter was set to 18 for both arms for the baseline scenario.

The `cutoff` parameter is defined for the whole study and represents the number of events that are needed for a certain study power. In the case of GALLIUM this is 370 events for 80% power, and we take the same number of events for our simulations.

3.3.2 Change in assessment times: Scenarios 2 to 7

Scenario 2: exact adherence to assessment schedule

It is not sure whether the Normal variation that is applied to the assessment schedule in Scenario 1 is already introducing some error in the analysis. Therefore there was chosen a scenario that does not allow for any deviation from the assessment schedule by setting the `sd` parameter to zero. This scenario has the mathematically most pure case of interval definition and is therefore interesting to look at, but at the same time it maps a case that might never occur in clinical reality.

Scenario 3: bigger variance in assessment times

To assess the effect of even more Normal variation, a third scenario was chosen with `sd` = 20 days.

Scenario 4: less variance in assessment times in control arm

Because of the open-label nature of the GALLIUM study there might be introduced by the doctors some unconscious bias that is called evaluation-time bias in the literature [3, Chapter 10]. For example, the doctors might think that the new treatment is better because its superiority has already been proven for other, closely related indications. This is the case for GALLIUM. Then they might be more strict with patients in the control arm to follow the assessment schedule than with patients in the treatment arm. A scenario like this has been considered in Scenario 4, where for the control arm the `sd` parameter is set to 5 days and for the treatment arm to 20 days. This scenario does violate the uninformative censoring assumption described in Section 1.1.3.

Scenario 5: longer intraassessment intervals in both arms

How does the performance of the estimation methods change if the intraassessment intervals are longer? This question is tried to be answered in Scenario 5, where every intraassessment interval was prolongued by 14 days, resulting in the following assessment schedule: 92, 225, 312, 447, 583, 780, 1007, 1173, 1370, 1566, 1763, 1960, 2156, 2535 and 2915 days.

Scenario 6: longer intraassessment intervals in treatment arm

Another type of evaluation-time bias (see Scenario 4) is considered in Scenario 6: here it is thought that the treatment arm patients have the prolongued schedule that is described in Scenario 5, but not the control arm patients. This scenario might be realistic for the same reasons as Scenario 4. This scenario does violate the uninformative censoring assumption described in Section 1.1.3 as well.

Scenario 7: missingness of every second assessment time

Scenario 7 is quite an extreme scenario. We assume for it that every second assessment is not done for all patients, resulting in this schedule: 197, 391, 696, 1061, 1426, 1792 and

2339 days. With this scenario we would like to see the changes in estimation precision if much less assessments are performed.

3.3.3 Change in proportion of deaths: scenarios 8 and 9

What is the effect of the proportion of death events on estimation precision? This question was tried to answer with the following two scenarios.

Scenario 8: smaller probability of death event

In Scenario 8 the `deathprob` parameter was set to 0.1.

Scenario 9: bigger probability of death event

In Scenario 9 we use a higher probability of a death event, 0.5.

3.3.4 Increase in percentage of dropout: scenarios 10 and 11

Scenario 10: more dropouts in both arms

The dropout rate is generally difficult to estimate. Therefore another scenario with a higher annual dropout rate of 10% is created, Scenario 10. The dropout rate is converted to the `censscale` parameter by the formula explained for Scenario 1 in Section 3.3.1, resulting in a value of 3467.

Scenario 11: more dropouts in control arm

Open-label studies might be susceptible to another type of bias that is explained in [3, Chapter 10], the attrition bias. One form of this bias occurs when patients in the control arm more often dropout from the study, for example due to switching to the new treatment, because they (or their doctor) thinks the new treatment is more effective. This behaviour is modeled in Scenario 11, where the annual dropout rate of the control arm patients is increased to 10% and the dropout rate of the treatment arm patients stays at 2.5%. Therefore the `censscale` parameter of the treatment arm changes to 3467. This scenario does violate the uninformative censoring assumption described in Section 1.1.3.

3.3.5 Increase in events at cutoff: scenario 12

Finally we consider a scenario where more events are observed before the analysis cutoff. The new number of events is 520. It was chosen to be close to the maximum number of events that could be obtained with this setting, to get the biggest possible deviation form the baseline scenario.

Section	Description	parameter		value
a)	**Scenario 1**			
1	number of patients in study arm	n	=	600
	parameter of Weibull distribution	shape	=	0.9
	parameter of Weibull distribution (control arm)	scale	=	3293
	parameter of Weibull distribution (treatment arm)	scale	=	4446
2	percentage of death events	deathprob	=	0.25
3	parameter of Exponential distr. for censoring	censscale	=	14427
4	vector of assessment times	assess.vec	=	
	(78, 197, 270, 391, 513, 696, 909, 1061, 1244, 1426, 1609, 1792, 1974, 2339, 2705)			
	parameter of Normal distr. for intraassess. length var.	sd	=	10
5	number of patients recruited per month per study arm	recr	=	18
	number of events needed at analysis (whole study)	cutoff	=	370
-	number of datasets simulated per scenario	M	=	1000
b)	**Scenario 2**			
4	parameter of Normal distr. for intraassess. length var.	sd	=	0
c)	**Scenario 3**			
4	parameter of Normal distr. for intraassess. length var.	sd	=	20
d)	**Scenario 4**			
4	parameter of Normal distr. (control arm)	sd	=	5
	parameter of Normal distr. (treatment arm)	sd	=	20
e)	**Scenario 5**			
4	vector of assessment times	assess.vec	=	
	(92, 225, 312, 447, 583, 780, 1007, 1173, 1370, 1566, 1763, 1960, 2156, 2535, 2915)			
	(all intervals: plus 14 days)			
f)	**Scenario 6**			
4	vector of assessment times (control arm)	assess.vec	=	(like Scen. 1)
	vector of assessment times (treatment arm)	assess.vec	=	(like Scen. 5)
g)	**Scenario 7**			
4	vector of assessment times	assess.vec	=	
	(197, 391, 696, 1061, 1426, 1792, 2339) (only every second assessment)			
h)	**Scenario 8**			
2	percentage of death events	deathprob	=	0.1
i)	**Scenario 9**			
2	percentage of death events	deathprob	=	0.5
j)	**Scenario 10**			
3	parameter of Exponential distr. for censoring	censscale	=	3467
k)	**Scenario 11**			
3	parameter of Exponential distr. (control arm)	censscale	=	3467
3	parameter of Exponential distr. (treatment arm)	censscale	=	14427
l)	**Scenario 12**			
5	number of events needed at analysis (whole study)	cutoff	=	520

Table 3.3: Parameters used to simulate Scenario 1 (a) and deviations from these parameters to simulate the other scenarios (b to l).

Chapter 4

Simulation results

All the different scenarios were simulated 1000 times with the procedure described in Chapter 3. The resulting 1000 datasets per scenario were analyzed with the estimation methods described in Chapter 2. The resulting estimates are analyzed and summarized in this chapter.

4.1 Estimation of survival function at certain quantiles and times

The survival function was estimated by study arm, by a nonparametric MLE function (called "step" in later figures of this section) and a parametric Weibull function (called "weib" in later figures of this section), as described in Section 2.2. Examples of these estimated functions are shown in Figure 4.1.

When we look at the light blue lines in Figure 4.1, that indicate the points of the survival function that were analyzed further in fingerprint plots, we can see that they follow a certain sequence. In arm1 (control arm) this sequence is 500 days, 0.8 quantile, 1000 days, 0.65 quantile, 1500 days, 0.5 quantile along the function. In arm2 (treatment arm) the sequence is nearly the same, only 1500 days and 0.65 quantile are switched.

We are aware that the estimation of quantiles and timepoints is statistically a different procedure and therefore, strictly speaking, the resulting RMSE and bias values cannot be compared directly. But, as we will see in the following result sections, their relative size drifts along the survival function are comparable. Because of that, and for the sake of simplicity, we will sometimes summarize the results from quantiles and timepoints in the same sentence in the next sections.

Instead of dividing the estimates by estimation direction, we will map the estimates to different stages of the function and study. We can map 500 days, 0.8 survival quantile and 1000 days to the *early stage*, 0.65 survival quantile and 1500 days to the *late stage* and the 0.5 survival quantile (the median) to the *too late stage*, because it is not covered in most datasets, as we will see in Section 4.1.1.

The median is chosen as a quantile of interest because of its general importance in oncology, where normally the median survival (or other endpoint) is reported as a primary summary of the survival function estimates. Nevertheless, the GALLIUM study that is mimicked with our data, is not designed to reach the median PFS, because the required

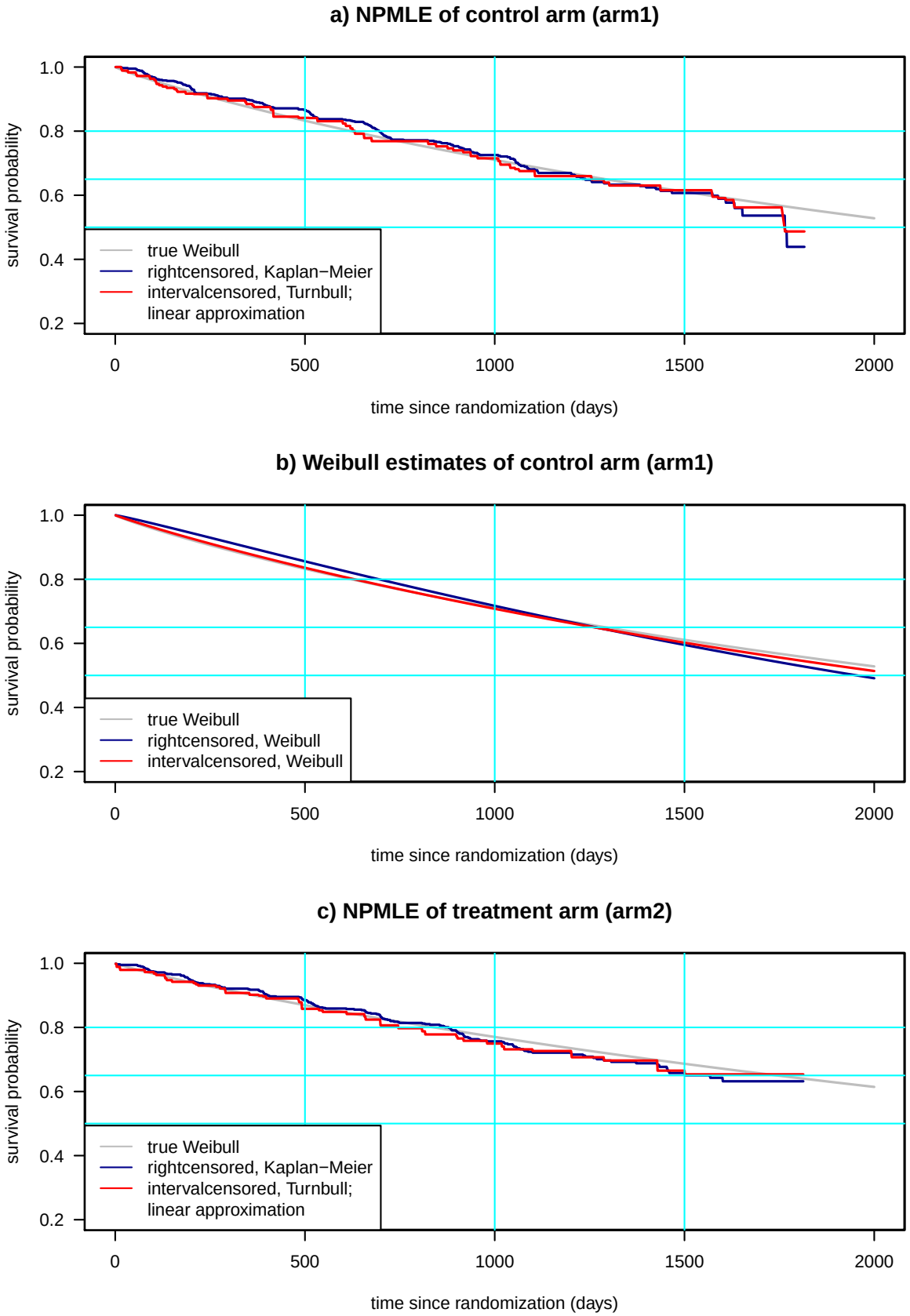

Figure 4.1: Example of estimated survival functions of from a dataset that has been simulated with Scenario 1: a) nonparametric MLE (NPMLE) functions of the control arm (arm1), b) Weibull functions of the control arm (arm1), c) nonparametric MLE (NPMLE) functions of the treatment arm (arm2). The marks for censorings are not included in blue lines of plots (a + c) because they would cover the whole curve. The light blue lines indicate the survival quantiles and timepoints that were analyzed further in fingerprint plots.

number of events at analysis cutoff is most likely to be reached before the median.

The same terminology of *early* and *late stage* of the function can be used for both study arms. That will be useful later on.

In a second plot type we will use the estimates of five timepoints, 500 days, 750 days, 1000 days, 1250 days and 1500 days. They can also be graded in an early and a late stage group. The first three timepoints (500 to 1000 days) are put in the early stage group and the later two (1250 days and 1500 days) in the late stage group.

4.1.1 RMSE and bias

The root mean squared error (RMSE) is a measure of the accuracy of an estimation method. It shows the root of the mean squared error (MSE), which is the average squared deviation of the estimate from the true value [17, Chapter 3]. The RMSE is used here to compare right- and interval-censoring estimation methods of survival functions for different settings.

The MSE can be split into two additive components, the squared bias and the variance. It is also interesting to look at one of these components to see where the MSE comes from. In this thesis we look primarily at the bias. We expect it to be bigger in the rightcensoring estimates because of the attribution of progression events to the end of the intraassessment interval, as described in Section 1.2.1.

In one plot type we will compare the size of the bias with the size of the RMSE. From the gap between the bias and corresponding RMSE value one can guess the share of the variance that is included in the estimation process, because

$$\text{RMSE} = \sqrt{\text{bias}^2 + \text{Var}} \quad \text{and therefore} \quad \text{Var} = \text{RMSE}^2 - \text{bias}^2 \quad .$$

RMSE and bias values are calculated for different data subsets (control arm (arm1) and treatment arm (arm2) of the studies), different estimation methods (NPMLE and parametric Weibull) and different survival times (500, 1000 and 1500 days) and quantiles (0.8, 0.65, 0.5) of the survival function. So there are a total of 48 RMSE values per scenario and the same number of bias values. In order to be able to compare them between scenarios, they will plotted in two plots each with colorcoding that forms a kind of fingerprint of the scenarios.

4.1.2 Scenario 1: like GALLIUM study

Results from the baseline scenario are looked at more thoroughly than from the other scenarios, because it is a realistic scenario that fulfills all the statistical assumptions of noninformative censoring. A close look is also important because deviations from results of the baseline scenario will be discussed later.

RMSE

Figure 4.2 (a + b) shows the fingerprint plots of the RMSE values of all data subsets, estimation methods, censoring methods and survival times and quantiles for the baseline Scenario 1. We look now at differences between all these groups.

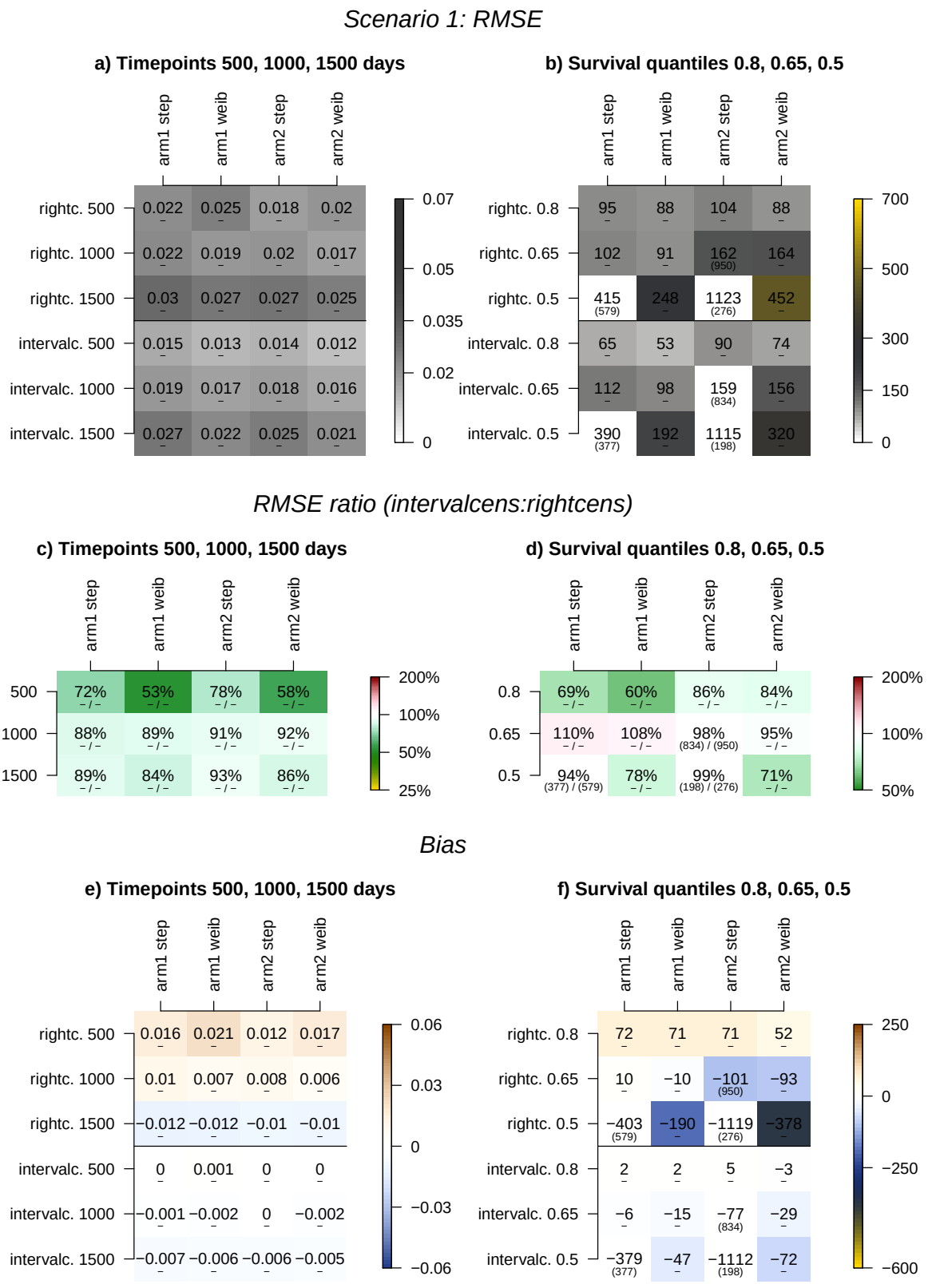

Figure 4.2: RMSE (a + b), RMSE ratio (c + d) and bias (e + f) estimates of 1000 datasets of Scenario 1 in colors. The estimates at timepoints 500, 1000 and 1500 days are shown in the left plots (a, c + e) and the survival quantiles 0.8, 0.65 and 0.5 in the right plots (b, d + f). The values are colorcoded as indicated in the legends. We differentiate both study arms and estimation variants on the x-axis, both censoring types and three timepoints or quantiles on the y-axis of the plots. Sometimes the NPMLE (step) functions were not defined at the readout quantile in all datasets. Then the number of datasets with a valid estimate is displayed in brackets below the RMSE value. Estimates from less than 900 datasets do not get the color coding because they may be subject to bias.

Times and quantiles RMSE values at timepoints and survival quantiles are shown in separate plots. Figure 4.2 a) shows the RMSE value in terms of survival probability and Figure 4.2 b) in terms of survival time in days. The RMSE values in these two plots have a quite different range. This is caused by the difference in axis scale which goes from 0 to 2000 on the x-axis (time) and from 0 to 1 on the y-axis (survival probability).

The colorcoding connects the two ranges and allows comparison of both types of estimates and differentiation of early and late stage estimates independent of their type.

When looking at Figure 4.2 b) we see as well that the NPMLE (step) function RMSE values at the median are not coloured for both arms and censoring types. This is because the estimation is done with less than 900 datasets (from a total of 1000) and therefore not completely reliable. For the other datasets the function did not reach that quantile and is therefore undefined at that point, as described in Methods Section 2.2.2 in the Example paragraphs. Estimation for parametric Weibull functions is possible at any quantile, because parametric survival functions are defined for all $t \geq 0$.

Censoring First we would like to compare the RMSE values of the right-censoring and interval-censoring estimation methods, the main goal of our study. To better visualize this comparison, another plot type was produced. It is displayed in Figure 4.2 (c + d) and shows the percentage of the size of the intervalcens RMSE in terms of the rightcens RMSE.

But first we look at the fingerprint plot in Figure 4.2 (a + b) again and we see that the RMSE values of the intervalcensored estimates are continuously increasing during the course of the study. That means the accuracy of the estimates is continuously decreasing. A potential explanation for this gain in RMSE is the decrease in the number of patients during the course of the study. In the nonparametric case the number of patients considered is called the "risk set", explained in Section 2.2.2). The RMSE values of the rightcensored estimates are stable in early stage, and they increase in late stage of the function.

The ratio of intervalcensored to rightcensored RMSE, as it is visualized in Figure 4.2 (c + d), gives the following picture. The percentage of intervalcensored is between 50% and 90% for the early stage, between 85% and 110% for the late stage. We can conclude from this data that for estimation at the early stage of a survival function the intervalcensoring analysis method is better, at a late stage it is about the same as the rightcensoring. The bigger difference in the early stage might be due to a higher bias of the rightcensoring estimates, what is explored in the next section.

But before we go to the bias section, we would like to do some other comparisons.

Study arms and estimation type When we compare the study arms in the censoring ratio Figure 4.2 (c + d), we see that they are comparable. That means the accuracy gain when doing the intervalcensoring analysis is about the same for both study arms. The same is approximately true for the two estimation methods, NPMLE and Weibull. In the very early stage of the function (500 days) and in the too late stage (0.5 quantile) the accuracy increase seems to be even bigger in Weibull estimation. This is due to higher rightcensoring RMSE values at these points. So Weibull estimation with rightcensoring methods is less accurate in the extremes of the functions.

For the comparison of the study arms we can do another ratio figure, similar to the one in censoring. In Figure 4.3 we display the percentages of the arm 2 values in terms of the arm1 values. The result is quite surprising: In plot a) all the percentages are below 100%, that means the RMSE values are smaller for arm2, but in plot b) it is opposite. There, apart from one exception, all percentages are above 100%, what means the RMSE values are bigger for arm2.

This phenomenon cannot be explained easily. The observations in Figure 4.3 b) do better match with intuition. In general, the measurement accuracy has to do with the proportion of events (exact and intervalcensored events) from all observations. Here this is lower in arm2, because in the treatment arm the median time to event is longer. An estimation with a lower number of events is generally less accurate. Therefore the RMSE values for arm2 should be bigger than for arm1. We have no explanation for the observations in Figure 4.3 a) that go in the opposite direction.

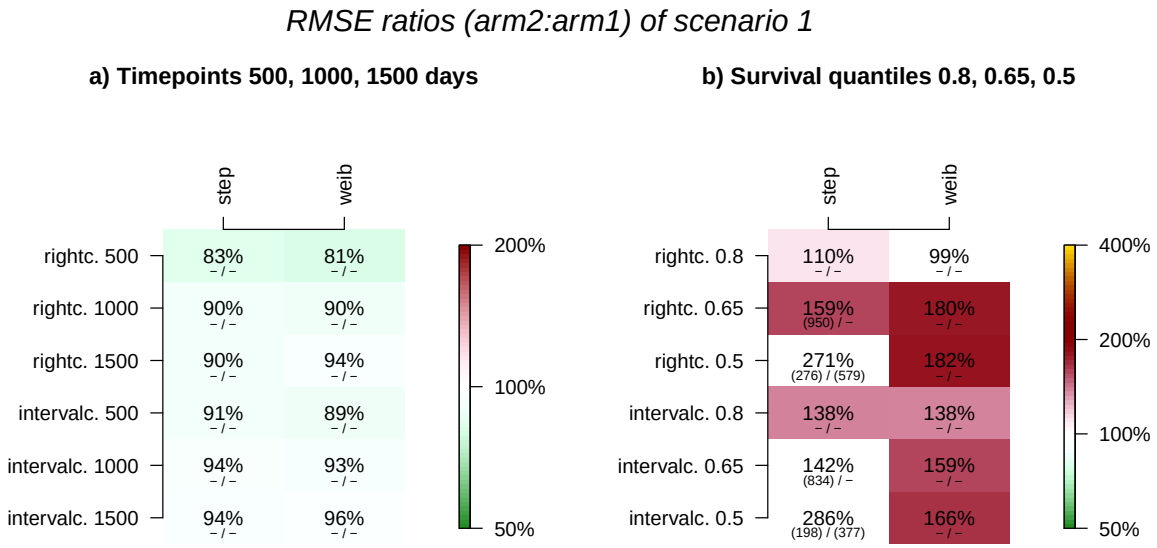

Figure 4.3: RMSE ratios of study arm2 : arm1 of Scenario 1 in colors: the more reddish the color the bigger the increase in RMSE, the more greenish the color the bigger the decrease in RMSE. We look at the timepoints in plot a) and at the survival quantiles in plot b) and differentiate the estimation variants on the x-axis, both censoring types and three timepoints or quantiles on the y-axis of the plots. Sometimes the NPMLE functions were not defined at the readout quantile in all datasets. Then the number of datasets with a valid estimate (from a total of 1000) are displayed in brackets below the RMSE value. Estimates from less than 900 datasets in at least one estimation do not get the color coding because they are not completely reliable.

Bias

By looking at the bias one can find out more about the general direction of the error and about its share of the RMSE. The fingerprint plot of the bias of Scenario 1 is shown in Figure 4.2 (e + f).

We can see from the more coloured upper parts of Figure 4.2 (e + f) that the bias is bigger in general for the rightcensoring estimates. In early stage the rightcensoring bias is positive, in late stage it is mostly negative. The bias is much smaller in the

intervalcensoring estimates, nearly zero in early stage and slightly negative in late stage. The bigger positive bias of rightcensoring estimates at early stage can be explained by the incorrect handling of progression events, where the late end of an intraassessment interval is taken as an exact event. In intervalcensoring estimation the progression events are treated correctly as intervalcensored and no bias is introduced in the estimates. The bigger negative bias of rightcensoring estimates at late stage is more difficult to explain.

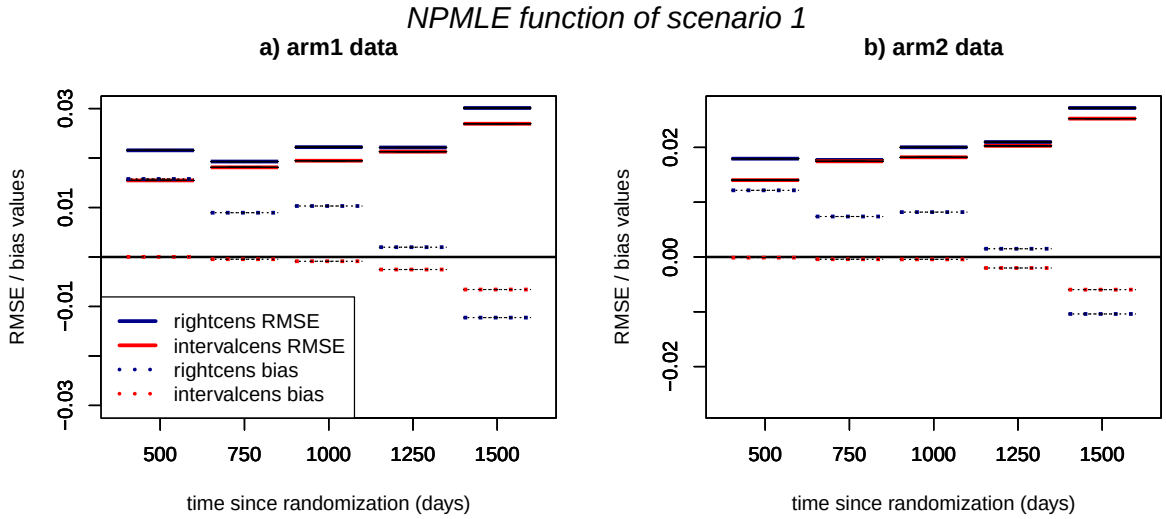

Figure 4.4: RMSE and bias values of NPMLE function estimates of Scenario 1 in order of appearance on the function: Solid lines represent the RMSE, dotted lines the bias values. We look at arm1 in plot a) and at arm2 in plot b). We compare the estimates at five timepoints: 500 days, 750 days, 1000 days, 1250 days and 1500 days.

To better compare censoring differences at different timepoints another figure type is introduced. Figure 4.4 shows the NPMLE function RMSE and bias values at five timepoints in solid and dotted lines of the two study arms. There the bias and RMSE values can be followed over time and the bias can be compared directly to the size of the RMSE. From the gap between the bias and corresponding RMSE value one can guess the share of the variance, as explained in Section 4.1.1.

One can see in early stage (timepoints 500 days to 1000 days) that the much bigger bias in rightcensoring estimation does only convert partially to a higher RMSE, because in intervalcensoring we have a much bigger variance in the estimation. This might come from the bigger uncertainty that is implied in intervalcensoring estimation of progression events.

But the RMSE (including both bias and variance) is absolutely still smaller in intervalcensoring estimation, so it is recommended to use that estimation anyway, because it does not only have a much smaller bias but also the squared error of its estimation is smaller. This result is not surprising. The intervalcensoring method models the data correctly and should therefore produce smaller RMSE values.

In late stage (timepoints 1250 days and 1500 days) the situation is more balanced. The RMSE values are closer together and the bias is mostly negative for both censoring methods. But also there the bias of the rightcensoring estimates is generally bigger and therefore it is recommended to use intervalcensoring estimation functions also for point estimates at late timepoints.

4.1.3 Scenarios 2 to 7: change in assessment times

We can divide these scenarios in two groups. In the first group containing Scenarios 2 to 4 just the variance of the intraassessment period length was varied according to a normal distribution, in the second group containing Scenarios 5 to 7 the intraassessment period length was changed. Scenario 6 is not discussed here, because its arm1 is like the one of Scenario 1 and its arm2 like the one of Scenario 5.

RMSE

Scenarios 2 - 4: change variation of intraassessment period lengths The RMSE fingerprint plots of Scenarios 2 to 4 are only shown in the Appendix Section 8.3 in Figures 8.1, 8.2 and 8.3, because they are very similar to the plots of Scenario 1 in Figure 4.2. We can conclude that the amount of normal variation in the intraassessment length does not influence estimation of the survival function. This result was expected.

One remark could be done about Scenario 2, where we have observations with exact assessment schedules, that means many events at exactly the same date. In this scenario estimation procedures run remarkably smooth, even if it is an extreme setting with many ties. Only in intervalcensoring estimation 5 of the 1000 datasets could not be estimated because the algorithm did not converge. This problem might be due to the special data structure, because in no other scenario similar problems occurred with that algorithm.

Scenario 5: add two weeks to all intraassessment periods The RMSE fingerprint plots of Scenario 5 are more interesting and shown in Figure 4.5 (a + b). We see some slight shifts in the RMSE of rightcensoring estimates and NPMLE functions.

At the first timepoint (500 days) the RMSE is decreased compared to Scenario 1, at the two following points (0.8 quantile and 1000 days) it is increased. In early stage, apart from the earliest timepoint, rightcensoring estimation seems to become less accurate when the length of the assessment intervals is increased. This observation is expected because we expect the rightcensoring bias to increase with an increase in assessment interval length.

Anyway, in late stage (at timepoint 1500 days) there is observed a slight decrease of rightcensoring RMSE in NPMLE functions again. In general, the RMSE shifts are only small. They show a complex pattern for timepoints, that does not go always in the expected direction. For the quantiles the RMSE increases compared to Scenario 1, as expected. the intervalcens RMSE values in contrast, are quite stable. We will see in the next section whether the pattern of the bias is the same.

The shifts mentioned above are only observed in NPMLE function estimation (marked with "step" in the figure). Weibull function estimation is stable and seems to be more robust towards slight changes in assessment interval length.

Figure 4.5 (c + d) show whether the rightcensoring RMSE increase is reflected in a percentage decrease of the intervalcens : rightcens ratio. When looking at the percentages we can conclude that they changed in the same way as explained for the rightcens RMSE, and also only for NPMLE functions. This because there is observed nearly no change in the intervalcens RMSE values.

Scenario 7: leave out every second assessment This quite extreme scenario gives some interesting results that are shown in the fingerprint Figure 4.6. The changes in colors

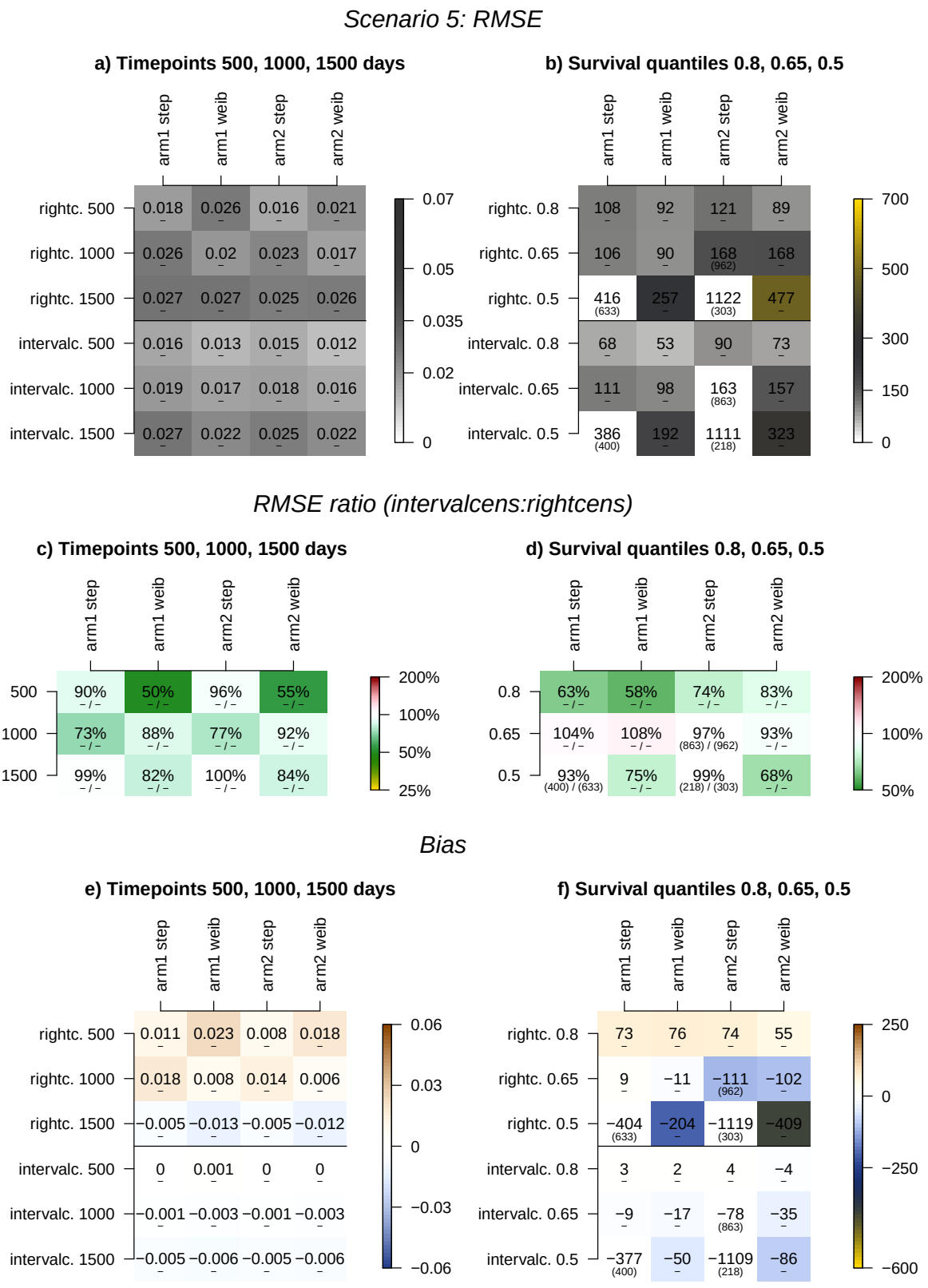

Figure 4.5: RMSE (a + b), RMSE ratio (c + d) and bias (e + f) estimates of 1000 datasets of Scenario 5 in colors. The estimates at timepoints 500, 1000 and 1500 days are shown in the left plots (a, c + e) and the survival quantiles 0.8, 0.65 and 0.5 in the right plots (b, d + f). The values are colorcoded as indicated in the legends. We differentiate both study arms and estimation variants on the x-axis, both censoring types and three timepoints or quantiles on the y-axis of the plots. Sometimes the NPMLE (step) functions were not defined at the readout quantile in all datasets. Then the number of datasets with a valid estimate is displayed in brackets below the RMSE value. Estimates from less than 900 datasets do not get the color coding because they may be subject to bias.

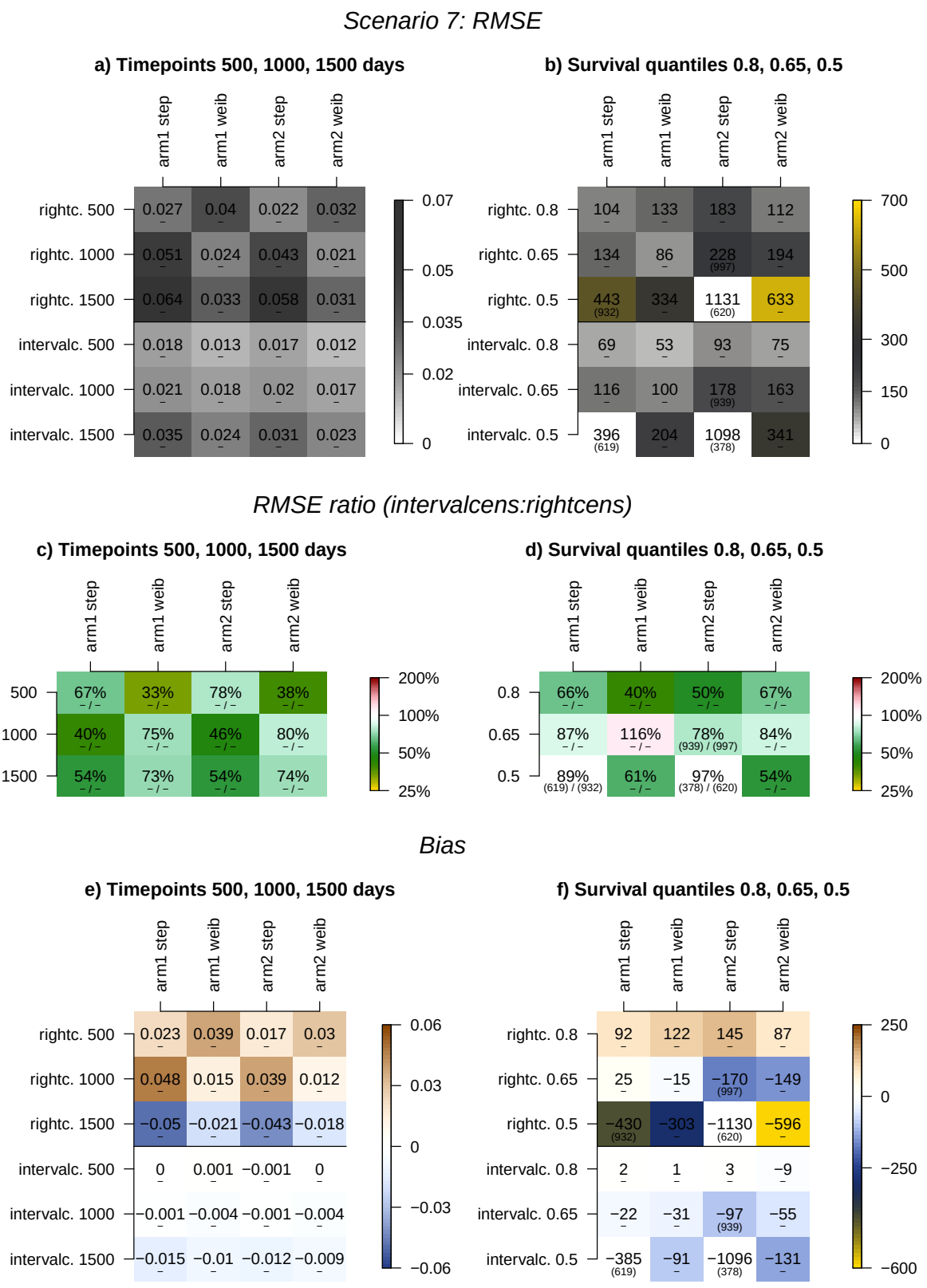

Figure 4.6: RMSE (a + b), RMSE ratio (c + d) and bias (e + f) estimates of 1000 datasets of Scenario 7 in colors. The estimates at timepoints 500, 1000 and 1500 days are shown in the left plots (a, c + e) and the survival quantiles 0.8, 0.65 and 0.5 in the right plots (b, d + f). The values are colorcoded as indicated in the legends. We differentiate both study arms and estimation variants on the x-axis, both censoring types and three timepoints or quantiles on the y-axis of the plots. Sometimes the NPMLE (step) functions were not defined at the readout quantile in all datasets. Then the number of datasets with a valid estimate is displayed in brackets below the RMSE value. Estimates from less than 900 datasets do not get the color coding because they may be subject to bias.

are quite dramatic and more extreme in the upper half of the plots, what means in the rightcensoring estimation setting. All RMSE changes , as shown in Figure 4.6 (a + b) are increases. This is expected, because we have much less information for the progression events if we only have half the number of assessments.

However, the size of the increase in rightcensoring RMSE varies between the stages of the curve and the estimation functions. In very early stage (at timepoint 500 days) the increase is much bigger for the Weibull function. In all later stages (timepoints 1000 and 1500 days, quantiles 0.8 and 0.65) there is observed a bigger increase for NPMLE functions. Only in the too late stage the increase is bigger for the Weibull function again. So the same effect that has been detected for Scenario 1, that the Weibull estimation with rightcensoring methods is less accurate in the extremes of the functions, is seen here even to a bigger extent.

Intervalcensoring RMSE did not change much, even with the halved number of assessments. And with intervalcensoring the RMSE increased a bit nearly only with NPMLE functions. So intervalcensoring estimation is quite robust to these dramatic changes in the assessment schedule.

These big differences between righcensoring and intervalcensoring RMSE estimates are translated into dramatically decreased percentages in the censoring ratio plot in Figure 4.6 (c + d) compared to Scenario 1. The much more green and yellow coloured plot reflects this shift.

Bias

Scenario 5: add two weeks to all intraassessment periods Figure 4.5 (e + f) shows the size of bias values for Scenario 5. The decreased and increased RMSE values with respect to Scenario 1 that we had observed in Figure 4.5 (a + b) for rightcensored estimates are reflected in increased and decreased bias values in Figure 4.5 (a + b). Only the increased RMSE value of the 0.8 quantile is not reflected in an increased bias value. We can summarize that in general the RMSE changes in Scenario 5 can be explained by bias changes, but not for all the cases. The changes might be too small to be separable from simulation error.

Scenario 7: leave out every second assessment The increases in bias in Figure 4.6 (e + f) are also reflections of the increases in RMSE in plots (a + b) of the same figure. Again, RMSE changes in this scenario can be explained by bias changes.

We can conclude from this scenario: the fewer assessments, the bigger the difference between right- and intervalcensoring estimation, and therefore the bigger the accuracy gain by using intervalcensoring estimation methods.

Before we go on we look a bit closer to estimation error at five different timepoints of the nonparametric survival function, that is shown in Figure 4.7. There we see that rightcensoring RMSE and bias can vary a lot depending on the exact place of the curve we look at. We see that at the timepoints that were not considered for the fingerprint plots, 750 days and 1250 days, the RMSE and bias of the rightcensored estimates is much smaller than for the other timepoints, 500, 1000 and 1500 days. This discrepancy might make sense when we compare it to the assessment schedule. We see there that e.g. the timepoints 1000 days and 1500 days are close to a planned assessment timepoint, but e.g.

the timepoint 1250 days is about in the middle between two planned assessments. So we might be able to explain the lower RMSE and bias value at 1250 days by its position in the middle of an intraassessment period, where it might fit better to the true function.

We know that the nonparametric rightcensoring survival function reacts a lot to the assessment timepoints by forming a "bump" on the curve whenever an assessment is done, because all progression events are mapped to the assessment timepoint. This behaviour is more extreme in Scenario 7, because there are less assessments, and it might explain the big differences in estimation accuracy between timepoints in this scenario.

Intervalcensoring estimation is much more stable in this respect, as we see a constant slight increase in intervalcensoring RMSE values in Figure 4.7. Also parametric estimation is more stable, what is shown in Figure 4.8.

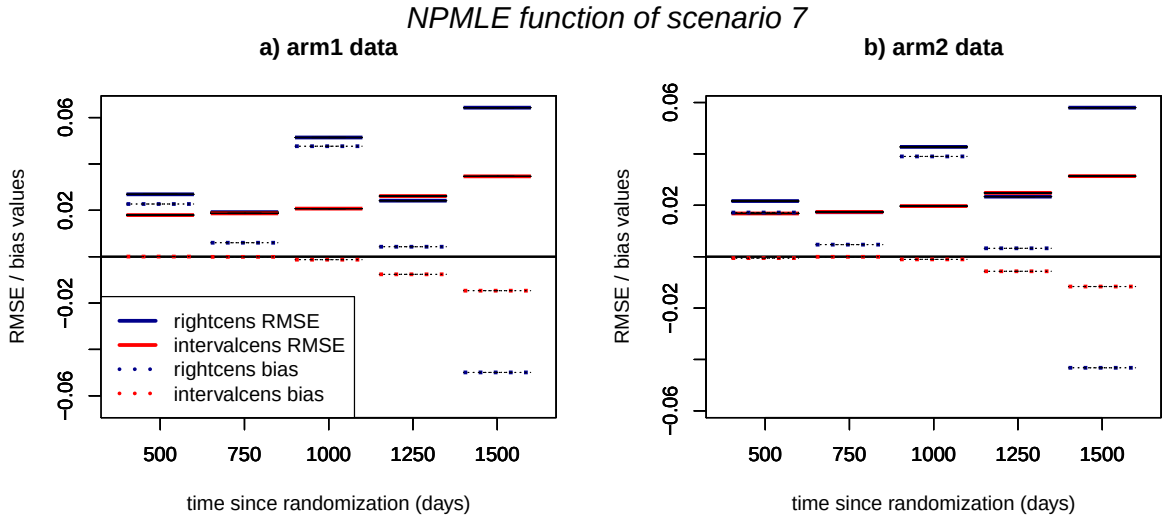

Figure 4.7: RMSE and bias values of NPMLE function estimates of Scenario 7 in order of appearance on the function: Solid lines represent the RMSE, dotted lines the bias values. We look at arm1 in plot a) and at arm2 in plot b). We compare the estimates at five timepoints: 500 days, 750 days, 1000 days, 1250 days and 1500 days.

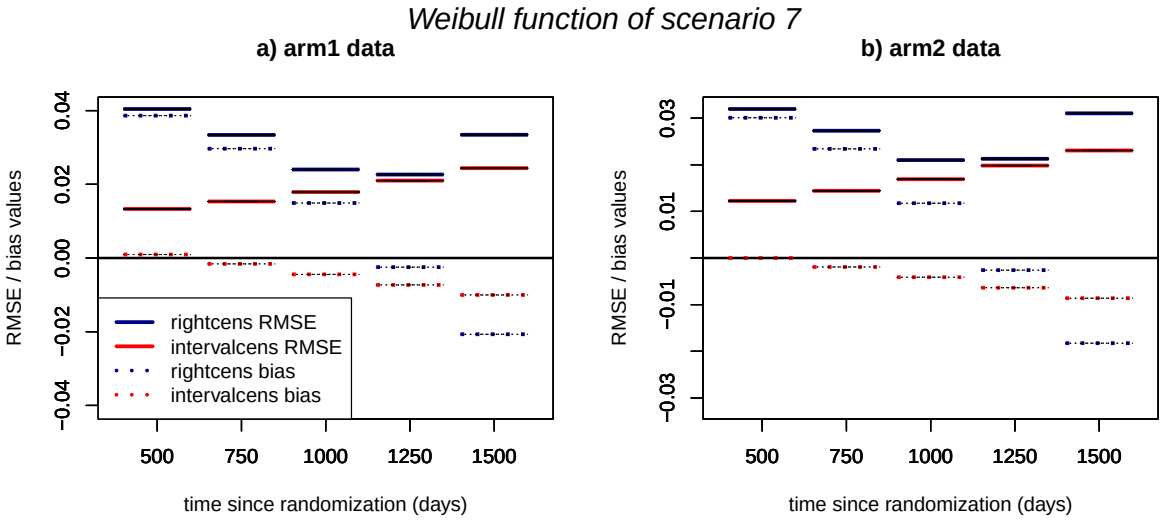

Figure 4.8: RMSE and bias values of Weibull function estimates of Scenario 7 in order of appearance on the function: Solid lines represent the RMSE, dotted lines the bias values. We look at arm1 in plot a) and at arm2 in plot b). We compare the estimates at five timepoints: 500 days, 750 days, 1000 days, 1250 days and 1500 days.

4.1.4 Scenarios 8 and 9: change in proportion of deaths

RMSE

Scenario 8: smaller proportion of death events Estimations with a proportion of death events of about 10% are tested in Scenario 8 and result in the RMSE values shown in Figure 4.9 (a + b). They show a slight increase in the rightcensoring RMSE values for both function types at early stage of the curve (timepoints 500 and 1000 days, quantile 0.8), but not anymore at late stage of the curve. These shifts can be explained by the increase in proportion of progression events, what increases the bias for rightcensoring estimations.

These shifts are also observed in Figure 4.9 (c + d), where they manifest in slightly decreased percentages for early stage points. The even slighter increase of percentages that is observed at late stage in arm1 NPMLE function estimates might be neglectable.

Scenario 9: bigger proportion of death events Estimations with a proportion of death events of about 50% are tested in Scenario 9 and result in the RMSE values shown in Figure 4.10 (a + b). Here we observe at early stage and for rightcensoring RMSE values the opposite shifts of those observed in Scenario 8. This is logical, because opposite to Scenario 8, in this scenario the proportion of progression events is smaller and therefore the amount of bias on rightcensored estimates might be also smaller. However, at late stage (timepoint 1500 days and quantile 0.65) of the curve the result diverges from the one of Scenario 8. Here the RMSE slightly increases in both censoring variants. Another unknown effect might play a bigger role at that stage of the curve.

The shifts at early stage of the rightcensoring RMSE are also observed in Figure 4.10 (c + d), where they manifest in slightly increased percentages for early stage points. At late stage also a slight increase in percentages is observed.

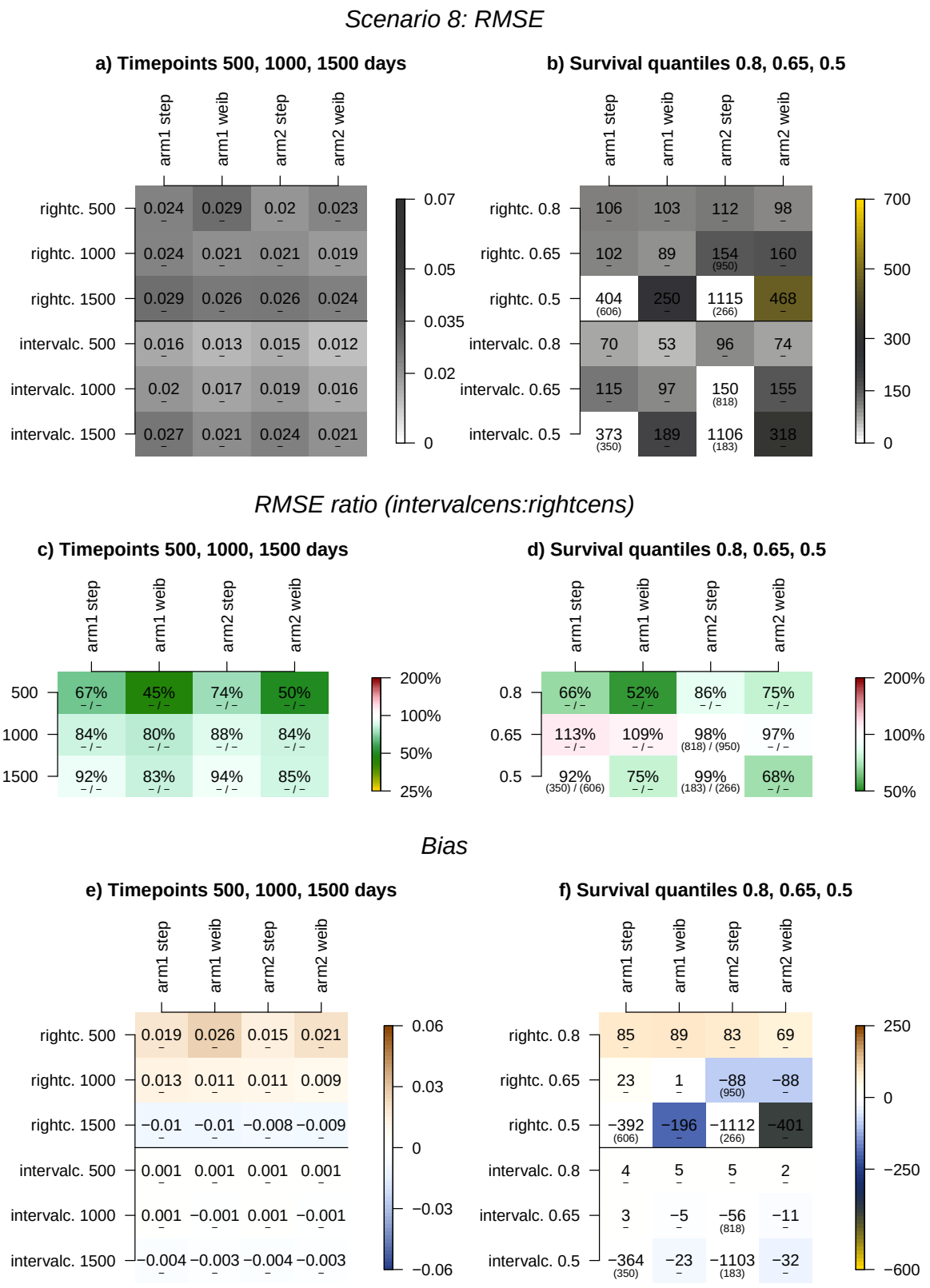

Figure 4.9: RMSE (a + b), RMSE ratio (c + d) and bias (e + f) estimates of 1000 datasets of Scenario 8 in colors. The estimates at timepoints 500, 1000 and 1500 days are shown in the left plots (a, c + e) and the survival quantiles 0.8, 0.65 and 0.5 in the right plots (b, d + f). The values are colorcoded as indicated in the legends. We differentiate both study arms and estimation variants on the x-axis, both censoring types and three timepoints or quantiles on the y-axis of the plots. Sometimes the NPMLE (step) functions were not defined at the readout quantile in all datasets. Then the number of datasets with a valid estimate is displayed in brackets below the RMSE value. Estimates from less than 900 datasets do not get the color coding because they may be subject to bias.

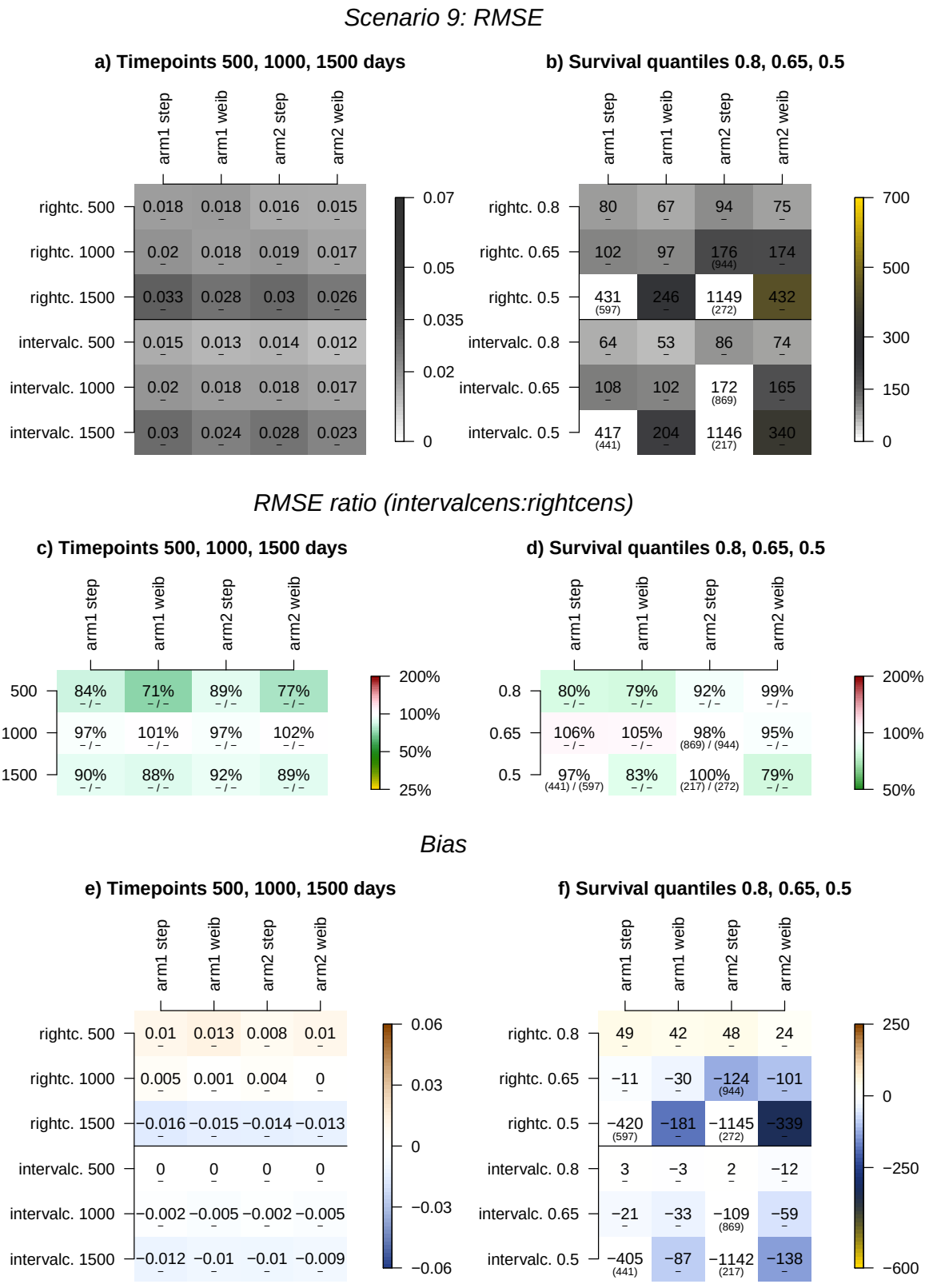

Figure 4.10: RMSE (a + b), RMSE ratio (c + d) and bias (e + f) estimates of 1000 datasets of Scenario 9 in colors. The estimates at timepoints 500, 1000 and 1500 days are shown in the left plots (a, c + e) and the survival quantiles 0.8, 0.65 and 0.5 in the right plots (b, d + f). The values are colorcoded as indicated in the legends. We differentiate both study arms and estimation variants on the x-axis, both censoring types and three timepoints or quantiles on the y-axis of the plots. Sometimes the NPMLE (step) functions were not defined at the readout quantile in all datasets. Then the number of datasets with a valid estimate is displayed in brackets below the RMSE value. Estimates from less than 900 datasets do not get the color coding because they may be subject to bias.

Bias

Bias estimates for Scenarios 8 and 9 are shown in Figure 4.9 (e + f) and Figure 4.10 (e + f). All plots reflect closely the shifts in RMSE of the respective scenarios. That means that the shifts in RMSE that were explained above might mainly come from shifts in bias. This was already expected in the section above, because then we have an explanation for the shifts, especially for those at early stage of the curve.

We can conclude that the bigger the proportion of intervalcensored events from all events, the more we can profit from intervalcensoring estimation compared to rightcensoring estimation. Intervalcensoring estimation should therefore be considered especially in studies with high proportions of truly intervalcensored events, like eg. rogression events.

4.1.5 Scenario 10 and 11: increase in percentage of dropout

RMSE

We estimate with an increased dropout rate of 10% for both arms in scenario 10. The resulting RMSE values for this scenario are shown in Figure 4.11 (a + b). By comparing them to Scenario 1 we see that the biggest shifts occur at late stage of the curve, especially at timepoint 1500 days. There the rightcensoring RMSE values decrease, the intervalcensoring values also slightly.

The shifts are so small that they result only in slight changes in intervalcensoring ratio percentages in Figure 4.11 (c + d). Anyway, these small changes are increases, that means with more dropouts the advantage of intervalcensoring estimation methods is decreasing.

Scenario 11 is a mixture of Scenarios 10 and 1 because arm1 has 10% dropout per year and arm2 2.5% dropout. But there are slight changes in number of events and dropouts per arm in this scenario compared to Scenarios 1 and 10, as it will be shown in Figure 4.15 in the later Section 4.2.2 of that thesis. Anyway, these changes do not lead to different RMSE and bias values in the fingerprint plots and therefore they are only shown in Figure 8.4 in Appendix Section 8.3.

Bias

As we see in Figure 4.11 (e + f) the negative bias at late stage timepoint 1500 days is decreased for both censoring variants. Therefore the RMSE decrease might come from this bias decrease. At late stage quantile 0.65 the negative bias decreases as well or even changes to positive bias.

To summarize it looks like estimation gets more accurate when we have more rightcensorings due to dropouts (instead of righcensorings at analysis cutoff, because the total number of events does stay the same). This especially at later stages of the curve. This makes sense, because in this scenario we might have to wait longer until we can observe the target number of events, because more patients drop out from the study. So we have more events at later timepoints. This makes estimation at later timepoints more precise.

The bigger ratios of intervalcensoring : rightcensoring RMSE at all stages are more difficult to explain. It might be the case that we have here a bigger share of information from the truly rightcensored observations than in Scenario 1 and therefore the share of information coming from incorrectly rightcensored observations (being progression events)

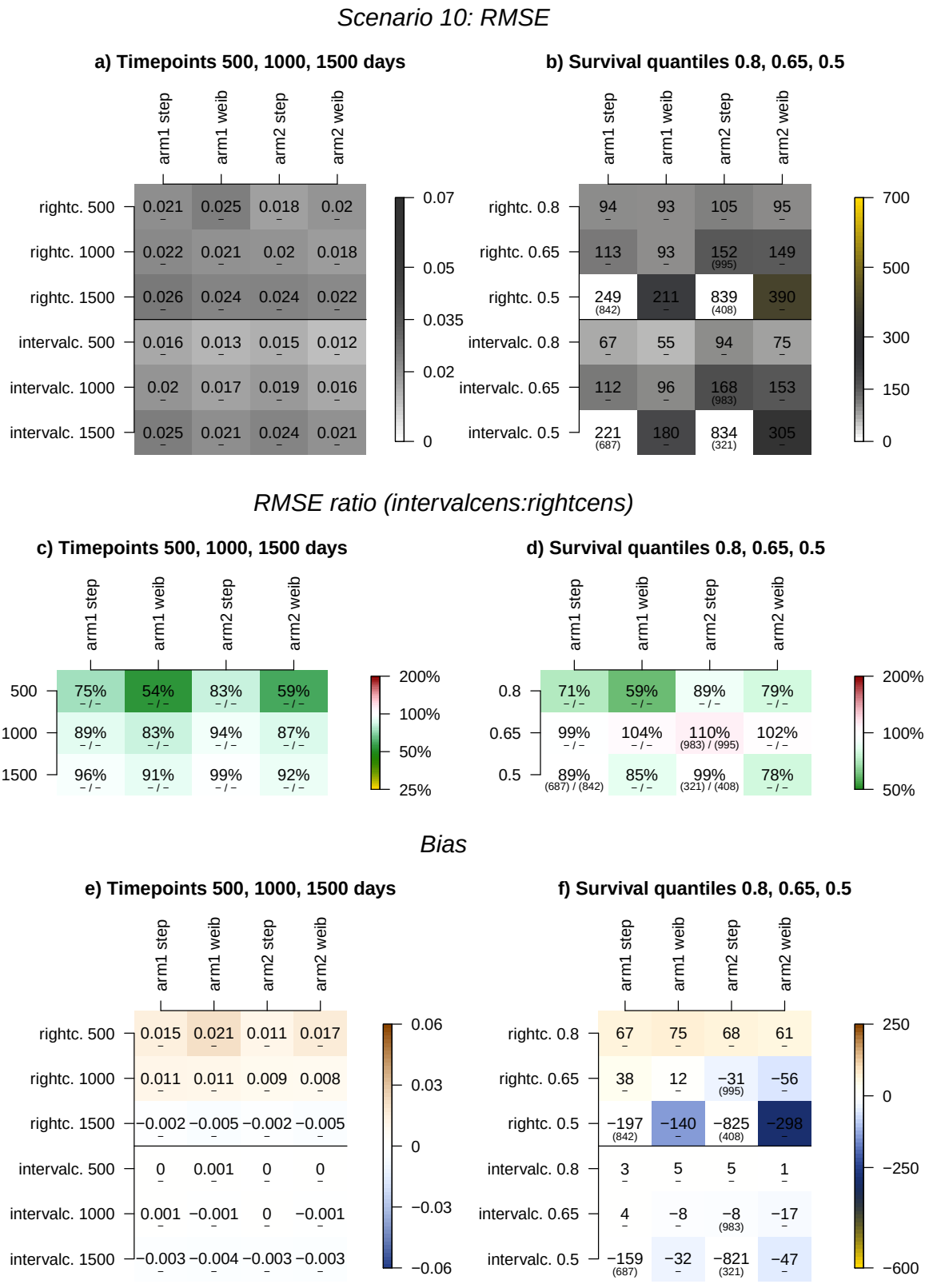

Figure 4.11: RMSE (a + b), RMSE ratio (c + d) and bias (e + f) estimates of 1000 datasets of Scenario 10 in colors. The estimates at timepoints 500, 1000 and 1500 days are shown in the left plots (a, c + e) and the survival quantiles 0.8, 0.65 and 0.5 in the right plots (b, d + f). The values are colorcoded as indicated in the legends. We differentiate both study arms and estimation variants on the x-axis, both censoring types and three timepoints or quantiles on the y-axis of the plots. Sometimes the NPMLE (step) functions were not defined at the readout quantile in all datasets. Then the number of datasets with a valid estimate is displayed in brackets below the RMSE value. Estimates from less than 900 datasets do not get the color coding because they may be subject to bias.

is smaller and thus the difference between rightcensoring and intervalcensoring estimation gets smaller. Anyway, this effect is quite small.

4.1.6 Scenario 12: increase in events at cutoff

RMSE

The last scenario with 520 events is also quite interesting to look at. Its RMSE values are shown in Figure 4.12 (a + b). At late stage (timepoint 1500 days and quantile 0.65) we see a decrease in RMSE for both censorings, functions and arms (with one exception, the RMSE at 0.65 quantile for rightcens and arm1 is increasing). At early stage of the curve (timepoint 1000 days and quantile 0.8) the Weibull function RMSE of rightcensoring are increasing. Only the very early stage (timepoint 500) RMSE values stay about the same.

Both these shifts are a sign that in this scenario the early stage is prolonged and the late stage shifted to later timepoints and smaller quantiles. Also in this scenario for more datasets the median is reached with the NPLME function.

When looking at the censoring ratio plots that are shown in Figure 4.12 (c + d) we see that here the percentages decrease for late early stage (timepoint 1000 days and quantile 0.8) and they increase a bit in late stage. These slight changes are also a sign that the stage border is shifted in this scenario.

This makes sense when we imagine the distribution of events along the Survival curve here. In Scenario 12 we have more events in the middle and late stage of the curve and some events occur at later timepoints and lower quantiles than before. So the estimation precision and the specific characteristics of estimation are also shifted along the survival function.

Bias

The shift of curve stage border can also be read from the bias estimates in Figure 4.12 (e + f). In rightcensoring at early and late stage the bias is shifted to more positive values and in intervalcensoring at late stage it is shifted as well.

These shifts in RMSE and bias to later times can also be seen nicely in the timecourse plots in Figure 4.13.

It seems that at late stage of the curve there is always a negative bias present, for both censorings and in all scenarios. This seems to be a specific property of survival function estimation that occurs for both function types and censoring analysis methods. The bias is smaller with intervalcensoring methods, but still present.

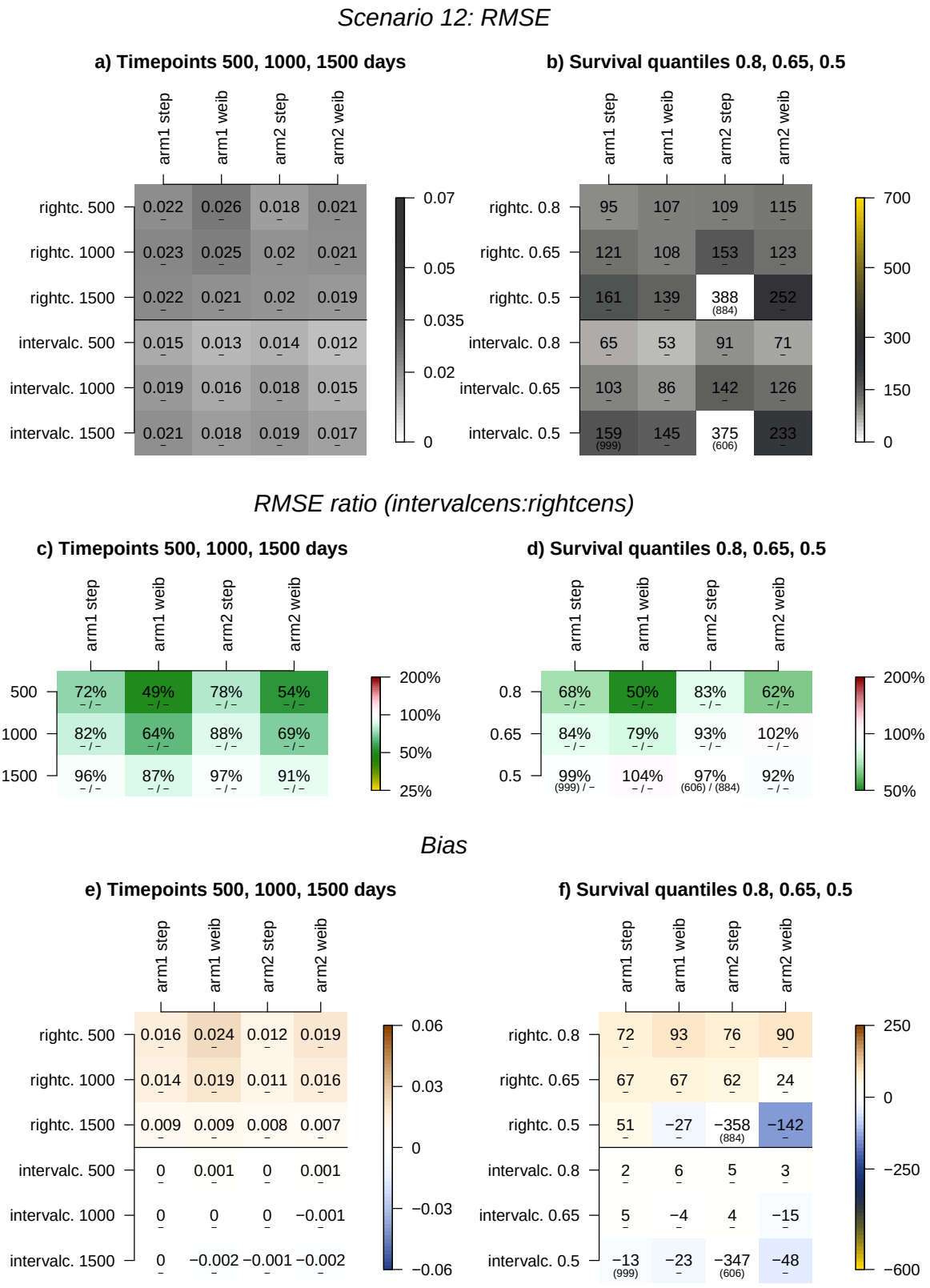

Figure 4.12: RMSE (a + b), RMSE ratio (c + d) and bias (e + f) estimates of 1000 datasets of Scenario 12 in colors. The estimates at timepoints 500, 1000 and 1500 days are shown in the left plots (a, c + e) and the survival quantiles 0.8, 0.65 and 0.5 in the right plots (b, d + f). The values are colorcoded as indicated in the legends. We differentiate both study arms and estimation variants on the x-axis, both censoring types and three timepoints or quantiles on the y-axis of the plots. Sometimes the NPMLE (step) functions were not defined at the readout quantile in all datasets. Then the number of datasets with a valid estimate is displayed in brackets below the RMSE value. Estimates from less than 900 datasets do not get the color coding because they may be subject to bias.

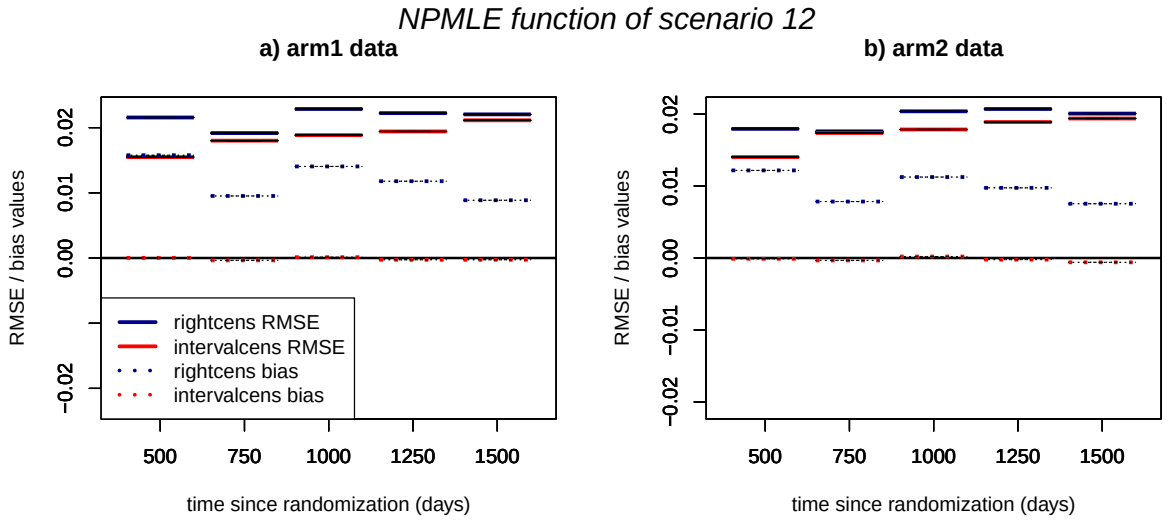

Figure 4.13: RMSE and bias values of NPMLE function estimates of Scenario 12 in order of appearance on the function: Solid lines represent the RMSE, dotted lines the bias values. We look at arm1 in plot a) and at arm2 in plot b). We compare the estimates at five timepoints: 500 days, 750 days, 1000 days, 1250 days and 1500 days.

4.2 Test for identity

As a test for identity of the two study arms, the logrank test was used, as described in Section 2.3. By testing two different types of studies (with a treatment effect and with no effect), two types of properties of the logrank test could be tested. This is on one hand the property to reject the null hypothesis for a study with a hazard rate of 1 with the correct probability of a Type I error and on the other hand the probability to reject the null hypothesis for a study with a hazard rate of 0.76 (the hazard rate of GALLIUM simulations). For the power, the underlying probability of rejecting a hazard ratio of 0.76 is approximated from Schoenfeld's sample size formula. This calculation is explained in Section 4.2.2.

The quality of the logrank test is highest when the proportion of rejected true null hypotheses is exactly matching the α-level, which is a two-sided 5% here. If the proportion of rejected true nulls is well below the α-level, the test is called conservative, because it does detect an effect at a lower proportion than indicated by the p-value. If the proportion of rejected hypotheses is well above the α-level, the test is called liberal, because it does detect an effect too often.

One part of the logrank test estimation process for intervalcensored data is nonparametric estimation of the survival function, as described in Section 2.3.1. For some datasets the algorithm used in this process does not converge and then no test statistic could be obtained. The test statistic of the corresponding rightcensoring analysis was also omitted for the nonconverged datasets to make the percentages of rejected datasets more comparable. It is assumed that nonconvergence is not related to the simulated dataset, otherwise the final percentages would be biased. The number of datasets that were amenable to estimation is indicated in the bottom of the plots in Figures 4.14 and 4.15.

4.2.1 Type I error

The Type I error is determined empirically by testing datasets that are following the null hypothesis of the test. The results of this estimation for all the scenarios are shown in Figure 4.14 and explained in the next sections.

Scenario 1: like GALLIUM study

For the baseline Scenario 1, the empirical Type I error is shown in Figure 4.14 a). The proportion of rejected null hypothesis is slightly above the targeted 5% and therefore the test is slightly liberal for this data. Anyway, this slight deviance from the ideal is seen for both types of logrank tests, for rightcensoring and intervalcensoring data.

Scenarios 2 to 7: change in assessment times

When we look first at the Scenarios 2 to 4 with differences in intraassessment period length variation, whose results are shown in Figure 4.14 b) to d), we see an effect on the Type I error.

Scenario 2 has an about 20% elevated Type I error, about the same for both rightcensoring and intervalcensoring data. It seems that the logrank test cannot cope well with identical assessment times for all observations and gives a too low p-value for rejection of the null hypothesis. That means this statistic is too liberal for this special data.

The empirical Type I error of Scenario 3 matches with the significance level. To summarize the results of Scenarios 1 to 3, it seems that the test is best if there is a lot of random normal variation of the intraassessment period lengths. The standard normal variation of 10 days standard deviation of Scenario 1 does seem to be not high enough to reach a correct Type I error.

Scenario 4 with variation difference between the two arms seems to perform about the same as standard Scenario 1. Its Type I error estimate is too high as well what means the statistic is liberal. The logrank tests seem to loose some performance if the random variation of the two arms is not the same.

When looking at Scenarios 5 and 6 whose results are shown in Figure 4.14 e) and f), we see not much change compared to Scenario 1. The increase of the interval lengths by two weeks seems to have not much effect on the precision of the Type I error.

Scenario 7 in Figure 4.14 g) gives too low percentages of rejected hypotheses for both rightcensoring and intervalcensoring methods. The conservativity of the test with this type of data is quite pronounced with an about 20% reduced Type I error. In this scenario, half the assessments are not done what results in double the length of intraassessment periods and half the number of assessments. One of these changes seems to be too extreme for the test to cope with, in order that it does not reach the wanted Type I error any more.

For all these scenarios the rigthcensoring and intervalcensoring results are comparable. The intervalcensoring test seems to have very similar properties as the rightcensoring test for Type I error detection when assessment times are varied.

Scenarios 8 and 9: change in proportion of deaths

Results for Scenarios 8 and 9 are shown in Figure 4.14 h) and i).

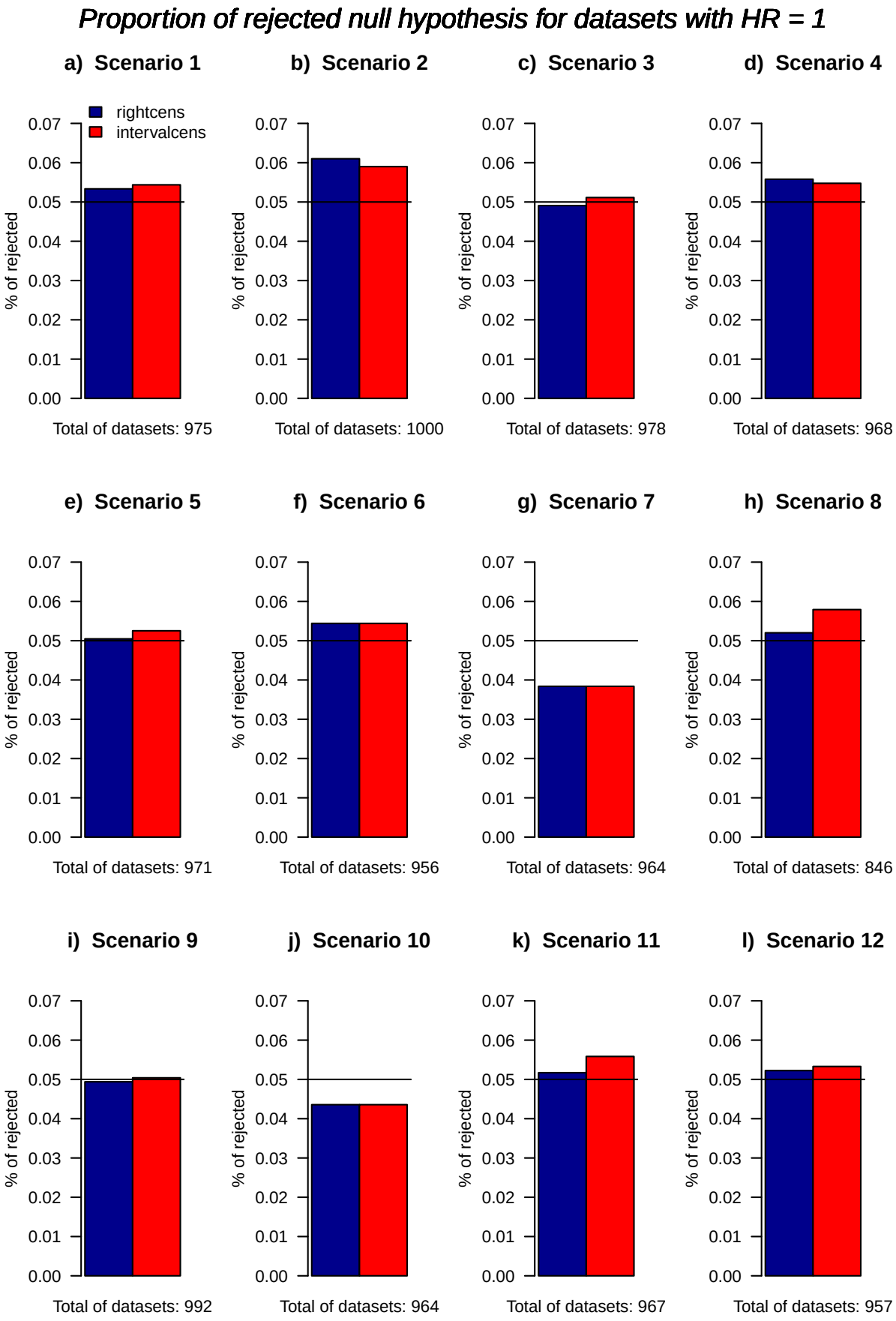

Figure 4.14: Proportion of rejected null hypothesis: Proportion of p-values < 0.05 of all datasets with true hazard ratio 1 that were logrank-tested successfully. Scenarios 1 - 12 in plots a) to l) with two values each, determined by logrank-testing with rightcensoring and intervalcensoring methods. The target α-level is 0.05, as indicated by a line.

The reduction in the proportion of deaths to 10% seems to influence the intervalcensoring test negatively by increasing its proportion of rejected and making it too liberal. This effect is not very pronounced and it is unsure whether this result is just due to simulation error.

When the proportion of deaths is increased to 50%, the calculated test statistic seem to correspond well with the correct Type I error, even better than with the standard proportion of deaths of 25% of Scenario 1.

Scenarios 10 and 11: increase in percentage of dropout

Figure 4.14 j) indicates that the performance of the test statistics changes for both rightcensoring and intervalcensoring methods when the percentage of yearly dropout is increased to 10% in both arms. The logrank test gets too conservative in both cases of censoring. The effect seems to be hidden if only one arm has increased dropout as it is shown in Figure 4.14 k).

Scenario 12: increase in events at cutoff

We see in Figure 4.14 l) that an increase of events at cutoff does not change the performance of both rightcensoring and intervalcensoring tests remarkably. We can conclude that the 370 events of Scenario 1 are enough for the logrank test to achieve its postulated properties.

4.2.2 Power

Power calculations for the GALLIUM study have been done based on an Exponential distribution and taking into account interim analyses. We are using a Weibull distribution for simulation and have only one final analysis timepoint. Therefore, we have to calculate the power in this context ourself. The sample size formula for survival data that only assumes proportional hazard and needs the hazard ratio of a study with two same-size arms is:

$$\texttt{n.events} = \frac{4 \cdot (z_{1-\beta} + z_{1-\frac{\alpha}{2}})^2}{(\log(\text{HR}))^2} \tag{4.1}$$

where in general z_α is the α-quantile of a standard Normal distribution and $1-\beta = \text{power}, \alpha = \text{significance level}, \texttt{n.events} = \text{total no of events}, \text{HR} = \text{hazard ratio}$. This sample size formula has been described by Schoenfeld in [30].

For our purposes we have to solve this equation for power:

$$z_{1-\beta} = \sqrt{\frac{\texttt{n.events} \cdot (\log(\text{HR}))^2}{4}} - z_{1-\frac{\alpha}{2}} \quad . \tag{4.2}$$

Then we have to calculate the hazard ratio for our Weibull distribution context. By using the hazard function of Section 2.2.1, setting its parameters $\mu = \texttt{scale}$ and $\alpha = \texttt{shape}$ and simplifying the ratio of the hazard functions of the two treatment arms,

$$\text{HR} = \frac{h_2(t)}{h_1(t)} = \frac{\frac{1}{\texttt{scale}_2} \cdot \texttt{shape}_2 \cdot \left(\frac{1}{\texttt{scale}_2} \cdot t\right)^{\texttt{shape}_2 - 1}}{\frac{1}{\texttt{scale}_1} \cdot \texttt{shape}_1 \cdot \left(\frac{1}{\texttt{scale}_1} \cdot t\right)^{\texttt{shape}_1 - 1}} \quad , \tag{4.3}$$

we get

$$\mathrm{HR} = \frac{\texttt{scale}_1^{\texttt{shape}}}{\texttt{scale}_2^{\texttt{shape}}} \quad , \tag{4.4}$$

if the `shape` parameter is the same for both arms ($\texttt{shape} = \texttt{shape}_1 = \texttt{shape}_2$), what is the case in our simulations. By inserting $\texttt{scale}_1 = 3293, \texttt{scale}_2 = 4446, \texttt{shape} = 0.9$ we get the hazard ratio HR = 0.76.

Inserting $\texttt{n.events} = 370, \mathrm{HR} = 0.76, \alpha = 0.05$ into (4.2), we get $1 - \beta = 0.74$ for Scenarios 1 - 11. For Scenario 12 we use $\texttt{n.events} = 520$ in (4.2) and get $1 - \beta = 0.87$. These values are tepicted in the legend of Figure 4.15.

The 74% power in most of the scenarios here is smaller than the power that was used for sample size calculations of the GALLIUM study. There, to get a power of 80%, the total number of events needed was 370.

The empirical power to detect an assumed hazard ratio for a given simulation scenario is determined by testing many datasets that have been simulated according to that scenario and then calculating the percentage of datasets that reject the null hypothesis. This percentage should correspond well to the calculated power of that study. It is shown for all scenarios in Figure 4.15 and its behaviour is discussed in the next sections.

Scenario 1: like GALLIUM study

As we see in Figure 4.15 a), the theoretical power level is reached quite accurately with the simulated datasets. There is no relevant difference between the rightcensoring and intervalcensoring tests.

Scenarios 2 to 7: change in assessment times

We look first at Scenarios 2 to 4 that vary the variation in intraassessment period lengths. Figure 4.15 b) to d) show the percentage of rejected datasets for these scenarios. They are all very close to the theoretical power level. So the logrank test seems to perform very well in this setting, different to the Type I error, where only Scenario 3 with highly variable intraassessment period lengths had a very precise percentage of rejected datasets.

There seems to be a difference between Scenarios 5 and 6, whose results are shown in Figure 4.15 e) and f). When both study arms have longer intraassessment intervals, the study power is still reached, but when only the intervals of the treatment arm are changed, a decrease in power of 2 to 4% is detected. This decrease is bigger with the rightcensoring method than with the intervalcensoring method. It might be related to the violation of the noninformative censoring assumption in Scenario 6.

In Scenario 7 that is shown in Figure 4.15 g) there is also a decrease in power detected, even if the two arms do not differ here in the assessment schedule. But this decrease is smaller and only about 1 to 2%.

In summary the power is met in most of these scenarios. Only if the assumption of informative censoring is not met (in Scenario 6) and if the intraassessment period length is too long or the number of assessments too small (in Scenario 7) the test statistic is too conservative. In both these cases the intervalcensoring method is closer to the targeted power than the rightcensoring method.

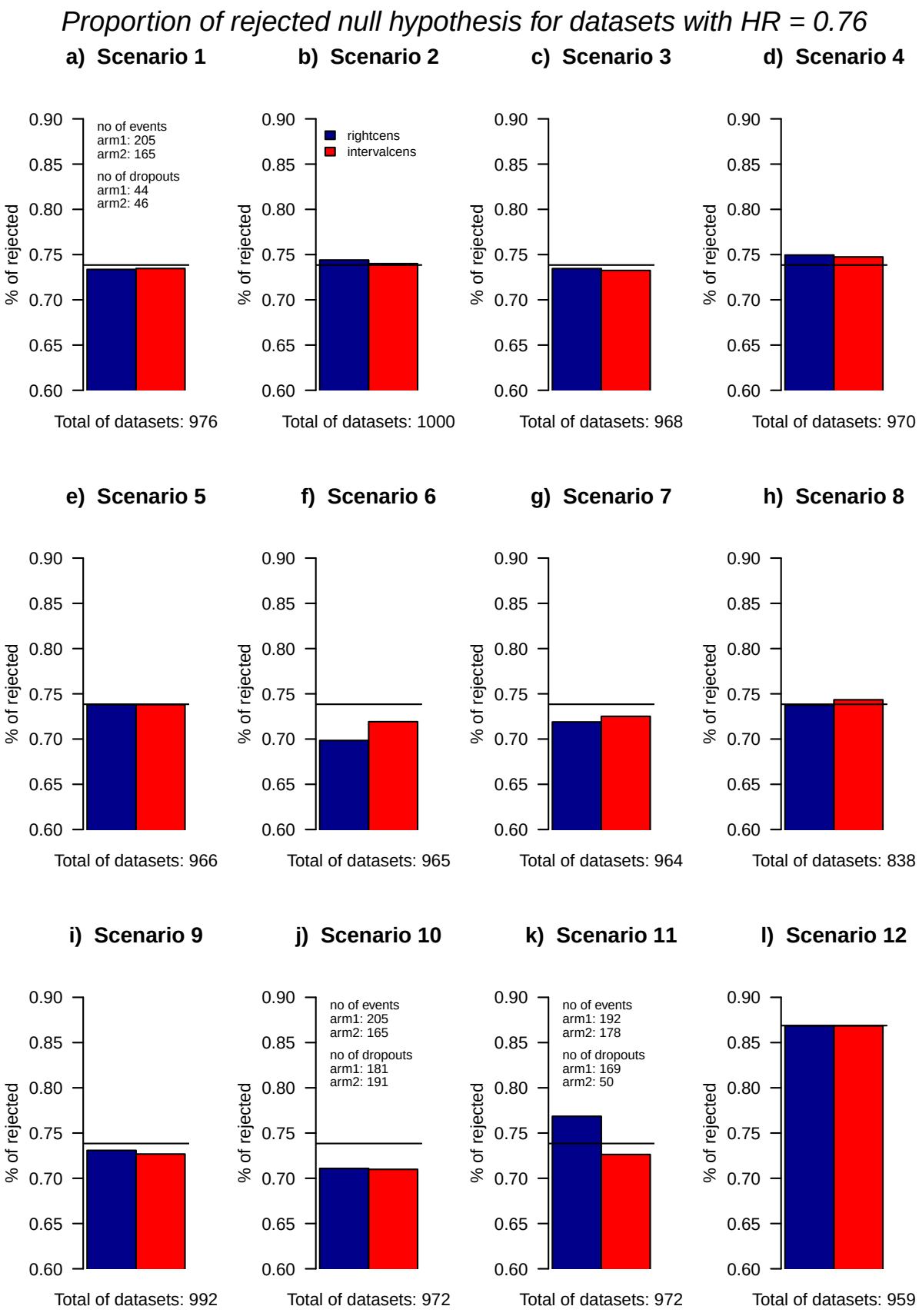

Figure 4.15: Proportion of rejected null hypothesis: Proportion of p-values < 0.05 of all datasets simulated like GALLIUM (hazard ratio 0.76) that were logrank-tested successfully. Scenarios 1 - 12 in plots a) to l) with two values each, determined by logrank-testing with rightcensoring and intervalcensoring methods. The target power is 0.74 for Scenarios 1 - 11 and 0.87 for Scenario 12, as indicated by a line. In a, j and k also the number of events and dropouts per arm are indicated, to serve for a possible explanation of the results.

Scenarios 8 and 9: change in proportion of deaths

Figure 4.15 h) and i) shows that the change in proportion of deaths does not have a big impact on the percentage of rejected datasets

Scenarios 10 and 11: increase in percentage of dropout

The results for Scenarios 10 and 11 that are shown in Figure 4.15 j) and k) are quite surprising. An increase of dropouts in both arms (as in Scenario 10) seems to decrease the power of detecting a difference with the logrank test for both estimation methods, but if the increase in dropouts is only in the control arm (as in Scenario 11) it seems to have the opposite effect for the rightcensoring method and less of the same effect in the intervalcensoring method. The intervalcensoring result is less surprising here, the power decrease seems to be related to the amount of dropouts. This makes sense, because when we have more dropouts (that are rightcensorings at early timepoints) compared to rightcensorings at the analysis timepoint, we have less information about the shape of the curve. Less information leads then to smaller power in detection of a difference between the curves.

It is difficult to say why the dropout increase in just one arm is inverting the effect of the dropout increase in both arms on power in the rightcensoring framework. The smaller difference between the number of events between the two arms (see numbers in the figure) cannot be the cause of such a shift. Maybe the rightcensoring logrank test is reacting more to violation of the noninformative censoring assumption in Scenario 11. The intervalcensoring method seems to be more robust towards such violations. After all, the effect we describe here is only small.

Scenario 12: increase in events at cutoff

In a study with more events the power is increased, as expected, and the percentage of rejected exactly matches the theoretical power, see Figure 4.15 l).

Chapter 5

Discussion

5.1 Estimation of survival function

5.1.1 Baseline Scenario 1

An overlay of different effects can explain the size of the RMSE values.

On one side there is the decreasing estimation precision in course of time because of a decreasing amount of events per time. This leads to increasing RMSE values in course of time, for both censoring methods.

Then there is the positive bias introduced by incorrectly assigning progression events to the first following assessment time in rightcensoring estimation. This is especially visible at early timepoints (and high quantiles), maybe because this effect is masked by a negative bias at later timepoints, that is explained in the next paragraph.

The observation of negative bias at late timepoints (and low quantiles) has led us to the thought that nonparametric MLEs of survival functions might be biased in general. A negative bias has been described and quantified in [32] for Kaplan-Meier estimators. This explains our observations for rightcensored estimates. We hypothesize that the Turnbull estimator might be biased in a similar way.

When comparing parametric and nonparametric estimators we saw that the rightcensoring parametric estimator has a bigger RMSE in the extremes of the function, being very early timepoints (very high quantiles) and very late timepoints (very low quantiles). This phenomenon could not be explained easily.

By comparing the study arms it was expected to see smaller RMSE values in the control arm (arm1), because it contained a bigger proportion of events and therefore the estimates could be determined more precisely. But this was only true for point estimation at quantiles, at timepoints the relation of the arms was opposite. This effect is difficult to explain.

As a general guidance for point estimations on survival functions with data similar to the GALLIUM study, we can say that especially when estimating points on early stage of the function it is recommended to use intervalcensoring methods, because they are not only much less biased, but also have a smaller RMSE. At late stage of the function the intervalcensoring method is still better, but the difference to the rightcensoring method is smaller.

5.1.2 Deviations in other scenarios

Differences in the variance of intraassessment period length (Scenarios 2 – 4) does not lead to substantial changes in the RMSE pattern.

When the assessment periods are prolonged (as done in Scenarios 5 – 7), the RMSE values of rightcensoring estimation are increased, because the bias introduced by progression events is increased. Intervalcensoring estimation is much more robust towards prolongation of assessment periods. Only with doubled assessment periods the intervalcensoring RMSE is increased a bit in NPMLE estimation. We can conclude that the longer the assessment periods the bigger the advantage of intervalcensoring estimation.

When the proportion of death events is changed (as in Scenarios 8 and 9), it changes the rightcensoring RMSE. The more death events (and thus the less progression events) the lower the rightcensoring RMSE. But the rightcensoring RMSE does not go below the level of intervalcensoring RMSE. Thus we get the biggest accuracy gain by doing intervalcensoring estimation in studies with few death events (and many progression events).

When there are more dropouts (as in Scenarios 10 and 11), there is a tendency towards lower rightcensoring RMSE, especially at late stage of the function. We can conclude that the intervalcensoring estimation is especially useful in studies with few dropouts.

When analyzing a study with more events (as in Scenario 12), we observe still the same effects that shape the size of RMSE values. These effects just appear at later timepoints (and lower quantiles) than what we were used in the other scenarios. From this we can conclude that the staging of the survival function (the separation in early and late stage) is an appropriate concept to describe its properties at different areas of the function. This because it is a dynamic concept that can be adapted to survival functions spanning very short or very long time periods.

5.2 Test for identity

5.2.1 Type I error

In general, the rightcensoring and intervalcensoring logrank test statistics behave similar with respect to Type I error. Only maybe in Scenarios 8 and 11 the test statistic could be slightly liberal in intervalcensoring methods and more accurate in rightcensoring methods. So the intervalcensoring logrank test seems to have slightly more problems with a low rate of exact events and maybe also with a varying dropout rate between arms. Anyway, these differences are quite small and it is unclear whether they are prominent enough to be consistent if more simulations were done per scenario. In general the two methods agree quite well for all scenarios we looked at.

There are some general deviations from ideal test statistics detected in Figure 4.14 that might be a bit surprising. One can give some general guidance for some of them. One should be careful in simulating assessment schedules that do not consider any deviation from the schedule as in Scenario 2, as one might detect too many positive results of a difference between study arms if there is none in reality. And one should keep in mind that testing of a study with very long assessment intervals and few assessments as in Scenario 7, or with too many dropouts as in Scenario 10, might result in a too conservative test.

5.2.2 Power

In most of the scenarios the rightcensoring and intervalcensoring methods lead to similar results in terms of power. Only in two of the four scenarios with remarkable deviations (that are Scenarios 6, 7, 10 and 11) the deviations for the rightcensoring logrank test are much bigger than for the intervalcensoring test. These are also the two scenarios where the noninformative censoring assumption is violated (Scenarios 6 and 11). It looks like the intervalcensoring logrank test is more robust with respect to these violations.

This bigger sensitivity of the rightcensoring method towards violation of the noninformative censoring assumption has not been seen in the analysis of Type I error in Section 4.2.1. It seems only to occur with data that comes from a study with a hazard ratio $\neq 1$.

The general deviations from ideal test properties that are detected should also be kept in mind. Too many dropouts (as in Scenario 10) might cause a power loss of the test similar to the reduced Type I error in the last section and a different assessment schedule for the two arms (as in Scenario 6) might as well be a reason for a loss in power. A smaller amount of power loss might be caused by a study with very long assessment intervals and few assessments as in Scenario 7.

5.3 Comparison to other simulation studies

Interesting other simulation studies on comparison of right- and intervalcensoring methods have been done [3, Chapter 10], [35, 31, 36]. These studies compare e.g. logrank tests and get to similar conclusions as our study.

The study described in the book [3, Chapter 10] especially stresses the importance of having the same assessment periods in different treatment arms, because noncompliance with this rule is a mayor source of bias. This has also been observed in our study, where Scenario 6 led to power loss.

Chapter 6

Conclusion and outlook

This simulation study provides an overview of the behaviour of right- and intervalcensoring methods in the context of oncology studies with a PFS endpoint.

While doing the programming and analyzing the results it became clear that other types of estimations could be done to improve knowledge about the behaviour of intervalcensoring estimation methods in this context and to make it usable on a regular basis in clinical trials. Some of these types of estimations are outlined in the next sections.

6.1 Estimation of root mean integrated squared error along the survival curve

Point estimation at a certain point of the survival function can show nicely the errors at that spot but it can never give a complete picture of the errors on the whole curve. As we have seen in Scenario 7 with very long intervalcensoring intervals, depending on where on the function we put that point estimate, the accuracy and precision of the resulting estimate might be quite different. But it is possible to integrate up all the squared error values of point estimates along the survival function and then take the mean and root of the integrated squared error [16]. With this method one gets one RMSE value of the whole curve and one can compare the overall MSE value of the whole curve. This would be interesting to do with the data in this thesis. It was also planned in the beginning to do so, but finally due to time restrictions this task was not completed.

6.2 Estimation of hazard ratio with intervalcensoring methods

The original aim of this thesis also included comparison of estimation methods for the hazard ratio. During the months of preparation of this thesis this turned out to be quite a challenge. There are a lot of articles written about hazard ratio estimation with intervalcensored data and a lot of different methods proposed [10, 11, 27, 34]. But besides one `R` package named `intcox` [18, 27] there is no implementation in `R` available for intervalcensoring data that is related to the Cox-model for rightcensoring data.

Two methods have been tried for this thesis, the `intcox` function in the `intcox` package

and a function written by David Dejardin, a colleague at Roche that combines the methods of Goggins [14] and Pan [28]. The first function was fast and worked fine for small datasets but with bigger datasets as the ones used in the Scenarios described in Section 3.3 it could not start the iteration process anymore. The second function was very slow in the beginning and could be fastened by exchanging all dataframes in the estimation process with matrices and vectors. But this was not enough to enable estimation of 1000 datasets in due time. Finally time was running out to consider parallelization or program another approach and this estimation procedure was not applied to the simulated datasets. The generated estimation functions are anyway added to the Appendix Section 8.2.4.

As communicated by Emanuela Pozzi, another colleague at Roche, there is also no such function implemented in `SAS`, the other statistical software that is used regularly at Roche. Considering this, it would be very important in the future to make available a suitable estimation method of hazard ratios with intervalcensored data in statistical software.

6.3 Application of intervalcensoring methods to real clinical data

During my time at Roche there was a request from a health authority to provide intervalcensoring analysis as a further sensitivity analysis for two clinical studies in oncology that had been submitted for filing. The responsible statistician approached me and together we could apply my estimation functions to her data.

The intervalcensoring results confirmed the results of the rightcensoring analysis. But during this analysis I got aware that normally in clinical trials, two types of analysis are performed, the unstratified analysis only with the treatment covariable, and the stratified analysis, where some additional variables are considered in the estimation of the hazard function and as a consequence the logrank test. With my functions we could only do the unstratified analysis, they do not allow for the addition of stratification factors.

Methods for intervalcensoring analysis considering stratification factors are currently not available in the repositories of `R` or `SAS`. In order that the intervalcensoring methods are becoming more widely used as a sensitivity analysis, it would be needed to establish and program these methods.

Chapter 7

Bibliography

[1] *A Study of Obinutuzumab (RO5072759) Plus Chemotherapy in Comparison With MabThera/Rituxan (Rituximab) Plus Chemotherapy Followed by GA101 or MabThera/Rituxan Maintenance in Patients With Untreated Advanced Indolent Non-Hodgkin's Lymphoma (GALLIUM)*. ClinicalTrials.gov. U.S. National Institutes of Health, 2014. URL: http://clinicaltrials.gov/ct2/show/record/NCT01332968.

[2] D Böhning. "A vertex-exchange-method in D-optimal design theory". In: *Metrika* 33.1 (1986), pp. 337–347.

[3] Ding-Geng D Chen, Jianguo Sun, and Karl E Peace. *Interval-Censored Time-to-Event Data: Methods and Applications*. CRC Press, 2012.

[4] Bruce D Cheson et al. "Revised response criteria for malignant lymphoma". In: *Journal of Clinical Oncology* 25.5 (2007), pp. 579–586.

[5] Thomas D Cook and David L DeMets. *Introduction to statistical methods for clinical trials*. CRC Press, 2007.

[6] Marie Laure Delignette-Muller et al. *fitdistrplus: help to fit of a parametric distribution to non-censored or censored data*. R package version 1.0-2. 2014.

[7] EA Eisenhauer et al. "New response evaluation criteria in solid tumours: revised RECIST guideline (version 1.1)". In: *European journal of cancer* 45.2 (2009), pp. 228–247.

[8] E.E.A. Enger. *Concepts in Biology' 2007 Ed.2007 Edition*. Rex Bookstore, Inc. ISBN: 9780071260428. URL: http://books.google.ch/books?id=1E853Gfo7VkC.

[9] Michael P. Fay and Pamela A. Shaw. "Exact and Asymptotic Weighted Logrank Tests for Interval Censored Data: The interval R Package". In: *Journal of Statistical Software* 36.2 (2010), pp. 1–34. URL: http://www.jstatsoft.org/v36/i02/.

[10] Yanqin Feng and Jianguo Sun. "Hazard ratio estimation for two-sample case under interval-censored failure time data". In: *Biometrical Journal* 56.2 (2014), pp. 219–229. ISSN: 1521-4036. DOI: 10.1002/bimj.201200273. URL: http://dx.doi.org/10.1002/bimj.201200273.

[11] Dianne M Finkelstein. "A proportional hazards model for interval-censored failure time data". In: *Biometrics* (1986), pp. 845–854.

[12] R. Gentleman and Alain Vandal. *Icens: NPMLE for Censored and Truncated Data*. R package version 1.36.0. 2014.

[13] David E Gerber. "Targeted therapies: a new generation of cancer treatments". In: *Am Fam Physician* 77.3 (2008), pp. 311–319.

[14] William B Goggins et al. "A Markov chain Monte Carlo EM algorithm for analyzing interval-censored data under the Cox proportional hazards model". In: *Biometrics* (1998), pp. 1498–1507.

[15] *Guidance for Industry: Clinical trial endpoints for the approval of cancer drugs and biologics*. Food and Drug Administration. USA, 2007. URL: `http://www.fda.gov/downloads/Drugs/GuidanceComplianceRegulatoyInformation/Guidance/UCM071590.pdf`.

[16] Jaroslaw Harezlak and Wanzhu Tu. "Estimation of survival functions in interval and right censored data using STD behavioural diaries". In: *Statistics in medicine* 25.23 (2006), pp. 4053–4064.

[17] Leonhard Held and Daniel Sabanés Bové. *Applied Statistical Inference*. Springer, 2014.

[18] Volkmar Henschel, Christiane Heiß, and Ulrich Mansmann. *intcox: Compendium to apply the iterative convex minorant algorithm to interval censored event data*. 2009.

[19] John D Kalbfleisch and Ross L Prentice. *The statistical analysis of failure time data*. 2nd ed. John Wiley & Sons, 2002.

[20] Edward L Kaplan and Paul Meier. "Nonparametric estimation from incomplete observations". In: *Journal of the American statistical association* 53.282 (1958), pp. 457–481.

[21] John P Klein and Melvin L Moeschberger. *Survival Analysis: Techniques for Censored and Truncated Data*. Springer, 2003.

[22] Ralf Küppers. "Mechanisms of B-cell lymphoma pathogenesis". In: *Nature Reviews Cancer* 5.4 (2005), pp. 251–262.

[23] Mary L Lesperance and John D Kalbfleisch. "An algorithm for computing the nonparametric MLE of a mixing distribution". In: *Journal of the American Statistical Association* 87.417 (1992), pp. 120–126.

[24] Wenbin Lu and Yu Liang. "Empirical likelihood inference for linear transformation models". In: *Journal of multivariate analysis* 97.7 (2006), pp. 1586–1599.

[25] Marloes Maathuis. *MLEcens: Computation of the MLE for bivariate (interval) censored data*. R package version 0.1-4. 2013. URL: `http://CRAN.R-project.org/package=MLEcens`.

[26] Robert Marcus et al. "CVP chemotherapy plus rituximab compared with CVP as first-line treatment for advanced follicular lymphoma". In: *Blood* 105.4 (2005), pp. 1417–1423.

[27] Wei Pan. "Extending the iterative convex minorant algorithm to the Cox model for interval-censored data". In: *Journal of Computational and Graphical Statistics* 8.1 (1999), pp. 109–120.

[28] Wei Pan. “A Multiple Imputation Approach to Cox Regression with Interval-Censored Data”. In: *Biometrics* 56.1 (2000), pp. 199–203.

[29] R Development Core Team. *R: A Language and Environment for Statistical Computing*. ISBN 3-900051-07-0. R Foundation for Statistical Computing. Vienna, Austria, 2009. URL: `http://www.R-project.org`.

[30] David A Schoenfeld. “Sample-size formula for the proportional-hazards regression model”. In: *Biometrics* (1983), pp. 499–503.

[31] AM Stone et al. “Research outcomes and recommendations for the assessment of progression in cancer clinical trials from a PhRMA working group”. In: *European Journal of Cancer* 47.12 (2011), pp. 1763–1771.

[32] Winfried Stute. “The bias of Kaplan-Meier integrals”. In: *Scandinavian Journal of Statistics* (1994), pp. 475–484.

[33] Jianguo Sun. “A non-parametric test for interval-censored failure time data with application to AIDS studies”. In: *Statistics in Medicine* 15.13 (1996), pp. 1387–1395.

[34] Jianguo Sun. *The statistical analysis of interval-censored failure time data*. Vol. 2. Springer, 2006.

[35] Xing Sun and Cong Chen. “Comparison of Finkelsteinâs method with the conventional approach for interval-censored data analysis”. In: *Statistics in Biopharmaceutical Research* 2.1 (2010), pp. 97–108.

[36] Xing Sun et al. “A Review of Statistical Issues with Progression-Free Survival as an Interval-Censored Time-to-Event Endpoint”. In: *Journal of Biopharmaceutical Statistics* 23.5 (2013), pp. 986–1003.

[37] Terry M Therneau. *Modeling survival data: extending the Cox model*. Springer, 2000.

[38] Terry M Therneau and Thomas Lumley. *Package 'survival'*. 2014.

[39] Bruce W Turnbull. “The empirical distribution function with arbitrarily grouped, censored and truncated data”. In: *Journal of the Royal Statistical Society. Series B (Methodological)* (1976), pp. 290–295.

[40] Chaitra Ujjani and Bruce D Cheson. “The optimal management of follicular lymphoma: an evolving field”. In: *Drugs* 73.13 (2013), pp. 1395–1403.

[41] Yi Wang. “Interval-censored Analysis of Progression-free Survival”. In: Genentech Summer Intern Program. 2011.

[42] Jon A Wellner and Yihui Zhan. “A hybrid algorithm for computation of the nonparametric maximum likelihood estimator from censored data”. In: *Journal of the American Statistical Association* 92.439 (1997), pp. 945–959.

[43] Chien-Fu Wu. “Some algorithmic aspects of the theory of optimal designs”. In: *The Annals of Statistics* (1978), pp. 1286–1301.

R version and packages used to generate this report:
R version: 3.1.1 (2014-07-10)
Base packages: `grid, splines, stats, graphics, grDevices, utils, datasets, methods, base`
Other packages: `scales, fields, maps, spam, MLEcens, survival, xtable`
This document was generated on Oktober 31, 2014 at 13:19.

Chapter 8

Appendix

8.1 Derivation of Kaplan-Meier estimator

The following document was downloaded via this link:
http://www.ics.uci.edu/∼vqnguyen/stat255/KM-Derivation.pdf
This is the website of Vinh Q. Nguyen, lecturer of the course Stat255 in Survival Analysis at University of California, Irvine.

Survival Analysis
Statistics 255, Fall 2012

Derivation of the Kaplan-Meier estimator as a nonparametric MLE

Let $t_1 < t_2 < \cdots < t_D$ be the observed failure times in a sample of size n from a population with survival function S. Suppose that d_j observations failed at time t_j and m_j observations were right-censored in the interval $[t_j, t_{j+1})$ at times $t_{j1}, \ldots, t_{jm_j}$, $j = 0, \ldots, D$, where $t_0 = 0$ and $t_{D+1} = \infty$. Let

$$n_j = (m_j + d_j) + \cdots + (m_D + d_D) = \sum_{l=j}^{D} (m_l + d_l)$$

denote the number of observations at risk just prior to time t_j.

The contribution to the likelihood for an observation that failed at time t_j is

$$\Pr[T = t_j] = f(t_j) = -\frac{dS(t)}{dt}\Big|_{t=t_j}.$$

Under the assumption of independence between the censoring time and failure time, the contribution to the likelihood for a censored observation at time t_{jl} is

$$\Pr[T > t_{jl}] = S(t_{jl}).$$

The probability of the observed data is thus

$$\begin{aligned} L &= \prod_{j=0}^{D} \left\{ \Pr[\text{fail at time } t_j]^{d_j} \prod_{l=1}^{m_l} \Pr[\text{censored at time } t_{jl}] \right\} \\ &= \prod_{j=0}^{D} \left\{ \left[-\frac{dS(t)}{dt}\Big|_{t=t_j} \right]^{d_j} \prod_{l=1}^{m_l} S(t_{jl}) \right\}. \end{aligned}$$

If we consider maximizing L with respect to S such that S only assigns positive probability to the observed failure times, then the maximizer S will be a discrete survival function and is discontinuous at $t_1 < t_2 < \cdots < t_D$. This implies $S(t_j^*) = S(t_j)$ for $t_j^* \in [t_j, t_{j+1})$. Since, $t_{jl} \in [t_j, t_{j+1})$, we have $S(t_{jl}) = S(t_j)$. Further, $-\frac{dS(t)}{dt}\big|_{t=t_j}$ can be written as

$$S(t_{j-1}) - S(t_j) = \Pr[T = t_j],$$

where Pr is the probability mass function (pmf) corresponding to the maximizer S. Thus, the likelihood as a function of the discrete survival function S is

$$L(S) = \prod_{j=0}^{D} \left\{ [S(t_{j-1}) - S(t_j)]^{d_j} \prod_{l=1}^{m_l} S(t_j) \right\}$$

Now let

$$\lambda_j = \Pr[T = t_j | T > t_{j-1}].$$

For $j = 1, \ldots, D$, we have

$$\Pr[T = t_j] = \Pr[T = t_j | T > t_{j-1}] \times \Pr[T > t_{j-1} | T > t_{j-2}] \times$$

$$
\begin{aligned}
&\cdots \times \Pr[T > t_2 | T > t_1] \times \Pr[T > t_1 | T > t_0] \\
&= \Pr[T = t_j | T > t_{j-1}] \prod_{l=1}^{j-1} \Pr[T > t_l | T > t_{l-1}] \\
&= \Pr[T = t_j | T > t_{j-1}] \prod_{l=1}^{j-1} (1 - \Pr[T = t_l | T > t_{l-1}]) \\
&= \lambda_j \prod_{l=1}^{j-1} (1 - \lambda_l),
\end{aligned}
$$

and

$$
\begin{aligned}
\Pr[T > t_j] &= \Pr[T > t_j | T > t_{j-1}] \times \Pr[T > t_{j-1} | T > t_{j-2}] \times \\
&\quad \cdots \times \Pr[T > t_2 | T > t_1] \times \Pr[T > t_1 | T > t_0] \\
&= \prod_{l=1}^{j} \Pr[T > t_l | T > t_{l-1}] \\
&= \prod_{l=1}^{j} (1 - \Pr[T = t_l | T > t_{l-1}]) \\
&= \prod_{l=1}^{j} (1 - \lambda_l),
\end{aligned}
$$

Thus, the likelihood can be reparameterized as a function of $\vec{\lambda} = (\lambda_1, \ldots, \lambda_D)^T$:

$$
\begin{aligned}
L(\vec{\lambda}) &= \prod_{j=1}^{D} \left\{ \lambda_j^{d_j} \left[\prod_{l=1}^{j-1} (1 - \lambda_l)^{d_j} \right] \left[\prod_{l=1}^{j} (1 - \lambda_l)^{m_j} \right] \right\} \\
&= \prod_{j=1}^{D} \left\{ \frac{\lambda_j^{d_j}}{(1 - \lambda_j)^{d_j}} \left[\prod_{l=1}^{j} (1 - \lambda_l)^{d_j} \right] \left[\prod_{l=1}^{j} (1 - \lambda_l)^{m_j} \right] \right\} \\
&= \prod_{j=1}^{D} \left\{ \frac{\lambda_j^{d_j}}{(1 - \lambda_j)^{d_j}} \left[\prod_{l=1}^{j} (1 - \lambda_l)^{d_j + m_j} \right] \right\} \\
&= \left[\prod_{j=1}^{D} \frac{\lambda_j^{d_j}}{(1 - \lambda_j)^{d_j}} \right] \left[\prod_{j=1}^{D} \prod_{l=1}^{j} (1 - \lambda_l)^{d_j + m_j} \right] \\
&= \left[\prod_{j=1}^{D} \frac{\lambda_j^{d_j}}{(1 - \lambda_j)^{d_j}} \right] (1 - \lambda_1)^{d_1 + m_1} (1 - \lambda_1)^{d_2 + d_2} (1 - \lambda_2)^{d_2 + m_2} \times \\
&\quad (1 - \lambda_1)^{d_3 + d_3} (1 - \lambda_2)^{d_3 + m_3} (1 - \lambda_3)^{d_3 + m_3} \times \cdots \\
&= \left[\prod_{j=1}^{D} \frac{\lambda_j^{d_j}}{(1 - \lambda_j)^{d_j}} \right] \left[\prod_{j=1}^{D} (1 - \lambda_j)^{\sum_{l=j}^{D} m_l + d_l} \right] \\
&= \left[\prod_{j=1}^{D} \frac{\lambda_j^{d_j}}{(1 - \lambda_j)^{d_j}} \right] \left[\prod_{j=1}^{D} (1 - \lambda_j)^{n_j} \right] \\
&= \prod_{j=1}^{D} \left[\lambda_j^{d_j} (1 - \lambda_j)^{n_j - d_j} \right]
\end{aligned}
$$

To this end,

$$\begin{aligned} l(\vec{\lambda}) &= \log L(\vec{\lambda}) \\ &= \sum_{j=1}^{D} d_j \log \lambda_j + (n_j - d_j)\log(1-\lambda_j) \\ U_j(\vec{\lambda}) &= \frac{\partial l((\vec{\lambda}))}{\partial \lambda_j} = \frac{d_j}{\lambda_j} - \frac{n_j - d_j}{1-\lambda_j} \end{aligned}$$

and solving for $U_j(\widehat{\vec{\lambda}}) \equiv \vec{0}$ we have $\widehat{\lambda}_j = d_j/n_j$. From this, we have

$$\widehat{S}(t) = \prod_{j:t_j \le t} (1 - \widehat{\lambda}_j) = \prod_{j:t_j \le t} (1 - d_j/n_j) \qquad \text{(the KM estimator).}$$

Now,

$$\begin{aligned} I_{jj}(\vec{\lambda}) &= -\mathbb{E}\left[\frac{\partial U_j(\vec{\lambda})}{\partial \lambda_j}\right] = \mathbb{E}\left[\frac{d_j}{\lambda_j^2} + \frac{n_j - d_j}{(1-\lambda_j)^2}\right] \\ I_{jk}(\vec{\lambda}) &= -\mathbb{E}\left[\frac{\partial U_j(\vec{\lambda})}{\partial \lambda_k}\right] = 0, \end{aligned}$$

and

$$\widehat{I}_{jj}(\vec{\lambda}) = \frac{d_j}{\widehat{\lambda}_j^2} + \frac{n_j - d_j}{(1-\widehat{\lambda}_j)^2} = \cdots = \frac{n_j^3}{d_j(n_j - d_j)}.$$

So, from likelihood theory, we have

$$\widehat{\vec{\lambda}} \dot{\sim} \mathcal{N}\left(\vec{\lambda}, I^{-1}(\vec{\lambda})\right),$$

and by the δ-method,

$$\log(1-\widehat{\lambda}_j) \dot{\sim} \mathcal{N}\left(\log(1-\lambda_j), \frac{1}{(1-\lambda_j)^2} I_{jj}^{-1}(\vec{\lambda})\right),$$

where

$$\widehat{\mathrm{Var}}[\log(1-\widehat{\lambda}_j)] = \frac{\widehat{I}_{jj}^{-1}(\vec{\lambda})}{1-\widehat{\lambda}_j} = \frac{n_j^2}{(n_j-d_j)^2} \times \frac{d_j(n_j-d_j)}{n_j^3} = \frac{d_j}{n_j(n_j-d_j)}$$

From this, $\log \widehat{S}_t = \sum_{j:t_j \le t} \log(1-\widehat{\lambda}_j)$ and

$$\widehat{\mathrm{Var}}[\log \widehat{S}(t)] = \sum_{j:t_j \le t} \widehat{\mathrm{Var}}[\log(1-\widehat{\lambda})] = \sum_{j:t_j \le t} \frac{d_j}{n_j(n_j-d_j)},$$

and again, by the δ-method,

$$\widehat{\mathrm{Var}}[\widehat{S}(t)] = \widehat{S}(t)^2 \sum_{j:t_j \le t} \frac{d_j}{n_j(n_j-d_j)},$$

yielding Greenwood's formula.

8.2 Functions defined for simulation and estimation

This section contains the functions that were defined and used to simulate the datasets and to do the estimations in this thesis.

8.2.1 Simulation: `WeibPFSSim1` and `WeibPFSSim2` functions

The functions `WeibPFSSim1` and `WeibPFSSim2` are used in a sequential manner. With `WeibPFSSim1` data of every study arm is simulated, one at a time. `WeibPFSSim2` is used to combine the study arms and to rightcensor all events after the analysis cutoff. See Chapter 3 for details about parameters.

`WeibPFSSim1`

```
> ## ============================================================================
> ## this script contains a function
> ## that simulates a dataset of a study arm for a PFS endpoint
> ## based on a Weibull parametric survival function
> ## applying right- and interval-censoring
> ## resulting in exact, right-censored and interval-censored observations
> ## without censoring at cutoff
> ## ============================================================================
>
>
> WeibPFSSim1 <- function (scenario, n, shape, scale, deathprob, censscale,
                           assess.vec, sd, M, seed = NA) {

   # ============================================================================
   # with simulation parameters:
   # simulate realistic dataset that contains:
   # - death events
   # - PFS events
   # - dropouts
   # and save in list

   # Args:
   ## 0. Scenario number
   # scenario = scenario number (for easy recognition)

   ## 1. exact event times: Weibull distribution - rweibull(n, shape, scale)
   # n = number of observations
   # shape = alpha
   # scale = mu or beta, = 1 / lambda

   ## 2. death events: exact event time by binomial distribution - rbinom(n, deathprob)
   # deathprob = probability of event being death

   ## 3. right-censoring by early drop out: by exponential distribution - rweib(n, 1, censlambda)
   # censscale = scale (1 / lambda) used for exponential distribution of right-censoring times

   ## 4. interval-censoring of PFS events: define assessment schedule
   # assess.vec = vector of assessment days
   # sd = standard deviation of normal rv generation for all widths -
   #     eg. rnorm(n, assess[2] - assess[1], sd) (= variation of widths)

   ## 5. number of simulations
   # M = simulations with the same dataset

   ## 6. seed for simulation (to reproduce same data)
   # seed = number corresponding to starting point of random variable generator
   #        (default is NA, what means no starting point is set)

   # Returns:
   # list of
   # $sim.data: list of M dataframes with 5 columns
   #        (event.time, rightcens.time, rightcens.ind, left.time, right.time)
   #        representing (uncensored, right-censored, interval-censored) data
   # $sim.par: vector of simulation parameters
```

```
  # $assess.par: vector of assessment times inserted as parameters
  # $assess.times: list of M matrices with n rows and length(assess.vec) columns
  #                representing all assessment times
  # ============================================================================

# M simulations:
sim.data <- list(NA)
assess.times <- list(NA)

## ============================================================================
## set seed
## ============================================================================
if (!is.na(seed)) {set.seed(seed)}

for (j in 1:M) {

    # setup result vectors
    ## ========================================================================
    event.time <- rep(NA, times = n)
    rightcens.time <- rep(NA, times = n)
    rightcens.ind <- rep(NA, times = n)
    left.time <- rep(NA, times = n)
    right.time <- rep(NA, times = n)

    # and simulation indicator
    sim.ind <- rep(NA, times = n)

    # 1. simulate events
    ## ========================================================================
    event.time <- rweibull(n, shape, scale)


    # 2. simulate death events: exact time known
    ## ========================================================================
    #   by binomial distribution
      sim.ind <- rbinom(n, 1, prob = deathprob)

    for (i in 1:n) {
      if (sim.ind[i] == 1) {
        rightcens.time[i] <- event.time[i]
        rightcens.ind[i]  <- 1
        left.time[i]      <- event.time[i]
        right.time[i]     <- event.time[i] }
    }

    # 3. simulate right-censoring by early drop out
    ## ========================================================================
    # find censoring by comparison of time value modeled by exponential distribution
    cens.time <- rweibull(n, 1, censscale)

    for (i in 1:n) {
      if (cens.time[i] <= event.time[i])  {
          rightcens.time[i] <- cens.time[i]
          rightcens.ind[i] <- 0
          left.time[i] <- cens.time[i]
          right.time[i] <- Inf }
    }

    # 4. simulate interval-censoring of PFS events: with assessment schedule
    ## ========================================================================
    # a) calculate width of time intervals
    width <- c(assess.vec[1], diff(assess.vec), Inf)

    #    and vary it by rnorm with sd = sd
    width.jit <- matrix(rep(NA, n * length(width)), nrow = n)
    for (i in 1:n) {width.jit[i, ] <- rnorm(length(width), mean = width, sd = sd) }

    # b) establish jittered assessment schedule
    assess.vec.jit <- apply(width.jit, 1, cumsum)
    assess.vec.jit <- rbind(rep(0, n), assess.vec.jit)
```

```
    # c) choose appropriate timepoints for patients not considered in 2. and 3.
  for (i in 1:n) {
    if (is.na(rightcens.ind[i])) {
      left.time[i] <- max(assess.vec.jit[, i][assess.vec.jit[, i] < event.time[i]])
      right.time[i] <- min(assess.vec.jit[, i][assess.vec.jit[, i] > event.time[i]])
      rightcens.ind[i] <- 1
      rightcens.time[i] <- right.time[i] }
  }

  # d) set patients with progression event after last assessment to right-censored at last assessment
  for (i in 1:n) {
    if (right.time[i] == Inf & rightcens.ind[i] == 1) {
      left.time[i] <- assess.vec.jit[, i][nrow(assess.vec.jit) - 1]
      right.time[i] <- Inf
      rightcens.ind[i] <- 0
      rightcens.time[i] <- left.time[i] }
  }

  # e) set patients with death event after last assessment to right-censored at last assessment
  for (i in 1:n) {
    if (right.time[i] == left.time[i] & right.time[i] > assess.vec.jit[, i][nrow(assess.vec.jit) - 1]) {
      left.time[i] <- assess.vec.jit[, i][nrow(assess.vec.jit) - 1]
      right.time[i] <- Inf
      rightcens.ind[i] <- 0
      rightcens.time[i] <- left.time[i] }
  }


  ## ============================================================================
  ## round all generated time values
  ## ============================================================================
      event.time <- round(event.time)
  rightcens.time <- round(rightcens.time)
       left.time <- round(left.time)
      right.time <- round(right.time)
  assess.vec.jit <- round(assess.vec.jit)

  ## ============================================================================
  ## avoid zeros, they would result in survreg-errors
  ## ============================================================================
      event.time <- ifelse(event.time == 0, 1, event.time)
  rightcens.time <- ifelse(rightcens.time == 0, 1, rightcens.time)
       left.time <- ifelse(left.time == 0, 1, left.time)
      right.time <- ifelse(right.time == 0, 1, right.time)

  ## ============================================================================
  ## generate data frame
  ## ============================================================================
  parsim <- data.frame(    event.time = event.time,
                       rightcens.time = rightcens.time,
                        rightcens.ind = rightcens.ind,
                            left.time = left.time,
                           right.time = right.time)


  ## ============================================================================
  ## put together list of data frames sim.data and list of matrices assess.times
  ## ============================================================================
  sim.data[[j]] <- parsim
  assess.times[[j]] <- assess.vec.jit[-c(1, nrow(assess.vec.jit)), ]

}

## ============================================================================
## generate vectors sim.par and assess.par
## ============================================================================

options(scipen = 4)
sim.par <- c(scenario, n, shape, scale, deathprob, censscale,
```

```
              sd, M, seed)
 names(sim.par) <- c("scenario", "n", "shape", "scale", "deathprob", "censscale",
                     "sd", "M", "seed")
 assess.par <- assess.vec

 ## ============================================================================
 ## output list of data and parameters
 ## ============================================================================

 list(sim.data = sim.data, sim.par = sim.par, assess.par = assess.par,
      assess.times = assess.times)

 }
>
```

WeibPFSSim2

```
> ## ============================================================================
> ## this script contains a function
> ## that simulates the censoring at analysis cutoff
> ## of PFS datasets of one to four study arms.
> ## The datasets contain uncensored, right-censored and interval-censored data
> ## ============================================================================
>
>
> WeibPFSSim2 <- function (WeibPFSSim1.data1,
                           WeibPFSSim1.data2 = NULL,
                           WeibPFSSim1.data3 = NULL,
                           WeibPFSSim1.data4 = NULL,
                           recr1, recr2 = NULL, recr3 = NULL, recr4 = NULL,
                           cutoff, arm, seed = NA) {

   # ============================================================================
   # with simulation parameters:
   # simulate with 1-4 realistic datasets:
   # - analysis cutoff
   # and save datasets in list

   # Args:
   # WeibPFSSim1.data1 = dataset to apply the cutoff on,
   #                     for description of structure see WeibPFSSim-function
   # WeibPFSSim1.data2, 3, 4 = optional additional datasets (other study arms),
   #                           added to apply an overall cutoff
   #                           these datasets need to have the same number of simulations
   #                           in order to be able to
   ##  right-censoring at clinical cutoff: (= administrative censoring)
   # recr1 = number of patients recruited per month
   # recr2, 3, 4 = for other datasets: number of patients recruited per month
   # cutoff = number of patients with event before cutoff (overall)
   # arm = number of study arms
   # seed = number corresponding to starting point of random variable generator
   #        (default is NA, what means no starting point is set)


   # Returns:
   # list of
   # $sim.data: list of M dataframes of all study arms with 6 columns
   #            (event.time, rightcens.time, rightcens.ind, left.time, right.time, treatment)
   #            representing (uncensored, right-censored, interval-censored) data
   #            after right-censoring at analysis cutoff
   ##           and treatment arm as a number from 0 to 3 (according to arms 1 to 4).
   # $assess.times: list of M matrices with n rows and length(assess.vec) columns
   #                representing all assessment times of all study arms
   # $arm1: list of data of first arm of study, with these objects:
   #        - $sim.data: list of M dataframes with 5 columns
   #        - $sim.par: vector of simulation parameters
   #        - $assess.par: vector of assessment times inserted as parameters
   # $arm2, 3, 4: optional, list of data of other study arms,
   #              with the same objects as arm1
```

```
  # ============================================================================

## ============================================================================
# check if there are not more than 4 study arms:
## ============================================================================
if (length(arm) > 4) {warning("This function can only handle up to 4 study arms.
                      You indicated usage of more arms by defining 'arm' > 4.")}

## ============================================================================
# check if M is the same for all study arms:
## ============================================================================
stopifnot(WeibPFSSim1.data1$sim.par[["M"]] == WeibPFSSim1.data2$sim.par[["M"]] &
          WeibPFSSim1.data1$sim.par[["M"]] == WeibPFSSim1.data3$sim.par[["M"]] &
          WeibPFSSim1.data1$sim.par[["M"]] == WeibPFSSim1.data4$sim.par[["M"]])

M <- WeibPFSSim1.data1$sim.par[["M"]]

## ============================================================================
## introduce lists for saving
## ============================================================================
sim.data <- list(NA)
assess.times <- list(NA)
arm1 <- list(NA)
arm2 <- list(NA)
arm3 <- list(NA)
arm4 <- list(NA)
sim.data1 <- list(NA)
sim.data2 <- list(NA)
sim.data3 <- list(NA)
sim.data4 <- list(NA)

## ============================================================================
## set seed
## ============================================================================
if (!is.na(seed)) {set.seed(seed)}

for (j in 1:M) {

    ## ========================================================================
    # read in data vectors
    ## ========================================================================
    event.time <- c(WeibPFSSim1.data1$sim.data[[j]]$event.time,
                    WeibPFSSim1.data2$sim.data[[j]]$event.time,
                    WeibPFSSim1.data3$sim.data[[j]]$event.time,
                    WeibPFSSim1.data4$sim.data[[j]]$event.time)
    rightcens.time <- c(WeibPFSSim1.data1$sim.data[[j]]$rightcens.time,
                        WeibPFSSim1.data2$sim.data[[j]]$rightcens.time,
                        WeibPFSSim1.data3$sim.data[[j]]$rightcens.time,
                        WeibPFSSim1.data4$sim.data[[j]]$rightcens.time)
    rightcens.ind <- c(WeibPFSSim1.data1$sim.data[[j]]$rightcens.ind,
                       WeibPFSSim1.data2$sim.data[[j]]$rightcens.ind,
                       WeibPFSSim1.data3$sim.data[[j]]$rightcens.ind,
                       WeibPFSSim1.data4$sim.data[[j]]$rightcens.ind)
    left.time <- c(WeibPFSSim1.data1$sim.data[[j]]$left.time,
                   WeibPFSSim1.data2$sim.data[[j]]$left.time,
                   WeibPFSSim1.data3$sim.data[[j]]$left.time,
                   WeibPFSSim1.data4$sim.data[[j]]$left.time)
    right.time <- c(WeibPFSSim1.data1$sim.data[[j]]$right.time,
                    WeibPFSSim1.data2$sim.data[[j]]$right.time,
                    WeibPFSSim1.data3$sim.data[[j]]$right.time,
                    WeibPFSSim1.data4$sim.data[[j]]$right.time)
    n1 <- WeibPFSSim1.data1$sim.par[["n"]]
    n2 <- WeibPFSSim1.data2$sim.par[["n"]]
    n3 <- WeibPFSSim1.data3$sim.par[["n"]]
    n4 <- WeibPFSSim1.data4$sim.par[["n"]]
    n <- length(right.time)
    assess.vec.jit1 <- rbind(rep(0, n1), WeibPFSSim1.data1$assess.times[[j]], rep(Inf, n1))

    if (length(n2) != 0) {
      assess.vec.jit2 <- rbind(rep(0, n2), WeibPFSSim1.data2$assess.times[[j]], rep(Inf, n2))
```

```
    } else { assess.vec.jit2 <- NULL}
    if (length(n3) != 0) {
      assess.vec.jit3 <- rbind(rep(0, n3), WeibPFSSim1.data3$assess.times[[j]], rep(Inf, n3))
    } else { assess.vec.jit3 <- NULL}
    if (length(n4) != 0) {
      assess.vec.jit4 <- rbind(rep(0, n4), WeibPFSSim1.data4$assess.times[[j]], rep(Inf, n4))
    } else { assess.vec.jit4 <- NULL}

    assess.vec.jit <- cbind(assess.vec.jit1, assess.vec.jit2, assess.vec.jit3, assess.vec.jit4)



## ============================================================================
# for first arm dataset
## ============================================================================

    # a) define arrival times (recr per month (30 days))
    arrival.time <- NULL
    for (i in 1:ceiling(n1 / recr1)) {
      arrival.time <- c(arrival.time, runif(recr1, min = (i - 1) * 30, max = i * 30))}
    arrival.time1 <- arrival.time[1:n1]

    # b) calculate total time
    total.time1 <- arrival.time1 + rightcens.time[1:n1]

## ============================================================================
# for additional datasets
## ============================================================================

   if (length(n2) != 0) {
      # a) define arrival times (recr per month (30 days))
      arrival.time <- NULL
      for (i in 1:ceiling(n2 / recr2)) {
        arrival.time <- c(arrival.time, runif(recr2, min = (i - 1) * 30, max = i * 30))}
      arrival.time2 <- arrival.time[1:n2]

      # b) calculate total time
      total.time2 <- arrival.time2 + rightcens.time[(n1+1):(n1+n2)]

      } else {arrival.time2 <- NULL
              total.time2 <- NULL }

    if (length(n3) != 0) {
      # a) define arrival times (recr per month (30 days))
      arrival.time <- NULL
      for (i in 1:ceiling(n3 / recr3)) {
        arrival.time <- c(arrival.time, runif(recr3, min = (i - 1) * 30, max = i * 30))}
      arrival.time3 <- arrival.time[1:n3]

      # b) calculate total time
      total.time3 <- arrival.time3 + rightcens.time[(n1+n2+1):(n1+n2+n3)]

      } else {arrival.time3 <- NULL
              total.time3 <- NULL }

    if (length(n4) != 0) {
      # a) define arrival times (recr per month (30 days))
      arrival.time <- NULL
      for (i in 1:ceiling(n4 / recr4)) {
        arrival.time <- c(arrival.time, runif(recr4, min = (i - 1) * 30, max = i * 30))}
      arrival.time4 <- arrival.time[1:n4]

      # b) calculate total time
      total.time4 <- arrival.time4 + rightcens.time[(n1+n2+n3+1):(n1+n2+n3+n4)]

      } else {arrival.time4 <- NULL
              total.time4 <- NULL }

    arrival.time <- c(arrival.time1, arrival.time2, arrival.time3, arrival.time4)
    total.time <- c(total.time1, total.time2, total.time3, total.time4)
```

```
## ============================================================================
# find cutoff time from cutoff patient numbers
## ============================================================================

    # c) find cutoff time from cutoff n
    cutoff.n <- cutoff
    total.time.events <- total.time[rightcens.ind == 1]
    if (cutoff.n >= length(total.time.events)) {
      warning("the maximum of events is smaller than the events requested for cutoff")
      cutoff.n <- length(total.time.events)
    }
    cut.time <- sort(total.time.events)[cutoff.n] + 0.1

    # d) apply censoring
    sim.ind <- rep(NA, times = n)
    rest.time <- rep(NA, times = n)

    for (i in 1:n) {
      if (total.time[i] >= cut.time)  {
        sim.ind[i] <- 0
        rest.time[i] <- cut.time - arrival.time[i]
        # differentiate positive and negative rest.times
        # (negative means: the patient was censored before arrival)
        # do not differentiate between right-censoring and event rest.times
        # because the right-censoring (by dropout) is only decided at the timepoint of censoring
        if (rest.time[i] >= 0) {
          left.time[i] <- max(assess.vec.jit[, i][assess.vec.jit[, i] < rest.time[i]])
        } else {
          left.time[i] <- 0 }

        right.time[i] <- Inf
        rightcens.ind[i] <- 0
        rightcens.time[i] <- left.time[i] }
    }


    ## ============================================================================
    ## round all generated time values
    ## ============================================================================
        event.time <- round(event.time)
    rightcens.time <- round(rightcens.time)
         left.time <- round(left.time)
        right.time <- round(right.time)
    assess.vec.jit <- round(assess.vec.jit)

    ## ============================================================================
    ## avoid zeros, they would result in survreg-errors
    ## ============================================================================
        event.time <- ifelse(event.time == 0, 1, event.time)
    rightcens.time <- ifelse(rightcens.time == 0, 1, rightcens.time)
         left.time <- ifelse(left.time == 0, 1, left.time)
        right.time <- ifelse(right.time == 0, 1, right.time)

    ## ============================================================================
    ## build treatment vector
    ## ============================================================================
    if (arm == 2)
      {treatment <- c(rep(0, n1), rep(1, n2))}
    if (arm == 3)
      {treatment <- c(rep(0, n1), rep(1, n2), rep(2, n3))}
    if (arm == 4)
      {treatment <- c(rep(0, n1), rep(1, n2), rep(2, n3), rep(3, n4))}

    ## ============================================================================
    ## generate data frame
    ## ============================================================================
    if (arm == 1) {
    parsim <- data.frame(    event.time = event.time,
```

```
                        rightcens.time = rightcens.time,
                         rightcens.ind = rightcens.ind,
                             left.time = left.time,
                            right.time = right.time)} else {
    parsim <- data.frame(    event.time = event.time,
                        rightcens.time = rightcens.time,
                         rightcens.ind = rightcens.ind,
                             left.time = left.time,
                            right.time = right.time,
                             treatment = treatment)}

    ## ============================================================================
    ## put together list of data frames sim.data and list of matrices assess.times
    ## ============================================================================
    sim.data[[j]] <- parsim
    sim.data1[[j]] <- parsim[1:n1, c(1:5)]

if (length(n2) != 0) {sim.data2[[j]] <- parsim[(n1+1):(n1+n2), c(1:5)]
                      } else {sim.data2[[j]] <- NULL }
if (length(n3) != 0) {sim.data3[[j]] <- parsim[(n1+n2+1):(n1+n2+n3), c(1:5)]
                      } else {sim.data3[[j]] <- NULL }
if (length(n4) != 0) {sim.data4[[j]] <- parsim[(n1+n2+n3+1):(n1+n2+n3+n4), c(1:5)]
                      } else {sim.data4[[j]] <- NULL }

    assess.times[[j]] <- assess.vec.jit[-c(1, nrow(assess.vec.jit)), ]
}

## ============================================================================
## generate vectors sim.par and assess.par
## ============================================================================
sim.par1 <- c(WeibPFSSim1.data1$sim.par, recr1, cutoff, arm, seed)
names(sim.par1) <- c("scenario", "n", "shape", "scale", "deathprob", "censscale",
                     "sd", "M", "seed", "recr", "cutoff", "arm", "seed2")
assess.par1 <- WeibPFSSim1.data1$assess.par

if (length(n2) != 0) {
  sim.par2 <- c(WeibPFSSim1.data2$sim.par, recr2, cutoff, arm, seed)
  names(sim.par2) <- c("scenario", "n", "shape", "scale", "deathprob", "censscale",
                       "sd", "M", "seed", "recr", "cutoff", "arm", "seed2")
} else {sim.par2 <- NULL}
assess.par2 <- WeibPFSSim1.data2$assess.par

if (length(n3) != 0) {
  sim.par3 <- c(WeibPFSSim1.data3$sim.par, recr3, cutoff, arm, seed)
  names(sim.par3) <- c("scenario", "n", "shape", "scale", "deathprob", "censscale",
                       "sd", "M", "seed", "recr", "cutoff", "arm", "seed2")
} else {sim.par3 <- NULL}
assess.par3 <- WeibPFSSim1.data3$assess.par

if (length(n4) != 0) {
  sim.par4 <- c(WeibPFSSim1.data4$sim.par, recr4, cutoff, arm, seed)
  names(sim.par4) <- c("scenario", "n", "shape", "scale", "deathprob", "censscale",
                       "sd", "M", "seed", "recr", "cutoff", "arm", "seed2")
} else {sim.par4 <- NULL}
assess.par4 <- WeibPFSSim1.data4$assess.par

## ============================================================================
## output list of data and parameters
## ============================================================================

list(sim.data = sim.data, assess.times = assess.times,
     arm1 = list(sim.data = sim.data1, sim.par = sim.par1, assess.par = assess.par1),
     arm2 = list(sim.data = sim.data2, sim.par = sim.par2, assess.par = assess.par2),
     arm3 = list(sim.data = sim.data3, sim.par = sim.par3, assess.par = assess.par3),
     arm4 = list(sim.data = sim.data4, sim.par = sim.par4, assess.par = assess.par4))

}
>
```

8.2.2 Estimation of survival quantiles and times: TimeStep..., TimeWeib..., SurvStep..., SurvWeib... functions

Four types of functions are defined:

TimeStep: to find survival times at certain survival quantiles, by nonparametric estimation of the survival function with the NPMLE

TimeWeib: to find survival times at certain survival quantiles, by parametric estimation if the survival function with the Weibull function

SurvStep: to find survival quantiles at certain survival times, by nonparametric estimation of the survival function with the NPMLE

SurvWeib: to find survival quantiles at certain survival times, by parametric estimation if the survival function with the Weibull function

For all these types of function, three versions are defined, for **uncensored** (complete), **rightcensored** and **intervalcensored** data. For uncensored data the rightcensoring methods are used.

`TimeStepUncens`

```
> ## ============================================================================
> ## this script contains a function
> ## that for specific survival probabilities (surv.grid) estimates the survival timepoints
> ## corresponding to looking at the quantile function of the nonparametric (Kaplan-Meier or Turnbull) distribution
> ## ============================================================================
>
>
> TimeStepUncens <- function (surv.grid, event.time, CI = FALSE) {

 # ============================================================================
 # with uncensored data:
 # calculate nonparametric (Kaplan-Meier) estimates of inverse survival function
 # and their confidence interval boundaries at specific survival prob. surv.grid
 # and save in vectors

 # Args:
 # surv.grid: vector of survival probabilities that should be evaluated
 # event.time: vector of event times of one dataset
 #             all events have to be observed, no censoring allowed
 # CI: logical value indicating if 95% confidence interval boundaries are added
 #     to the result and result is output in a list instead of a vector

 # Returns:
 # if confint = FALSE :
 # vector of time estimates for each survival probability in surv.grid
 # if confint = TRUE :
 # list of three vectors: uncens, uncens.CI.low, uncens.CI.up:
 # vectors of time estimates and their 95% confidence interval boundaries
 # for each survival probability in surv.grid
 # ============================================================================

 # initialize result vectors
 uncens        <- rep(NA, length(surv.grid))
 uncens.CI.low <- rep(NA, length(surv.grid))
 uncens.CI.up  <- rep(NA, length(surv.grid))

 # calculate estimate
 survfit.est <- survfit(Surv(event.time, rep(1, length(event.time))) ~ 1,
                        conf.type = "log-log")
```

```
for (i in 1:length(surv.grid)) {
uncens[i]        <- quantile(survfit.est, probs = 1 - surv.grid[i])$quantile[[1]]
uncens.CI.low[i] <- quantile(survfit.est, probs = 1 - surv.grid[i])$lower[[1]]
uncens.CI.up[i]  <- quantile(survfit.est, probs = 1 - surv.grid[i])$upper[[1]]
}

# decide between the two output versions
if (CI == TRUE) {output <- list(uncens, uncens.CI.low, uncens.CI.up)
                 names(output) <- c("uncens", "uncens.CI.low", "uncens.CI.up")
                       } else {output <- uncens}
output
}
```

TimeStepRightcens

```
> ## ============================================================================
> ## this script contains a function
> ## that for specific survival probabilities (surv.grid) estimates the survival timepoints
> ## corresponding to looking at the quantile function of the nonparametric (Kaplan-Meier or Turnbull) function
> ## ============================================================================
>
>
> TimeStepRightcens <- function (surv.grid, rightcens.time, rightcens.ind, CI = FALSE) {

   # ============================================================================
   # with rightcensored data:
   # calculate nonparametric (Kaplan-Meier) estimates of inverse survival function
   # and their confidence interval boundaries at specific survival prob. surv.grid
   # and save in vectors

   # Args:
   # surv.grid: vector of survival probabilities that should be evaluated
   # rightcens.time: vector of event and right-censoring times of one dataset
   # rightcens.ind: vector of censoring indicator (1 = event time, 0 = censoring time)
   # (right-censoring means: event has happened any time after the censoring time)
   # CI: logical value indicating if 95% confidence interval boundaries are added
   #     to the result and result is output in a list instead of a vector

   # Returns:
   # if confint = FALSE :
   # vector of time estimates for each survival probability in surv.grid
   # if confint = TRUE :
   # list of three vectors: uncens, uncens.CI.low, uncens.CI.up:
   # vectors of time estimates and their 95% confidence interval boundaries
   # for each survival probability in surv.grid
   # ============================================================================

   # initialize result vectors
   rightcens        <- rep(NA, length(surv.grid))
   rightcens.CI.low <- rep(NA, length(surv.grid))
   rightcens.CI.up  <- rep(NA, length(surv.grid))


   # calculate estimate
   survfit.est <- survfit(Surv(rightcens.time, rightcens.ind) ~ 1, conf.type = "log-log")

   for (i in 1:length(surv.grid)) {
     rightcens[i]        <- quantile(survfit.est, probs = 1 - surv.grid[i])$quantile[[1]]
     rightcens.CI.low[i] <- quantile(survfit.est, probs = 1 - surv.grid[i])$lower[[1]]
     rightcens.CI.up[i]  <- quantile(survfit.est, probs = 1 - surv.grid[i])$upper[[1]]
   }


   # decide between the two output versions
   if (CI == TRUE) {output <- list(rightcens, rightcens.CI.low, rightcens.CI.up)
                    names(output) <- c("rightcens", "rightcens.CI.low", "rightcens.CI.up")
   } else {output <- rightcens}
   output
 }
```

TimeStepIntervalcens

```
> ## ============================================================================
> ## this script contains a function
> ## that for specific survival probabilities (surv.grid) estimates the survival timepoints
> ## corresponding to looking at the quantile function of the nonparametric (Kaplan-Meier or Turnbull) function
> ## ============================================================================
>
>
> TimeStepIntervalcens <- function (surv.grid, left.time, right.time,
                                    approxim = "linear", tol = 10^-5) {

   # ============================================================================
   # with intervalcensored data:
   # calculate nonparametric (Turnbull) estimates of inverse survival function
   # at specific survival probability points surv.grid and save in vectors
   # In some survival prob. ranges the time ML-estimate is ambigous,
   # it can be any value between lower and upper boundaries.
   # The user can choose the output: either one boundary value or
   # a value assuming linear change between the unique border values

   # Args:
   # surv.grid: vector of survival probabilities that should be evaluated
   # left.time: vector of left boundary of interval-censored times of one dataset
   # right.time: vector of right boundary of interval-censored times of one dataset
   # approxim: define how non-unique estimates should be handled (see below)
   # tol: tolerance,
   # algorithm terminates when difference in MLE between two steps is below tolerance

   # Returns:
   # depending on handling of non-unique estimates:
   # if approxim = "linear" :
   # vector of time estimates with linear change in ambigous areas, followed by
   # the numerical value corresponding to the logical indicator of convergence
   # of the NPMLE algorithm (0 = FALSE, 1 = TRUE)
   # if approxim = "lower" :
   # vector of time estimates at lower boundary followed by the same convergence indicator
   # if approxim = "upper" :
   # vector of time estimates at upper boundary followed by the same convergence indicator
   # if approxim = "all" :
   # list of the three vectors above,
   # named intervalcens.linear, intervalcens.lower, intervalcens.upper
   # ============================================================================

   # check approxim entry
   approxim <- match.arg(approxim, c("lower", "upper", "linear", "all"))

   # initialize result vectors
   intervalcens.lower   <- rep(NA, length(surv.grid))
   intervalcens.upper  <- rep(NA, length(surv.grid))
   intervalcens.linear <- rep(NA, length(surv.grid))

   # save censored values as data, add dummy variable (0,1) as second variable
   # (because needed input format is observation rectangles)
   data <- cbind(left.time = left.time,
                 right.time = right.time,
                 dum.left = rep(0, length(left.time)),
                 dum.right = rep(1, length(left.time)))

   data <- as.matrix(data)

   # define boundaries (0 = open, 1 = closed)
   # (have to be 1,1 for exact values, and are set to 0,1 for intervals, because mathematically easier)
   bound1 <- matrix(rep(NA, times = 2 * length(left.time)), ncol = 2)
   for (i in 1:length(left.time)) {
     if (left.time[i] == right.time[i]) {bound1[i, ] <- c(1, 1)} else {
                                        bound1[i, ] <- c(0, 1)}
   }
   bound2 <- matrix(c(rep(0, times = length(left.time)),
                      rep(1, times = length(left.time))), ncol = 2)
```

```
  bound <- cbind(bound1, bound2)

  # ============================================================================
  # calculate NPMLE
  # ============================================================================
  MLEcens.est <- computeMLE(data, bound, tol = tol)

  # save convergence indicator
  converge <- MLEcens.est$conv

  # extract numbers from output
  step.lower.time <- MLEcens.est$rects[, 1]
  step.upper.time <- MLEcens.est$rects[, 2]
  surv.step <- 1 - cumsum(MLEcens.est$p)

  # and add first line: time = 0, survival = 1
  step.lower.time <- c(0, step.lower.time, Inf)
  step.upper.time <- c(0, step.upper.time, Inf)
  surv.step <- c(1,surv.step, 0)

  # display in a dataframe
  cbind(step.lower.time, step.upper.time, surv.step)

  # ============================================================================
  # calculate estimates for lower and upper bounds
  # ============================================================================

  # rounded values, needed for equality:
  r.surv.step <- round(surv.step, 8)
  r.surv.grid <- round(surv.grid, 8)

  for (i in 1:length(surv.grid)) {
  intervalcens.lower[i] <- if (any(r.surv.step == r.surv.grid[i])) {
    mean(c(step.lower.time[which(r.surv.step == r.surv.grid[i])],
         step.lower.time[which(r.surv.step == r.surv.grid[i]) + 1]))
    } else {
    step.lower.time[which(surv.step < surv.grid[i] &
                          c(1, surv.step[-length(surv.step)]) > surv.grid[i])]
  } }

  for (i in 1:length(surv.grid)) {
  intervalcens.upper[i] <- if (any(r.surv.step == r.surv.grid[i])) {
    mean(c(step.upper.time[which(r.surv.step == r.surv.grid[i])],
         step.upper.time[which(r.surv.step == r.surv.grid[i]) + 1]))
    } else {
        step.upper.time[which(surv.step < surv.grid[i] &
                          c(1, surv.step[-length(surv.step)]) > surv.grid[i])]
  } }


  # ============================================================================
  # calculate estimates for linear approximation
  # ============================================================================

if (approxim == "linear" | approxim == "all") {
  # no linear approximation possible at uppermost timestep: insert NA
  step.upper.time.lm <- ifelse(step.upper.time == Inf, NA, step.upper.time)

  # initialize model vector
  linear.model <- list()
  # create linear models for ambigous survival-time areas
  for (j in 2:(length(step.lower.time)-1)) {
    linear.model[[j]] <- lm(time.step ~ surv.step,
         data = data.frame( time.step = c(step.lower.time[j], step.upper.time.lm[j]),
                            surv.step = surv.step[(j - 1):j]))
    # some models have coeff NA:
    # - the ones before the last model are at exact timesteps and not needed later
    # - the last model is needed and returns NA for values above uppermost timestep
  }
```

```
  # split surv.grid values to apply suitable approximation procedure
  for (i in 1:length(surv.grid)) {
    # write down unique time values
    if (intervalcens.lower[i] == intervalcens.upper[i])
          {intervalcens.linear[i] <- intervalcens.lower[i]} else {

    # find suitable model for surv.grid value
     for (j in 2:(length(step.lower.time)-1)) {
       if (r.surv.grid[i] >= r.surv.step[j] & r.surv.grid[i] < r.surv.step[j - 1])
         {intervalcens.linear[i] <- suppressWarnings(predict(linear.model[[j]],
                                    newdata = data.frame(surv.step = surv.grid[i]),
                                    type = "response"))}
     }
       # suppressWarnings is necessary because else warning
       # "prediction from a rank-deficient fit may be misleading" is printed
       # this warning is not valid here because
       # we only predict inside the suitable survival probability interval

     # put NA to values where intervalcens.upper == Inf
     if (intervalcens.upper[i] == Inf) {intervalcens.linear[i] <- NA}
} } }

# ============================================================================
# add convergence indicator to intervalcens vectors
# ============================================================================

intervalcens.lower <- c(intervalcens.lower, converge)
intervalcens.upper <- c(intervalcens.upper, converge)
intervalcens.linear <- c(intervalcens.linear, converge)


# ============================================================================
# choose appropriate output:
# ============================================================================
if (approxim == "lower") {output <- intervalcens.lower}
  if (approxim == "upper") {output <- intervalcens.upper}
  if (approxim == "linear") {output <- intervalcens.linear}
  if (approxim == "all") {output <- list(intervalcens.lower = intervalcens.lower,
                                         intervalcens.upper = intervalcens.upper,
                                         intervalcens.linear = intervalcens.linear)}
  output
}
```

TimeWeibUncens

```
> ## ============================================================================
> ## this script contains a function
> ## that for specific survival probabilities (surv.grid) estimates the survival timepoints
> ## corresponding to looking at the quantile function of the parametric Weibull distribution
> ## ============================================================================
>
>
> TimeWeibUncens <- function (surv.grid, event.time) {

 # ============================================================================
 # with uncensored data:
 # calculate parametric Weibull estimates of inverse survival function
 # at specific survival probabilities surv.grid
 # and save in vectors

 # Args:
 # surv.grid: vector of survival probabilities that should be evaluated
 # event.time: vector of event times of one dataset
 #             all events have to be observed, no censoring allowed

 # Returns:
 # vector of time estimates for each survival probability in surv.grid
 # ============================================================================
```

```
# initialize result vector
uncens        <- rep(NA, length(surv.grid))

# calculate estimate
weibull.est <- survreg(Surv(event.time, rep(1, length(event.time))) ~ 1,
                       dist = "weibull")

summary(weibull.est)
summary(weibull.est)$table
vcov(weibull.est)

#extract parameters from summary
weib.b <- exp(summary(weibull.est)$table[2]) # = scale
weib.u <- summary(weibull.est)$table[1]      # = intercept
weib.b.sd <- exp(summary(weibull.est)$table[4]) # = scale
weib.u.sd <- summary(weibull.est)$table[3]      # = intercept

# compute time estimates for surv.grid
uncens <- qsurvreg(1 - surv.grid, weib.u, weib.b, distr = "weibull")

#output
uncens
}
```

TimeWeibRightcens

```
> ## ============================================================================
> ## this script contains a function
> ## that for specific survival probabilities (surv.grid) estimates the survival timepoints
> ## corresponding to looking at the quantile function of the parametric Weibull distribution
> ## ============================================================================
>
>
> TimeWeibRightcens <- function (surv.grid, rightcens.time, rightcens.ind) {

 # ============================================================================
 # with rightcensored data:
 # calculate parametric Weibull estimates of inverse survival function
 # at specific survival probabilities surv.grid
 # and save in vectors

 # Args:
 # surv.grid: vector of survival probabilities that should be evaluated
 # rightcens.time: vector of event and right-censoring times of one dataset
 # rightcens.ind: vector of censoring indicator (1 = event time, 0 = censoring time)
 # (right-censoring means: event has happened any time after the censoring time)

 # Returns:
 # vector of time estimates for each survival probability in surv.grid
 # ============================================================================

 # initialize result vector
 rightcens        <- rep(NA, length(surv.grid))

 # calculate estimate
 weibull.est <- survreg(Surv(rightcens.time, rightcens.ind) ~ 1,
                        dist = "weibull")

 summary(weibull.est)
 summary(weibull.est)$table
 vcov(weibull.est)

 #extract parameters from summary
 weib.b <- exp(summary(weibull.est)$table[2]) # = scale
 weib.u <- summary(weibull.est)$table[1]      # = intercept
 weib.b.sd <- exp(summary(weibull.est)$table[4]) # = scale
 weib.u.sd <- summary(weibull.est)$table[3]      # = intercept

 # compute time estimates for surv.grid
```

```
 rightcens <- qsurvreg(1 - surv.grid, weib.u, weib.b, distr = "weibull")

 #output
 rightcens
 }
```

TimeWeibIntervalcens

```
> ## ============================================================================
> ## this script contains a function
> ## that for specific survival probabilities (surv.grid) estimates the survival timepoints
> ## corresponding to looking at the quantile function of the parametric Weibull distribution
> ## ============================================================================
>
>
> TimeWeibIntervalcens <- function (surv.grid, left.time, right.time) {

 # ============================================================================
 # with intervalcensored data:
 # calculate parametric Weibull estimates of inverse survival function
 # at specific survival probabilities surv.grid
 # and save in vector

 # Args:
 # surv.grid: vector of survival probabilities that should be evaluated
 # left.time: vector of left boundary of interval-censored times of one dataset
 # right.time: vector of right boundary of interval-censored times of one dataset

 # Returns:
 # vector of time estimates for each survival probability in surv.grid
 # ============================================================================

 # initialize result vector
 intervalcens        <- rep(NA, length(surv.grid))

 # construct additional vector with indicator for interval- and right-censoring
 # (right-censored = 0, interval-censored = 3, not censored = 1)
 cens.ind <- rep(NA, length(left.time))
 cens.ind <- ifelse(!(right.time == Inf) & !(left.time == right.time), 3, 1)
 cens.ind <- ifelse(right.time == Inf, 0, cens.ind)

 # calculate estimate
 weibull.est <- survreg(Surv(left.time, right.time, cens.ind, type = "interval") ~ 1,
                        dist = "weibull")

 summary(weibull.est)
 summary(weibull.est)$table
 vcov(weibull.est)

 #extract parameters
 weib.b <- exp(summary(weibull.est)$table[2]) # = scale
 weib.u <- summary(weibull.est)$table[1]      # = intercept
 weib.b.sd <- exp(summary(weibull.est)$table[4]) # = scale
 weib.u.sd <- summary(weibull.est)$table[3]      # = intercept

 # compute time estimates for surv.grid
 intervalcens <- qsurvreg(1 - surv.grid, weib.u, weib.b, distr = "weibull")

 #output
 intervalcens
 }
```

SurvStepUncens

```
> ## ============================================================================
> ## this script contains a function
> ## that for specific timepoints (time.grid) calculates the
> ## nonparametric (Kaplan-Meier or Turnbull) survival estimates
```

```
> ## ============================================================================
>
>
> SurvStepUncens <- function (time.grid, event.time, CI = FALSE) {

 # ============================================================================
 # with uncensored data:
 # calculate nonparametric (Kaplan-Meier) estimates of survival function S(t)
 # and their confidence interval boundaries at specific timepoints time.grid
 # and save in vectors

 # Args:
 # time.grid: vector of timepoints that should be evaluated
 # event.time: vector of event times of one dataset
 #             all events have to be observed, no censoring allowed
 # CI: logical value indicating if 95% confidence interval boundaries are added
 #     to the result and result is output in a list instead of a vector

 # Returns:
 # if confint = FALSE :
 # vector of survival estimates for each timepoint in time.grid
 # if confint = TRUE :
 # list of three vectors: uncens, uncens.CI.low, uncens.CI.up:
 # vectors of survival estimates and their 95% confidence interval boundaries
 # for each timepoint in time.grid
 # ============================================================================

 # initialize result vectors
 uncens        <- rep(NA, length(time.grid))
 uncens.CI.low <- rep(NA, length(time.grid))
 uncens.CI.up  <- rep(NA, length(time.grid))

 # calculate estimate
 survfit.est <- survfit(Surv(event.time, rep(1, length(event.time))) ~ 1,
                        conf.type = "log-log")

 # extract numbers from summary
 step.time <- summary(survfit.est)$time
 surv.step <- summary(survfit.est)$surv
 surv.step.CI.low <- summary(survfit.est)$lower
 surv.step.CI.up <- summary(survfit.est)$upper

 # and add first line: time = 0, survival = 1
 step.time <- c(0,step.time)
 surv.step <- c(1,surv.step)
 surv.step.CI.low <- c(1,surv.step.CI.low)
 surv.step.CI.up <- c(1,surv.step.CI.up)

 # expand estimates to grid
 uncens        <- surv.step       [findInterval(time.grid, step.time)]
 uncens.CI.low <- surv.step.CI.low[findInterval(time.grid, step.time)]
 uncens.CI.up  <- surv.step.CI.up [findInterval(time.grid, step.time)]

 # decide between the two output versions
 if (CI == TRUE) {output <- list(uncens, uncens.CI.low, uncens.CI.up)
                  names(output) <- c("uncens", "uncens.CI.low", "uncens.CI.up")
 } else {output <- uncens}
 output
 }
```

SurvStepRightcens

```
> ## ============================================================================
> ## this script contains a function
> ## that for specific timepoints (time.grid) calculates the
> ## nonparametric (Kaplan-Meier or Turnbull) survival estimates
> ## ============================================================================
>
>
```

```
> SurvStepRightcens <- function (time.grid, rightcens.time, rightcens.ind, CI = FALSE) {

  # ================================================================================
  # with rightcensored data:
  # calculate nonparametric (Kaplan-Meier) estimates of survival function S(t)
  # and their confidence interval boundaries at specific timepoints time.grid
  # and save in vectors

  # Args:
  # time.grid: vector of timepoints that should be evaluated
  # rightcens.time: vector of event and right-censoring times of one dataset
  # rightcens.ind: vector of censoring indicator (1 = event time, 0 = censoring time)
  # (right-censoring means: event has happened any time after the censoring time)
  # CI: logical value indicating if 95% confidence interval boundaries are added
  #     to the result and result is output in a list instead of a vector

  # Returns:
  # if confint = FALSE :
  # vector of survival estimates for each timepoint in time.grid
  # if confint = TRUE :
  # list of three vectors: uncens, uncens.CI.low, uncens.CI.up:
  # vectors of survival estimates and their 95% confidence interval boundaries
  # for each timepoint in time.grid
  # ================================================================================

  # initialize result vectors
  rightcens        <- rep(NA, length(time.grid))
  rightcens.CI.low <- rep(NA, length(time.grid))
  rightcens.CI.up  <- rep(NA, length(time.grid))

  # calculate estimate
  survfit.est <- survfit(Surv(rightcens.time, rightcens.ind) ~ 1, conf.type = "log-log")

  # extract numbers from summary
  step.time <- summary(survfit.est)$time
  surv.step <- summary(survfit.est)$surv
  surv.step.CI.low <- summary(survfit.est)$lower
  surv.step.CI.up <- summary(survfit.est)$upper

  # and add first line: time = 0, survival = 1
  step.time <- c(0,step.time, max(rightcens.time))
  surv.step <- c(1,surv.step, 0)
  surv.step.CI.low <- c(1,surv.step.CI.low, 0)
  surv.step.CI.up <- c(1,surv.step.CI.up, 0)

  cbind(step.time, surv.step, surv.step.CI.low, surv.step.CI.up)

  # expand estimates to grid
  rightcens        <- surv.step       [findInterval(time.grid, step.time)]
  rightcens.CI.low <- surv.step.CI.low[findInterval(time.grid, step.time)]
  rightcens.CI.up  <- surv.step.CI.up [findInterval(time.grid, step.time)]

  # put NA to values beyond time.max, where rightcens == 0, because is undefined
  for (i in 1:length(time.grid)) {
    if (rightcens[i] == 0) {
      rightcens[i] <- NA
      rightcens.CI.low[i] <- NA
      rightcens.CI.up[i] <- NA}
  }

  # decide between the two output versions
  if (CI == TRUE) {output <- list(rightcens, rightcens.CI.low, rightcens.CI.up)
                   names(output) <- c("rightcens", "rightcens.CI.low", "rightcens.CI.up")
  } else {output <- rightcens}
  output
 }
```

SurvStepIntervalcens

```
> ## ============================================================================
> ## this script contains a function
> ## that for specific timepoints (time.grid) calculates the
> ## nonparametric (Kaplan-Meier or Turnbull) survival estimates
> ## ============================================================================
>
>
> SurvStepIntervalcens <- function (time.grid, left.time, right.time,
                                    approxim = "linear", tol = 10^-5) {

   # ============================================================================
   # with intervalcensored data:
   # calculate nonparametric (Turnbull) estimates of survival function S(t)
   # at specific timepoints time.grid and save in vectors
   # In some time ranges the survival ML-estimate is ambigous,
   # it can be any value between lower and upper boundaries.
   # The user can choose the output: either one boundary value or
   # a value assuming linear change between the unique border values

   # Args:
   # time.grid: vector of timepoints that should be evaluated
   # left.time: vector of left boundary of interval-censored times of one dataset
   # right.time: vector of right boundary of interval-censored times of one dataset
   # approxim: define how non-unique estimates should be handled (see below)
   # tol: tolerance,
   # algorithm terminates when difference in MLE between two steps is below tolerance

   # Returns:
   # depending on handling of non-unique estimates:
   # if approxim = "linear" :
   # vector of survival estimates with linear change in ambigous areas, followed by
   # the numerical value corresponding to the logical indicator of convergence
   # of the NPMLE algorithm (0 = FALSE, 1 = TRUE)
   # if approxim = "lower" :
   # vector of survival estimates at lower boundary followed by the same convergence indicator
   # if approxim = "upper" :
   # vector of survival estimates at upper boundary followed by the same convergence indicator
   # if approxim = "all" :
   # list of the three vectors above,
   # named intervalcens.linear, intervalcens.lower, intervalcens.upper
   # ============================================================================

   # check approxim entry
   approxim <- match.arg(approxim, c("lower", "upper", "linear", "all"))

   # initialize result vectors
   intervalcens.lower   <- rep(NA, length(time.grid))
   intervalcens.upper  <- rep(NA, length(time.grid))
   intervalcens.linear <- rep(NA, length(time.grid))

   # save censored values as data, add dummy variable (0,1) as second variable
   # (because needed input format is observation rectangles)
   data <- cbind(left.time = left.time,
                 right.time = right.time,
                 dum.left = rep(0, length(left.time)),
                 dum.right = rep(1, length(left.time)))

   data <- as.matrix(data)

   # define boundaries (0 = open, 1 = closed)
   # (have to be 1,1 for exact values, and are set to 0,1 for intervals, because mathematically easier)
   bound1 <- matrix(rep(NA, times = 2 * length(left.time)), ncol = 2)
   for (i in 1:length(left.time)) {
     if (left.time[i] == right.time[i]) {bound1[i, ] <- c(1, 1)} else {
                                        bound1[i, ] <- c(0, 1)}
   }
   bound2 <- matrix(c(rep(0, times = length(left.time)),
                      rep(1, times = length(left.time))), ncol = 2)
```

```
  bound <- cbind(bound1, bound2)

  # ============================================================================
  # calculate NPMLE
  # ============================================================================
  MLEcens.est <- computeMLE(data, bound, tol = tol)

  # save convergence indicator
  converge <- MLEcens.est$conv

  # extract numbers from output
  step.lower.time <- MLEcens.est$rects[, 1]
  step.upper.time <- MLEcens.est$rects[, 2]
  surv.step <- 1 - cumsum(MLEcens.est$p)

  # and add first line: time = 0, survival = 1
  step.lower.time <- c(0,step.lower.time)
  step.upper.time <- c(0,step.upper.time)
  surv.step <- c(1,surv.step)

  # display in a dataframe
  cbind(step.lower.time, step.upper.time, surv.step)

  # ============================================================================
  # choose estimates for lower and upper bounds
  # ============================================================================
  intervalcens.lower <- surv.step[findInterval(time.grid, step.lower.time)]
  intervalcens.upper <- surv.step[findInterval(time.grid, step.upper.time)]

  # ============================================================================
  # calculate linear change values:
  # ============================================================================
if (approxim == "linear" | approxim == "all") {
  # no linear approximation possible at uppermost timestep: insert NA
  step.upper.time.lm <- ifelse(step.upper.time == Inf, NA, step.upper.time)

  # initialize model vector
  linear.model <- list()
  # create linear models for ambigous survival-time areas
  for (j in 2:length(step.lower.time)) {
    linear.model[[j]] <- lm(surv.step ~ time.step,
         data = data.frame( time.step = c(step.lower.time[j], step.upper.time.lm[j]),
                            surv.step = surv.step[(j - 1):j]))
    # some models have coeff NA:
    # - the ones before the last model are at exact timesteps and not needed later
    # - the last model is needed and returns NA for values above uppermost timestep
  }

  # split time.grid values to apply suitable approximation procedure
  for (i in 1:length(time.grid)) {
    # write down unique survival values
    if (intervalcens.lower[i] == intervalcens.upper[i])
          {intervalcens.linear[i] <- intervalcens.lower[i]} else {

    # find suitable model for time.grid value
     for (j in 2:length(step.lower.time)) {
       if (time.grid[i] >= step.lower.time[j] & time.grid[i] < step.upper.time[j])
         {intervalcens.linear[i] <- suppressWarnings(predict(linear.model[[j]],
                                          newdata = data.frame(time.step = time.grid[i]),
                                          type = "response"))}
     }
       # suppressWarnings is necessary because else warning
       # "prediction from a rank-deficient fit may be misleading" is printed
       # this warning is not valid here because
       # we only predict inside the suitable time interval

    # put NA to values where intervalcens.lower < 10^-3
    if (intervalcens.lower[i] < 10^-3) {
      intervalcens.linear[i] <- NA}
          } } }
```

```
# ============================================================================
# add convergence indicator to intervalcens vectors
# ============================================================================

intervalcens.lower <- c(intervalcens.lower, converge)
intervalcens.upper <- c(intervalcens.upper, converge)
intervalcens.linear <- c(intervalcens.linear, converge)

# ============================================================================
# choose appropriate output:
# ============================================================================

  if (approxim == "lower") {output <- intervalcens.lower}
  if (approxim == "upper") {output <- intervalcens.upper}
  if (approxim == "linear") {output <- intervalcens.linear}
  if (approxim == "all") {output <- list(intervalcens.lower = intervalcens.lower,
                                         intervalcens.upper = intervalcens.upper,
                                         intervalcens.linear = intervalcens.linear)}
  output
}
```

SurvWeibUncens

```
> ## ============================================================================
> ## this script contains a function
> ## that for specific timepoints (time.grid) calculates the
> ## parametric Weibull survival estimates
> ## ============================================================================
>
>
> SurvWeibUncens <- function (time.grid, event.time) {

 # ============================================================================
 # with uncensored data:
 # calculate parametric Weibull estimates of survival function S(t)
 # at specific timepoints time.grid
 # and save in vectors

 # Args:
 # time.grid: vector of timepoints that should be evaluated
 # event.time: vector of event times of one dataset
 #             all events have to be observed, no censoring allowed

 # Returns:
 # vector of survival estimates for each timepoint in time.grid
 # ============================================================================

 # initialize result vector
 uncens        <- rep(NA, length(time.grid))

 # calculate estimate
 weibull.est <- survreg(Surv(event.time, rep(1, length(event.time))) ~ 1,
                        dist = "weibull")

 summary(weibull.est)
 summary(weibull.est)$table
 vcov(weibull.est)

 #extract parameters from summary
 weib.b <- exp(summary(weibull.est)$table[2]) # = scale
 weib.u <- summary(weibull.est)$table[1]      # = intercept
 weib.b.sd <- exp(summary(weibull.est)$table[4]) # = scale
 weib.u.sd <- summary(weibull.est)$table[3]      # = intercept


 # compute surv estimates for time.grid
 uncens <- 1 - psurvreg(time.grid, weib.u, weib.b, distr = "weibull")
```

```
#output
uncens
}
```

SurvWeibRightcens

```
> ## ============================================================================
> ## this script contains a function
> ## that for specific timepoints (time.grid) calculates the
> ## parametric Weibull survival estimates
> ## ============================================================================
>
>
> SurvWeibRightcens <- function (time.grid, rightcens.time, rightcens.ind) {

 # ============================================================================
 # with rightcensored data:
 # calculate parametric Weibull estimates of survival function S(t)
 # at specific timepoints time.grid
 # and save in vector

 # Args:
 # time.grid: vector of timepoints that should be evaluated
 # rightcens.time: vector of event and right-censoring times of one dataset
 # rightcens.ind: vector of censoring indicator (1 = event time, 0 = censoring time)
 # (right-censoring means: event has happened any time after the censoring time)

 # Returns:
 # vector of survival estimates for each timepoint in time.grid
 # ============================================================================

 # initialize result vector
 rightcens        <- rep(NA, length(time.grid))

 # calculate estimate
 weibull.est <- survreg(Surv(rightcens.time, rightcens.ind) ~ 1,
                        dist = "weibull")

 summary(weibull.est)
 summary(weibull.est)$table
 vcov(weibull.est)

 #extract parameters from summary
 weib.b <- exp(summary(weibull.est)$table[2]) # = scale
 weib.u <- summary(weibull.est)$table[1]      # = intercept
 weib.b.sd <- exp(summary(weibull.est)$table[4]) # = scale
 weib.u.sd <- summary(weibull.est)$table[3]      # = intercept


 # compute surv estimates for time.grid
 rightcens <- 1 - psurvreg(time.grid, weib.u, weib.b, distr = "weibull")

 #output
 rightcens
 }
```

SurvWeibIntervalcens

```
> ## ============================================================================
> ## this script contains a function
> ## that for specific timepoints (time.grid) calculates the
> ## parametric Weibull survival estimates
> ## ============================================================================
>
>
> SurvWeibIntervalcens <- function (time.grid, left.time, right.time) {

 # ============================================================================
```

```
# with intervalcensored data:
# calculate parametric Weibull estimates of survival function S(t)
# at specific timepoints time.grid
# and save in vector

# Args:
# time.grid: vector of timepoints that should be evaluated
# left.time: vector of left boundary of interval-censored times of one dataset
# right.time: vector of right boundary of interval-censored times of one dataset

# Returns:
# vector of survival estimates for each timepoint in time.grid
# ============================================================================

# initialize result vector
intervalcens        <- rep(NA, length(time.grid))

# construct additional vector with indicator for interval- and right-censoring
# (right-censored = 0, interval-censored = 3, not censored = 1)
cens.ind <- rep(NA, length(left.time))
cens.ind <- ifelse(!(right.time == Inf) & !(left.time == right.time), 3, 1)
cens.ind <- ifelse(right.time == Inf, 0, cens.ind)

# calculate estimate
weibull.est <- survreg(Surv(left.time, right.time, cens.ind, type = "interval") ~ 1,
                       dist = "weibull")

summary(weibull.est)
summary(weibull.est)$table
vcov(weibull.est)

# extract parameters
weib.b <- exp(summary(weibull.est)$table[2]) # = scale
weib.u <- summary(weibull.est)$table[1]      # = intercept
weib.b.sd <- exp(summary(weibull.est)$table[4]) # = scale
weib.u.sd <- summary(weibull.est)$table[3]      # = intercept

# compute surv estimates for grid
intervalcens <- 1 - psurvreg(time.grid, weib.u, weib.b, distr = "weibull")

# output
intervalcens
}
```

8.2.3 Estimation of Log-rank test p-values: LogRankTest... functions

Three functions are defined, for uncensored (complete), rightcensored and intervalcensored data. For uncensored data the rightcensoring methods are used.

LogRankTestUncens

```
> ## ============================================================================
> ## this script contains a function
> ## that for two study arms calculates the
> ## logrank test statistic and its p-value of equal survival curves
> ## ============================================================================
>
>
> LogRankTestUncens <- function (event.time1, event.time2, stat = FALSE) {

 # ============================================================================
 # with uncensored data:
 # calculate logrank test statistic
 # and save
```

```
# Args:
# event.time1: vector of event times of the first study arm
# event.time2: vector of event times of the second study arm
# stat: logical value indicating if chisquare statistic values are added
#       to the result and result is output in a list instead of a vector

# Returns:
# if stat = FALSE :
# p-value of the logrank test
# if stat = TRUE :
# list of two values: p-value, chisquare statistic:
# logrank test p-value and its chisquare statistic
# ================================================================================

# form dataset from vectors
data <- data.frame(event.time = c(event.time1, event.time2),
                   treatment = c(rep(0, times = length(event.time1)),
                                 rep(1, times = length(event.time2))))
data$treatment <- as.factor(data$treatment)

# calculate estimate
survdiff.mod <- survdiff(Surv(data$event.time, rep(1, nrow(data))) ~ treatment,
                         data = data, rho = 0)

# extract numbers from summary
statistic <- survdiff.mod$chisq
p.value    <- 1 - pchisq(survdiff.mod$chisq, df = 1)

# put estimates to vector
uncens      <- p.value
uncens.stat <- statistic

# decide between the two output versions
if (stat == TRUE) {output <- list(uncens, uncens.stat)
                   names(output) <- c("uncens", "uncens.stat")
} else {output <- uncens}
output
}
```

LogRankTestRightcens

```
> ## ================================================================================
> ## this script contains a function
> ## that for two study arms calculates the
> ## logrank test statistic and its p-value of equal survival curves
> ## ================================================================================
>
>
> LogRankTestRightcens <- function (rightcens.time1, rightcens.time2,
                            rightcens.ind1, rightcens.ind2, stat = FALSE) {

 # ================================================================================
 # with rightcensored data:
 # calculate logrank test statistic
 # and save

 # Args:
 # rightcens.time1: vector of rightcensored event times of the first study arm
 # rightcens.time2: vector of rightcensored event times of the second study arm
 # rightcens.ind1: vector of rightcensoring event time indicators of the first study arm
 #                 (0 = censored, 1 = event)
 # rightcens.ind2: vector of the same indicators of the second study arm
 # stat: logical value indicating if chisquare statistic values are added
 #       to the result and result is output in a list instead of a vector

 # Returns:
 # if stat = FALSE :
 # p-value of the logrank test
 # if stat = TRUE :
```

```
 # list of two values: p-value, chisquare statistic:
 # logrank test p-value and its chisquare statistic
 # ================================================================================

 # check length of vectors:
 if (length(rightcens.time1) != length(rightcens.ind1))
 {warning("rightcens.time1 and rightcens.ind1 vectors do not have the same size")}
 if (length(rightcens.time2) != length(rightcens.ind2))
 {warning("rightcens.time2 and rightcens.ind2 vectors do not have the same size")}

 # form dataset from vectors
 data <- data.frame(rightcens.time = c(rightcens.time1, rightcens.time2),
                    rightcens.ind = c(rightcens.ind1, rightcens.ind2),
                    treatment = c(rep(0, times = length(rightcens.time1)),
                                  rep(1, times = length(rightcens.time2))))
 data$treatment <- as.factor(data$treatment)

 # calculate estimate
 survdiff.mod <- survdiff(Surv(rightcens.time, rightcens.ind) ~ treatment,
                          data = data, rho = 0)

 # extract numbers from summary
 statistic <- survdiff.mod$chisq
 p.value    <- 1 - pchisq(survdiff.mod$chisq, df = 1)

 # put estimates to vector
 rightcens        <- p.value
 rightcens.stat <- statistic

 # decide between the two output versions
 if (stat == TRUE) {output <- list(rightcens, rightcens.stat)
                    names(output) <- c("rightcens", "rightcens.stat")
 } else {output <- rightcens}
 output
 }
>
```

LogRankTestIntervalcens

```
> ## ================================================================================
> ## this script contains a function
> ## that for two study arms calculates the
> ## logrank test statistic and its p-value of equal survival curves
> ## ================================================================================
>
>
> LogRankTestIntervalcens <- function (left.time1, left.time2,
                                 right.time1, right.time2,
                                 maxit = 10000, stat = FALSE, type = "pFinkelstein") {

 # ================================================================================
 # with intervalcensored data:
 # calculate logrank test statistic (three variants)
 # and save

 # Args:
 # left.time1: vector of left borders of intervalcensored event times of the first study arm
 # left.time2: vector of left borders of the second study arm
 # right.time1: vector of right borders of intervalcensored event times of the first study arm
 # right.time2: vector of right borders of the second study arm
 # maxit: the maximum of iterations used in the icfit NPMLE algorithm
 # stat: logical value indicating if chisquare statistic values are added
 #       to the result and result is output in a list instead of a vector
 # type: one of four: "pFinkelstein", "pSun", "sFinkelstein" or "sSun"
 #       these are test statistics described in two articles:
 #       Finkelstein (1986) and Sun (1996), in two forms: permutation or score
 #       see documentation of ictest function for more details

 # Returns:
```

```
# if stat = FALSE :
# a vector with two entries: the p-value of the logrank test and
#                            the numerical value corresponding to the
#                            logical indicator of convergence of the NPMLE algorithm
#                            (0 = FALSE, 1 = TRUE)
# if stat = TRUE :
# list: first element: vector of p-value and convergence indicator, (of logrank test)
#       second element: numeric of chisquare  or z statistic:
#                       (chisquare for score form or z for permutation form of statistic)
# ============================================================================

# check length of vectors:
if (length(left.time1) != length(right.time1))
{warning("left.time1 and right.time1 vectors do not have the same size")}
if (length(left.time2) != length(right.time2))
{warning("left.time2 and right.time2 vectors do not have the same size")}

# check types
type <- match.arg(type, c("pFinkelstein", "pSun", "sFinkelstein", "sSun"))


# form dataset from vectors
data <- data.frame(left.time = c(left.time1, left.time2),
                   right.time = c(right.time1, right.time2),
                   treatment = c(rep(0, times = length(right.time1)),
                                 rep(1, times = length(right.time2))))
data$treatment <- as.factor(data$treatment)

# calculate estimate

if (type == "pFinkelstein")
{icfit.mod <- icfit(Surv(left.time, right.time, type = "interval2") ~ 1, data = data,
                    control = icfitControl(maxit = maxit))
ictest.res <- ictest(Surv(left.time, right.time, type = "interval2") ~ treatment,
                     data = data, scores = "logrank2",
                     alternative = "two.sided", icFIT = icfit.mod)}

if (type == "pSun")
{icfit.mod <- icfit(Surv(left.time, right.time, type = "interval2") ~ 1, data = data,
                    control = icfitControl(maxit = maxit))
 ictest.res <- ictest(Surv(left.time, right.time, type = "interval2") ~ treatment,
                     data = data, scores = "logrank1",
                     alternative = "two.sided", icFIT = icfit.mod)}

if (type == "sFinkelstein")
{icfit.mod <- icfit(Surv(left.time, right.time, type = "interval2") ~ 1, data = data,
                    control = icfitControl(maxit = maxit))
 ictest.res <- ictest(Surv(left.time, right.time, type = "interval2") ~ treatment,
                     data = data, scores = "logrank2", method = "scoretest",
                     alternative = "two.sided", icFIT = icfit.mod)}

if (type == "sSun")
{icfit.mod <- icfit(Surv(left.time, right.time, type = "interval2") ~ 1, data = data,
                    control = icfitControl(maxit = maxit))
 ictest.res <- ictest(Surv(left.time, right.time, type = "interval2") ~ treatment,
                     data = data, scores = "logrank1", method = "scoretest",
                     alternative = "two.sided", icFIT = icfit.mod)}

# extract numbers from summary
statistic <- ictest.res$statistic
p.value    <- ictest.res$p.value
converge <- ictest.res$fit$converge

# put estimates to vector
intervalcens        <- c(p.value, converge)
intervalcens.stat <- statistic


# decide between the two output versions
if (stat == TRUE) {output <- list(intervalcens, intervalcens.stat)
```

```
                names(output) <- c("intervalcens", "intervalcens.stat")
 } else {output <- intervalcens}
 output
 }
>
```

8.2.4 Estimation of proportional hazard hazard ratios: `HazRat...` functions

Three functions are defined, for uncensored (complete), rightcensored and intervalcensored data. For uncensored data the rightcensoring methods are used. For intervalcensored data one can choose between two types of estimation.

`HazRatUncens`

```
> ## ============================================================================
> ## this script contains a function
> ## that for two study arms calculates the
> ## hazard ratio estimates determined by the Cox model
> ## ============================================================================
>
>
> HazRatUncens <- function (event.time1, event.time2, CI = FALSE) {

 # ============================================================================
 # with uncensored data:
 # calculate hazard ratio estimates by Cox regression
 # and save

 # Args:
 # event.time1: vector of event times of the first study arm
 # event.time2: vector of event times of the second study arm
 # CI: logical value indicating if 95% confidence interval boundaries are added
 #     to the result and result is output in a list instead of a vector

 # Returns:
 # if confint = FALSE :
 # hazard ratio estimate (second study arm / first study arm)
 # if confint = TRUE :
 # list of three values: uncens, uncens.CI.low, uncens.CI.up:
 # hazard ratio estimate and its 95% confidence interval boundaries
 # ============================================================================

 # form dataset from vectors
 data <- data.frame(event.time = c(event.time1, event.time2),
                    treatment = c(rep(0, times = length(event.time1)),
                                  rep(1, times = length(event.time2))))
 data$treatment <- as.factor(data$treatment)

 # calculate estimate
 coxph.mod <- coxph(Surv(event.time, rep(1, nrow(data))) ~ treatment, data = data)

 # extract numbers from summary
 hazrat.est    <- summary(coxph.mod)$coef[2]
 hazrat.CI.low <- summary(coxph.mod)$conf.int[3]
 hazrat.CI.up  <- summary(coxph.mod)$conf.int[4]

 # put estimates to vector
 uncens        <- hazrat.est
 uncens.CI.low <- hazrat.CI.low
 uncens.CI.up  <- hazrat.CI.up

 # decide between the two output versions
 if (CI == TRUE) {output <- list(uncens, uncens.CI.low, uncens.CI.up)
                  names(output) <- c("uncens", "uncens.CI.low", "uncens.CI.up")
```

```
 } else {output <- uncens}
 output
 }
>
```

HazRatRightcens

```
> ## ============================================================================
> ## this script contains a function
> ## that for two study arms calculates the
> ## hazard ratio estimates determined by the Cox model
> ## ============================================================================
>
>
> HazRatRightcens <- function (rightcens.time1, rightcens.time2,
                          rightcens.ind1, rightcens.ind2, CI = FALSE) {

 # ============================================================================
 # with rightcensored data:
 # calculate hazard ratio estimates by Cox regression
 # and save

 # Args:
 # rightcens.time1: vector of rightcensored event times of the first study arm
 # rightcens.time2: vector of rightcensored event times of the second study arm
 # rightcens.ind1: vector of rightcensoring event time indicators of the first study arm
 #                 (0 = censored, 1 = event)
 # rightcens.ind2: vector of the same indicators of the second study arm
 # CI: logical value indicating if 95% confidence interval boundaries are added
 #     to the result and result is output in a list instead of a vector

 # Returns:
 # if confint = FALSE :
 # hazard ratio estimate (second study arm / first study arm)
 # if confint = TRUE :
 # list of three values: rightcens, rightcens.CI.low, rightcens.CI.up:
 # hazard ratio estimate and its 95% confidence interval boundaries
 # ============================================================================

 # check length of vectors:
 if (length(rightcens.time1) != length(rightcens.ind1))
 {warning("rightcens.time1 and rightcens.ind1 vectors do not have the same size")}
 if (length(rightcens.time2) != length(rightcens.ind2))
 {warning("rightcens.time2 and rightcens.ind2 vectors do not have the same size")}

 # form dataset from vectors
 data <- data.frame(rightcens.time = c(rightcens.time1, rightcens.time2),
                    rightcens.ind = c(rightcens.ind1, rightcens.ind2),
                    treatment = c(rep(0, times = length(rightcens.time1)),
                                  rep(1, times = length(rightcens.time2))))
 data$treatment <- as.factor(data$treatment)

 # calculate estimate
 coxph.mod <- coxph(Surv(rightcens.time, rightcens.ind) ~ treatment, data = data)

 # extract numbers from summary
 hazrat.est    <- summary(coxph.mod)$coef[2]
 hazrat.CI.low <- summary(coxph.mod)$conf.int[3]
 hazrat.CI.up  <- summary(coxph.mod)$conf.int[4]

 # put estimates to vector
 rightcens        <- hazrat.est
 rightcens.CI.low <- hazrat.CI.low
 rightcens.CI.up  <- hazrat.CI.up

 # decide between the two output versions
 if (CI == TRUE) {output <- list(rightcens, rightcens.CI.low, rightcens.CI.up)
                  names(output) <- c("rightcens", "rightcens.CI.low", "rightcens.CI.up")
 } else {output <- rightcens}
```

```
 output
 }
>
>
```

HazRatIntervalcens

```
> ## ============================================================================
> ## this script contains a function
> ## that for two study arms calculates the
> ## hazard ratio estimates determined by a proportional hazard model
> ## ============================================================================
>
>
> HazRatIntervalcens <- function (left.time1, left.time2,
                                  right.time1, right.time2, CI = FALSE, type = "intcox",
                                  Pan.numrep = rep(20,50)) {

 # ============================================================================
 # with intervalcensored data:
 # calculate hazard ratio estimates by Cox regression
 # and save

 # Args:
 # left.time1: vector of left borders of intervalcensored event times of the first study arm
 # left.time2: vector of left borders of the second study arm
 # right.time1: vector of right borders of intervalcensored event times of the first study arm
 # right.time2: vector of right borders of the second study arm
 # CI: logical value indicating if 95% confidence interval boundaries are added
 #     to the result and result is output in a list instead of a vector
 # type: one of two: "intcox" or "Pan", intcox is default
 #       these are estimation methods described in the following articles:
 #       intcox: Henschel (2009), Pan: Goggins (1998) and Pan (2000)
 #       many thanks to David Dejardin who coded the second one
 #Pan.numrep: vector of the form rep(number, repetitions), filled with two numbers:
 #            - number: number of datasets generated for estimation
 #            - repetitions: number of iteration steps

 # Returns:
 # if confint = FALSE :
 # hazard ratio estimate (second study arm / first study arm)
 # if confint = TRUE :
 # list of three values: intervalcens, intervalcens.CI.low, intervalcens.CI.up:
 # hazard ratio estimate and its 95% confidence interval boundaries
 # ============================================================================

 # check length of vectors:
 if (length(left.time1) != length(right.time1))
 {warning("left.time1 and right.time1 vectors do not have the same size")}
 if (length(left.time2) != length(right.time2))
 {warning("left.time2 and right.time2 vectors do not have the same size")}

 # check types
 type <- match.arg(type, c("intcox", "Pan"))


 # ============================================================================
 # dataset creation
 # ============================================================================

 # form dataset from vectors
 data <- data.frame(left.time = c(left.time1, left.time2),
                    right.time = c(right.time1, right.time2),
                    treatment = c(rep(0, times = length(right.time1)),
                                  rep(1, times = length(right.time2))))
 data$treatment <- as.factor(data$treatment)

 # reconfigure dataset in two ways:
 # - introduce small interval for exact (= death) events
```

```
# - replace "Inf" by "NA" for right-censored events
data[data$right.time == data$left.time, "left.time"] <- data[data$right.time == data$left.time, "left.time"] - 0.1
data[data$right.time - 0.1 == data$left.time, "right.time"] <- data[data$right.time - 0.1 == data$left.time, "right.time"] +
data[data$right.time == "Inf", "right.time"] <- NA


# ============================================================================
# intcox estimation
# ============================================================================

if (type == "intcox") {
# calculate estimate
intcox.mod <- intcox(Surv(left.time, right.time, type = "interval2") ~ treatment,
                     data = data)

# output warning if algorithm does not converge
if (intcox.mod$termination != 1) {hazrat.est <- NA
                                  warning("The algorithm did not converge.")}

# extract numbers from summary
hazrat.est    <- summary(intcox.mod)$coef[2]
hazrat.CI.low <- NA
hazrat.CI.up  <- NA

# put estimates to vector
intervalcens        <- hazrat.est
intervalcens.CI.low <- hazrat.CI.low
intervalcens.CI.up  <- hazrat.CI.up }

# ============================================================================
# Pan estimation
# ============================================================================

if (type == "Pan") {
# choose correct ratio (treated/control):
data$treat.ratio <- as.numeric(data$treatment)

# add variables T1, T2, cov
data$T1 <- data$left.time
data$T2 <- data$right.time
data$cov <- data$treat.ratio

# set iteration parameters
nmcint <- Pan.numrep
crit <- 0.00001
rmeanl <- 40


# ============================================================================
#start Pan function
iterda <- length(nmcint)
T1_ <- data$T1
T2_ <- data$T2
C_ <- is.finite(data$T2)
cov <- data$cov
ldat <- length(T1_)

#### initial step
# uniform generation of T
betalist <- NULL
tempL0 <- NULL
for ( imc in c(1:nmcint[1]) ) {
  T_ <- T1_
  T_[C_]<- runif(n=sum(C_), min= T1_[C_], max=T2_[C_])
  pc <- coxph( Surv(T_ , C_) ~ cov , method= "efron")
  betalist <- c(betalist, pc$coefficients )
  bh <- basehaz(pc, centered = FALSE)
  tempL0 <- rbind( tempL0, matrix(c(rep( imc,length(bh$time)), bh$time, bh$hazard), ncol = 3))
  colnames(tempL0) <- c("mcint", "time", "val")
}
```

```
# define chazval and L0list functions
chazval <- function(timech, L0v.time, L0v.val) {
  L0vtime <- c(0, L0v.time)
  L0val <- c(0, L0v.val)
  tmpd.time <- rep(timech, each = length(L0vtime))
  tmpd.valtime <- rep(L0vtime, length(timech))
  tmpd.val <- rep(L0val, length(timech))
  tmpd.time2 <- tmpd.time[tmpd.time >= tmpd.valtime]
  tmpd.val2 <- tmpd.val[tmpd.time >= tmpd.valtime]
  reschval <- tapply(tmpd.val2, tmpd.time2, FUN = max, simplify = TRUE)
  reschval[timech > max(L0vtime)] <- NA
  return(reschval)
}

L0list  <- function(L0l) {
  tm_ <- sort(unique( L0l[, "time"]))
  lss_ <- unique(L0l[, "mcint"])
  L0r <- NULL
  for (ils in lss_) {
    timils <- L0l[, "time"][L0l[, "mcint"] == ils]
    valils <- L0l[, "val"][L0l[, "mcint"] == ils]
    L0r <- cbind(L0r, chazval(tm_, timils, valils))
  }
  result <- matrix(c(tm_, cummax(apply(L0r, 1, mean, na.rm = TRUE))), ncol = 2)
  colnames(result) <- c("time", "val")
  return(result)
}

curL0 <- L0list(tempL0)
curS <- matrix(c(curL0[, "time"], exp(-curL0[, "val"])), ncol = 2)
colnames(curS) <- c("time", "val")
curbeta <- mean(betalist )

tempL <- NULL
beta <- NULL
rbeta <- NULL
LL <- NULL

#### DA iteration
dares <- NULL
for (idagm in c(1: iterda)) {
  curmcint <- nmcint[idagm]
  matmc <- matrix(nrow = ldat, ncol = curmcint, data = 0)
  for (idata in c(1:ldat)) {
    cjd <- NULL
    gent <- NULL

    if (C_[idata] == 1) { #2
      tvl <- c(1,curS[, "val"]^exp( data$cov[idata] * curbeta))
      tpvl <- c(1,head(tvl, n=-1))
      covS.time <- c(0,curS[, "time"])
      covS.inc <- tpvl - tvl
      subcovS.time <- covS.time[covS.time > T1_[idata] & covS.time <= T2_[idata]]
      subcovS.inc <- covS.inc[covS.time > T1_[idata] & covS.time <= T2_[idata]]
      cinc <- cumsum(subcovS.inc / sum(subcovS.inc))
      lcinc <- length(cinc)
      cmpmat <- matrix(runif(n = curmcint), nrow = curmcint, ncol = lcinc)

      ff <- function( m_, inc_, t_) {
        t_[which(m_ <= inc_ )[1]] }
      matmc[idata, ] <- apply(cmpmat, MARGIN=1, ff, inc_ = cinc, t_ = subcovS.time)
    } else  {
      matmc[idata, ] <- T1_[idata]
    }

    # if (idata==200) {print( matmc) ;stop()}
  }

  betalist <- NULL
```

```
  varlist <- NULL
  tempL0 <- NULL
  for ( imc in 1:curmcint) {
    pc <- coxph(Surv(matmc[, imc], C_) ~ cov, method = "efron")

    betalist <- c(betalist, pc$coefficients )
    varlist <- c(varlist, pc$var )
    bh <- basehaz(pc, centered = FALSE)
    tempL0 <- rbind(tempL0, matrix(c(rep(imc, length(bh$time)), bh$time, bh$hazard), ncol = 3))
    colnames(tempL0) <- c("mcint", "time", "val")
    lastitdata <- list(lstbetalist = betalist, lstL0 = tempL0, lstvar = varlist, m = curmcint)
    }

  curL0 <- L0list(tempL0)
  dares <- rbind(dares,  cbind(iter = rep(idagm, length(curL0[, "time"])), curL0))
  curS <- matrix(c(curL0[, "time"], exp(-curL0[, "val"])), ncol = 2)
  colnames(curS) <- c("time", "val")
  curS <- data.frame( time= curL0[, "time"], val= exp(-curL0[, "val"]))
  curbeta <- mean(betalist )
  beta <- c(beta,  curbeta)
  varbeta <- mean(lastitdata$lstvar) + (1+1/lastitdata$m)* var(lastitdata$lstbetalist)
  names(beta) <- NULL
  rbeta  <- c(rbeta, mean(tail(beta,n= rmeanl )))
  diffs <- ifelse(length(beta) > rmeanl, abs( tail( rbeta, n=1) - tail( rbeta, n=2)[1]),1)
  if (diffs < crit) { break }
}
res <- list( P = data.frame(curS[, "time"], 1 - curS[, "val"]), S = curS, iter = dares,
             betalist = beta, beta = tail(beta, n = 1), varbeta = varbeta, lastitdata = lastitdata)
}
# ================================================================================
intervalcens <- res$beta
intervalcens.CI.low <- res$beta - qnorm(0.975) * res$varbeta ^ 0.5
intervalcens.CI.up <- res$beta + qnorm(0.975) * res$varbeta ^ 0.5


# decide between the two output versions
if (CI == TRUE) {output <- list(intervalcens, intervalcens.CI.low, intervalcens.CI.up)
                 names(output) <- c("intervalcens", "intervalcens.CI.low", "intervalcens.CI.up")
} else {output <- intervalcens}
output
}
>
```

8.3 Additional results of survival function estimation

Results for scenarios whose RMSE and bias values are similar to Scenario 1 are not included in the Results Section 4.1. For the sake of completeness they are added here.

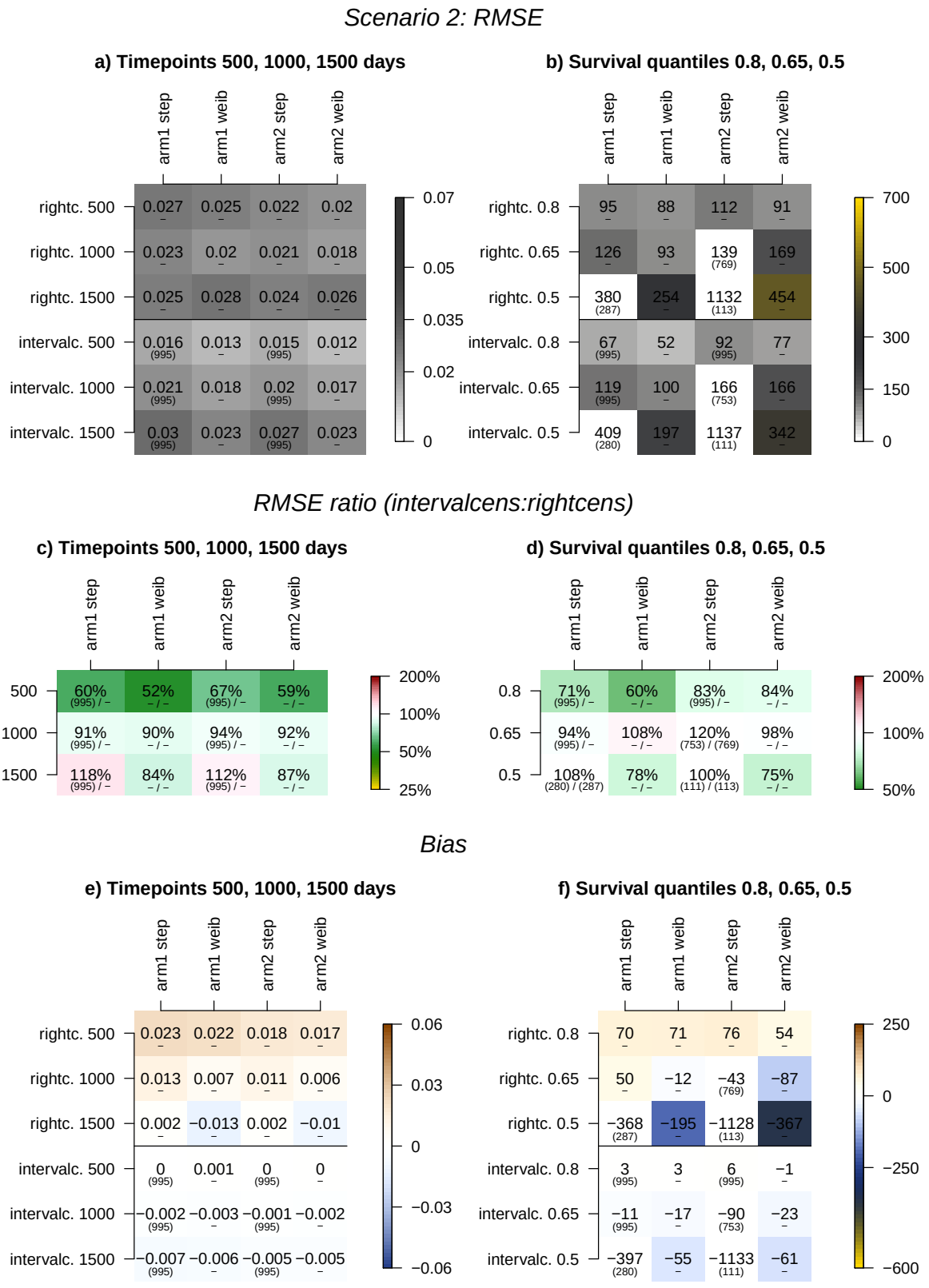

Figure 8.1: RMSE (a + b), RMSE ratio (c + d) and bias (e + f) estimates of 1000 datasets of Scenario 2 in colors. The estimates at timepoints 500, 1000 and 1500 days are shown in the left plots (a, c + e) and the survival quantiles 0.8, 0.65 and 0.5 in the right plots (b, d + f). The values are colorcoded as indicated in the legends. We differentiate both study arms and estimation variants on the x-axis, both censoring types and three timepoints or quantiles on the y-axis of the plots. Sometimes the NPMLE (step) functions were not defined at the readout quantile in all datasets. Then the number of datasets with a valid estimate is displayed in brackets below the RMSE value. Estimates from less than 900 datasets do not get the color coding because they are not completely reliable.

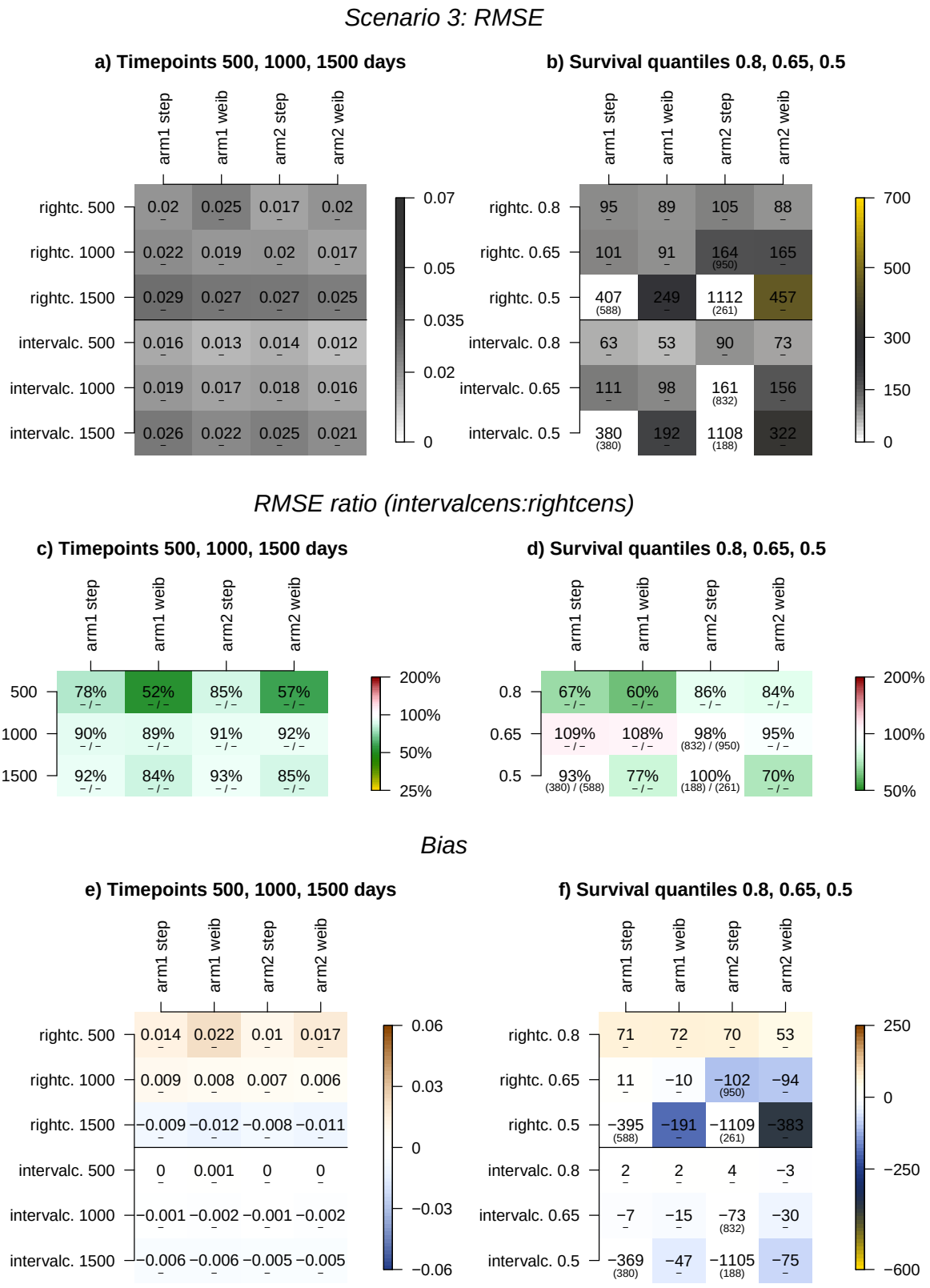

Figure 8.2: RMSE (a + b), RMSE ratio (c + d) and bias (e + f) estimates of 1000 datasets of Scenario 3 in colors. The estimates at timepoints 500, 1000 and 1500 days are shown in the left plots (a, c + e) and the survival quantiles 0.8, 0.65 and 0.5 in the right plots (b, d + f). The values are colorcoded as indicated in the legends. We differentiate both study arms and estimation variants on the x-axis, both censoring types and three timepoints or quantiles on the y-axis of the plots. Sometimes the NPMLE (step) functions were not defined at the readout quantile in all datasets. Then the number of datasets with a valid estimate is displayed in brackets below the RMSE value. Estimates from less than 900 datasets do not get the color coding because they are not completely reliable.

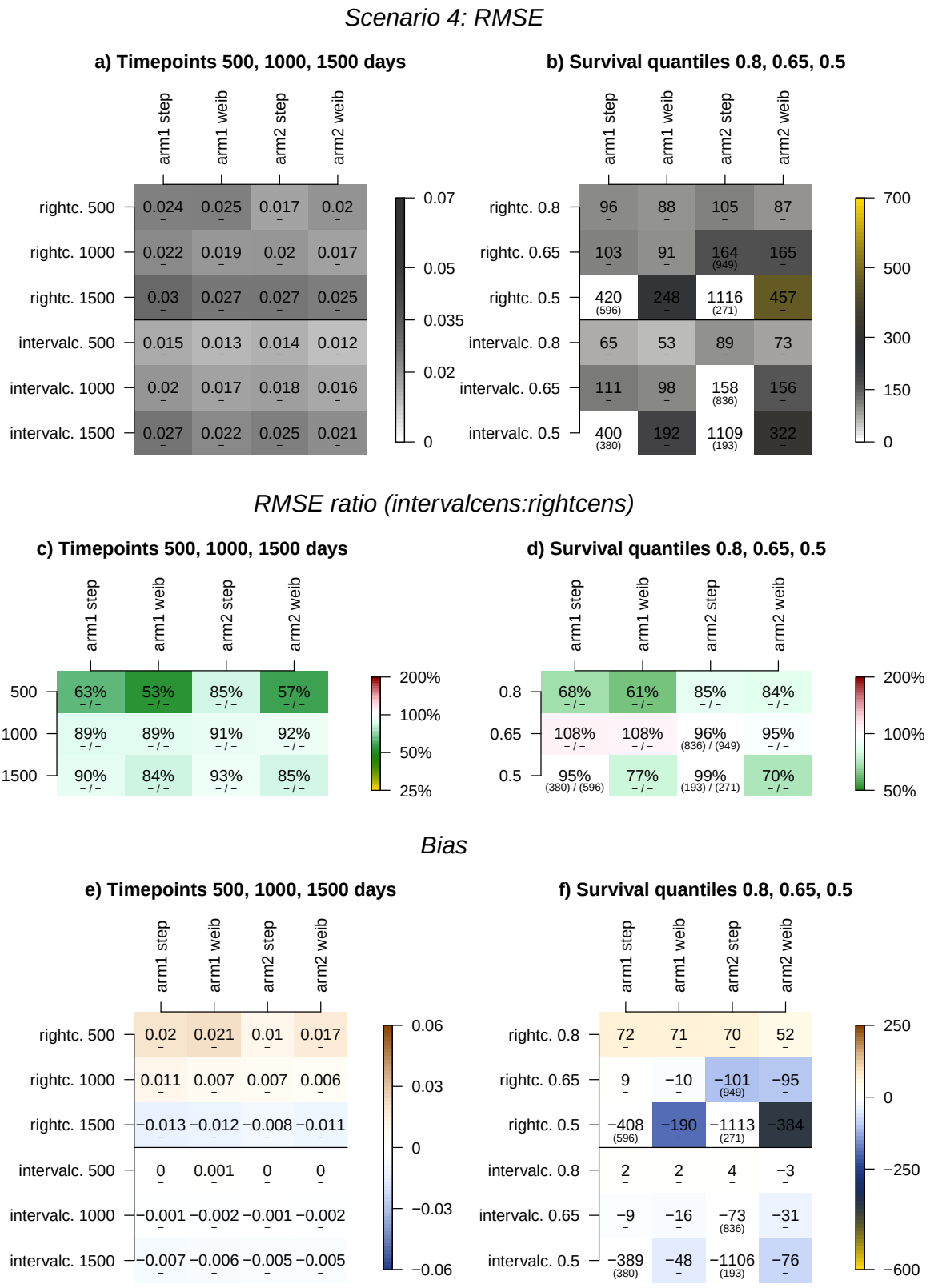

Figure 8.3: RMSE (a + b), RMSE ratio (c + d) and bias (e + f) estimates of 1000 datasets of Scenario 4 in colors. The estimates at timepoints 500, 1000 and 1500 days are shown in the left plots (a, c + e) and the survival quantiles 0.8, 0.65 and 0.5 in the right plots (b, d + f). The values are colorcoded as indicated in the legends. We differentiate both study arms and estimation variants on the x-axis, both censoring types and three timepoints or quantiles on the y-axis of the plots. Sometimes the NPMLE (step) functions were not defined at the readout quantile in all datasets. Then the number of datasets with a valid estimate is displayed in brackets below the RMSE value. Estimates from less than 900 datasets do not get the color coding because they are not completely reliable.

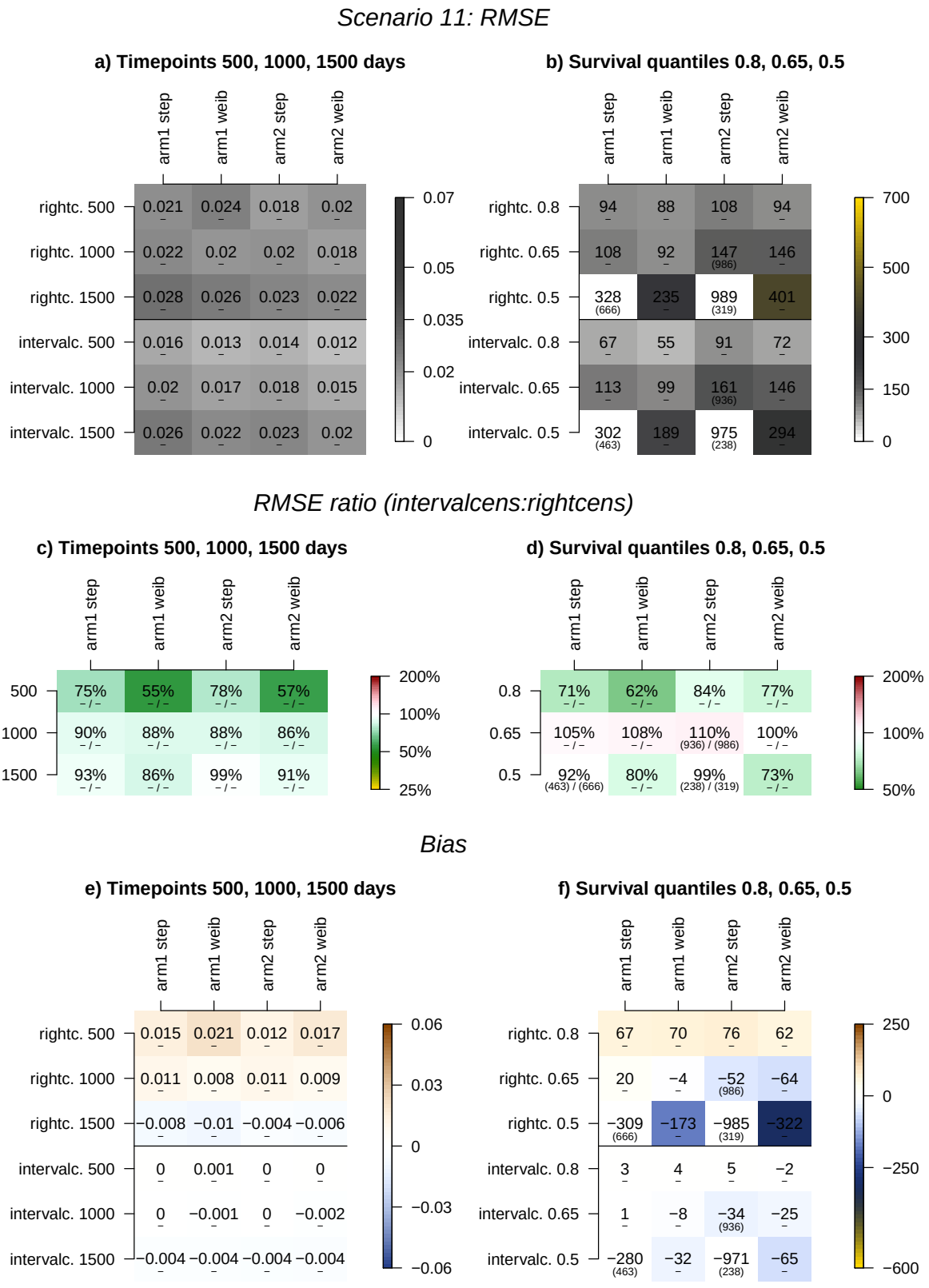

Figure 8.4: RMSE (a + b), RMSE ratio (c + d) and bias (e + f) estimates of 1000 datasets of Scenario 11 in colors. The estimates at timepoints 500, 1000 and 1500 days are shown in the left plots (a, c + e) and the survival quantiles 0.8, 0.65 and 0.5 in the right plots (b, d + f). The values are colorcoded as indicated in the legends. We differentiate both study arms and estimation variants on the x-axis, both censoring types and three timepoints or quantiles on the y-axis of the plots. Sometimes the NPMLE (step) functions were not defined at the readout quantile in all datasets. Then the number of datasets with a valid estimate is displayed in brackets below the RMSE value. Estimates from less than 900 datasets do not get the color coding because they are not completely reliable.