# Variable Selection and Model Choice in Survival Models with Time-Varying Effects

**Diplomarbeit**
von
Benjamin Hofner

**Betreuung**
Prof. Dr. Torsten Hothorn
Dr. Thomas Kneib


Ludwig-Maximilians-Universität München
Institut für Statistik

07. Juli 2008

# Acknowledgements

I specially want to thank ...

- **Torsten Hothorn** and **Thomas Kneib** for the excellent supervision and fruitful discussions that helped to enrich the presented work.
- **Wolfgang Hartl** for the data that lead to this diploma thesis and his help to describe and understand the data.
- **Helmut Küchenhoff** for his long time support and for the project that led to this diploma thesis.
- **Robert Hable** for his help with deriving the functional gradient.
- **Susann Bräunig** for proof-reading and lots of coffee breaks that helped me to refresh my mind.
- **Isabel Elsäßer** for proof-reading.
- **Bernd Fellinghauer** for fruitful discussions, proof-reading and a good time in Berlin and elsewhere. (His comment on the high computational effort of time-varying base-learners: "Somehow you give the word '*time*-varying' a totally new meaning...")
- **My family** for their assistance and that they (at least) try to understand what I am doing.
- **Amelie Elsäßer** for proof-reading, and especially her strong support and moral assistance over all the time.

# Contents

# 1. Introduction

Attempts to analyze censored data have a long tradition. Daniel Bernoulli, for example, tried to analyze the impact of vaccination against smallpox on mortality and morbidity already in 1766 (Bernoulli & Blower 2004, reprint with discussion). Nowadays, more advanced standard methods are broadly available. Classical survival models had a break-through with the well-known, omnipresent Cox model (Cox 1972).

In this thesis we want to derive new methods for the estimation of flexible Cox-type models. This class of models aims to fit censored survival data allowing for smooth and time-varying effects of covariates. We can obtain very flexible models in this class. However, additional difficulties arise when we try to perform variable selection or model choice. To overcome these problems we propose an estimation scheme based on component-wise boosting that allows estimation of the model and has a built-in variable selection approach (cf. Bühlmann & Hothorn 2007). By decomposing the effects into their linear and flexible components, component-wise boosting can also be utilized to choose an appropriate modeling alternative (Kneib et al. 2007).

The thesis is organized as follows: Chapter 2 gives some basic insights in survival models and presents some possible extensions to the Cox model. Chapter 3 goes into more technical details. It demonstrates how the ideas from Chapter 2 can technically be included in the model and how inference can be done. We introduce the concept of P-splines and sketch their application in survival models. Degrees of freedom for flexible survival models with P-splines are defined. The chapter ends with an approach for variable selection and model choice in classical estimation frameworks called two-stage stepwise procedure. In the subsequent chapters we will illustrate means to incorporate model choice and variable selection in the boosting framework, which is the main focus of this work. Therefore, we introduce the concept of functional gradient descent boosting and the related likelihood-based boosting in Chapter 4. The general approaches are illustrated by some special algorithms and the current state of boosting for survival models is presented. We will also give some theoretical and practical background on the choice of base-learners. Chapter 5 addresses the newly derived likelihood-based boosting algorithm $\text{Cox}_{\text{flex}}$Boost for flexible boosting in Cox-type additive models. The failure of functional gradient descent boosting in this model class is discussed and illustrated. Problems in the likelihood-based $\text{Cox}_{\text{flex}}$Boost algorithm are outlined and possible solutions are presented. A simulation study is

conducted in Chapter 6 to asses the performance of the proposed $\text{Cox}_{\text{flex}}$Boost algorithm. Furthermore, we use the $\text{Cox}_{\text{flex}}$Boost procedure to derive a prognostic model for surgical patients with severe sepsis. The results are compared with those of the two-stage stepwise procedure applied on the same data set. The appendix includes the derivation of the functional gradient for the full likelihood of survival models with time-varying effects. Some computational considerations that were made for the implementation of the boosting procedure are considered and a demonstration of the usage of the implemented software is given. Finally, additional figures for the simulation study in Chapter 6 are presented.

# 2. Survival Models

Survival models or, more generally speaking, time-to-event models have got a broad range of applications. They can be useful, for example, in quality control (e.g., time until failure of a machine) or social sciences (e.g., time until marriage). However, the most important area of application is medical research. Examples here are death caused by a special illness, death after surgery, or time until relapse of a tumor. Death is, after healing, one of the most important and drastic outcomes in medicine. Hence, time until death is often used as a measure for therapeutic progress or for determination of risk factors for patients.

As we focus on medical applications (see Section 3.3 and Chapter 6), in the following we will refer to time-to-event models as survival models without restricting the methodological results to clinical or epidemiological research.

A typical problem for survival data is censoring, i.e., the event of interest (e.g., death caused by a special illness) is not observed until the end of the study period for some individuals. This can be due to drop-outs, competing events (e.g., death caused by something else) or just the end of the study.

In general it is supposed that the censoring time $C$ is independent of the true survival time $T^*$ (i.e., censoring at random). We denote the observed survival time with $T = \min(T^*, C)$; $t$ represents the corresponding realization. In addition an indicator for non-censoring

$$\delta = I_{\{T^* \leq C\}} = \begin{cases} 1 & \text{if } T^* \leq C \\ 0 & \text{if } T^* > C \end{cases}$$

is introduced. The pair of observed survival time and non-censoring indicator

$$y_i = (t_i, \delta_i), \quad i = 1, \ldots, n. \tag{2.1}$$

is then regarded as observed outcome for the $i$-th observation.

## 2.1. Definitions and Interrelations

The (true) survival time $T^*$, which is greater or equal to zero, can be seen as a random variable and thus be described in terms of the distribution function

$F_{T^*}(t) = P(T^* \leq t)$, the density $f_{T^*}(t) = F'_{T^*}(t)$ or the survival function

$$S_{T^*}(t) = P(T^* \geq t) = 1 - F_{T^*}(t). \tag{2.2}$$

Another well known quantity to determine the distribution of survival times is the hazard rate

$$\lambda_{T^*}(t) := \lim_{\Delta t \to 0} \frac{1}{\Delta t} P(t \leq T^* < t + \Delta t | T^* \geq t). \tag{2.3}$$

It describes the infinitesimal probability of an event (e.g., death) given that the observed individual is under surveillance until time $t$ (e.g., the individual survived until $t$). The cumulative hazard rate

$$\Lambda_{T^*}(t) = \int_0^t \lambda_{T^*}(u) du \tag{2.4}$$

is just the integral over time of the hazard rate. All defined functions are directly linked to each other and uniquely determine the distribution of $T^*$. Thus, it is sufficient to specify one of them. One important connection should be mentioned here: One can show that

$$S_{T^*}(t) = \exp\{-\Lambda_{T^*}(t)\} \tag{2.5}$$

which we will need later on (see Sec. 6.1.1). The links of the other functions can be found, for instance, in Klein & Moeschberger (2003). For simplicity, the index $T^*$ is dropped for all quantities, in the following.

## 2.2. Cox PH Model

The most commonly used model for survival time data, and other data with censored outcomes, is the Cox proportional hazards model (Cox 1972). It specifies the hazard rate (2.3) as a semi-parametric model

$$\lambda_i(t) = \lambda(t, x_i) = \lambda_0(t) \exp\left(\sum_{j=1}^{p} x_{ij}\beta_j\right) = \lambda_0(t) \exp\left(\boldsymbol{x}_i'\boldsymbol{\beta}\right), \tag{2.6}$$

where $\lambda_0(t)$ is the baseline hazard rate, $\boldsymbol{x}_i$ is a $p \times 1$ vector of covariates for observation $i$ ($i = 1, \ldots, n$) and $\boldsymbol{\beta}$ is the corresponding $p \times 1$ coefficient vector. Note that in the Cox model $\boldsymbol{\beta}$ and $\boldsymbol{x}_i$ do not have a term for the intercept. The intercept is implicitly included in the baseline hazard $\lambda_0(t)$. An additional term in the parametric part of the model would make the Cox model unidentifiable.

The model is called semi-parametric, as only the effects of the covariates are parameterized, whereas the baseline hazard rate remains unspecified and is usually estimated in a nonparametric way using the Breslow estimator (Breslow 1974).

The key assumption, from which the Cox PH model derived its name, is the proportional hazards property (PH property): With time-constant covariates the hazard ratio

$$\frac{\lambda(t, \boldsymbol{x}_i)}{\lambda(t, \boldsymbol{x}_j)} = \frac{\lambda_0(t) \exp(\boldsymbol{x}_i' \boldsymbol{\beta})}{\lambda_0(t) \exp(\boldsymbol{x}_j' \boldsymbol{\beta})} = \exp\left[(\boldsymbol{x}_i - \boldsymbol{x}_j)' \boldsymbol{\beta}\right] \tag{2.7}$$

is always constant over time and thus the hazard rates are proportional. The PH property is induced by the strict separation of time-dependent effects in the baseline hazard rate and time-constant covariate effects in the second factor of (2.6), the exponential function.

### 2.2.1. Interpretation

One of the big advantages of the Cox model is the easy interpretation of the coefficients $\beta_j$, or rather, $\exp(\beta_j)$. In the case of a binary covariate $x_j$, the latter can be seen as hazard ratio when the other covariates are fixed. The hazard ratio for an increase of 1 for only one covariate $x_j$, given that the other covariates are fixed, is

$$\frac{\lambda(t|x_j = x+1, \boldsymbol{x}_{-j})}{\lambda(t|x_j = x, \boldsymbol{x}_{-j})} \overset{(2.7)}{=} \exp\left\{[(x+1) - x]\beta_j\right\} = \exp(\beta_j), \tag{2.8}$$

where $x_j$ is the $j$-th covariate and $\boldsymbol{x}_{-j}$ are the predictor variables without covariate $j$. All covariates except $x_j$ are kept fix for the interpretation of $\beta_j$. With $x = 0$, which leads to comparison of $x_j = 1$ with $x_j = 0$, Equation (2.8) can be seen as the special case for binary covariates $x_j$.

More generally, $\exp(\beta_j)$ can be interpreted as multiplicative factor on the baseline hazard rate $\lambda_0(t)$, where for $\beta_j > 0$ an increase of the corresponding $x$-value leads to an increased hazard rate which corresponds to a decreased mean survival time and analogously for $\beta_j < 0$ to an increased mean survival time (given that the other covariates are fixed).

The covariates form a log-linear model for the hazard rate, that means for a continuous variable that for a distance of 10, e.g., from 1 to 11 or from 101 to 111 the same hazard ratio is assumed. This can, in some applications, be very reasonable, whereas in others this might be very misleading. For example age very often has got threshold values where the risk structure changes (cf. Therneau & Grambsch 2000). Hence, modeling age in a non-linear parametric or even non-parametric way circumvents the log-*linearity* and might lead to improved models.

### 2.2.2. Violation of PH Assumption

As research has shown, the Cox model is very sensible against deviations from the PH assumption (see Bender et al. 2005). However, if the PH assumption holds, it leads to a model, which is easy to interpret and can be fitted easily in every standard statistical software package.

Nowadays, more sophisticated modeling possibilities are becoming widely available. For example, more and more software packages provide the opportunity to specify Cox models with time-varying coefficients (see Sec. 2.3.2). However, one needs to be careful with adding time-varying coefficients to "heal" non-proportional hazards, as there are different causes for non-proportionality in the Cox model (see Therneau & Grambsch 2000) including:

- omission of (important) covariates
- incorrect functional form of covariates
- incorrect model; other models would be more appropriate (e.g., accelerated failure time (AFT) models or additive hazard models).

In all these cases, using a Cox model can lead to a model, where the proportional hazards assumption is violated. Thus possibly meaningful covariates should be chosen with care. Very often, the Cox model is applied just for convenience and habit. Here as well, researchers should check if this is the best alternative. Keeping these problems in mind, ways to determine the functional form and to detect and model time-varying effects are needed.

## 2.3. Extensions to Cox PH Models

To overcome the restrictions of the Cox model as discussed in Section 2.2 some relaxations are needed. Firstly, one might want to include non-linear effects to the model to relax the log-linear form of covariate effects. Secondly, the proportional hazards assumption might not hold. Hence, time-varying effects should be used. How to include these concepts in the Cox model is discussed in the following. Technical details will be given in Chapter 3.

### 2.3.1. Nonparametric Effects

As stated in Section 2.2.1, the covariate effects in the Cox model form a log-linear model for the hazard rate. In many circumstances this might not hold. Therefore, transformations of the variables are necessary. Classically this can be solved by examining a scatterplot of the martingale residuals of the null

model (i.e., with $\hat{\boldsymbol{\beta}} = \mathbf{0}$) plotted against each covariate together with a scatterplot smoother (Therneau et al. 1990). From the smoother one can derive the functional form under some assumptions by visual inspection. As a result transformed covariates (e.g., $x^2$ instead of $x$) can be used or cut-points might be introduced.

More advanced approaches use smooth functions $f(v)$ of a covariates. This can be done with any classical smoother, such as smoothing splines, regression splines or P-splines (see, e.g., Therneau & Grambsch 2000). These methods have the advantage that no visual inspection is needed to determine the functional form. Moreover, no fixed transformation is needed. The functional form is chosen adaptively from the data. The P-spline approach will be used in this thesis and is explained in Section 3.1.1. Despite being very flexible, it is still feasible to estimate the induced model.

Another approach utilizes fractional polynomials (Royston & Altman 1994) for flexible modeling of continuous covariates. A fractional polynomial of degree $m$ is defined for positive valued covariates $x > 0$ as

$$\Phi(x; \boldsymbol{\beta}, \boldsymbol{p}) = \beta_0 + \sum_{j=1}^{m} \beta_j x^{p_j}, \tag{2.9}$$

where $\boldsymbol{p} = (p_1, \ldots, p_m)$ is a vector of powers with $p_1 < \ldots < p_m$ and $\boldsymbol{\beta} = (\beta_0, \ldots, \beta_m)'$ is a vector of coefficients. Royston & Altman (1994) propose to choose $\boldsymbol{p}$ from a fixed set of candidate values $\mathcal{P} = \{-2, -1, -0.5, 0, 0.5, 1, 2, \ldots, \max(3, m)\}$, where a redefinition for the exponent 0 is applied: $x^0 := \log x$. Variables with negative values require a shift such that non-negativity is guaranteed. Despite the flexibility, especially for fractional polynomials with a high degree $m$, a major drawback of this approach remains: The approach strongly depends on the modeling alternatives given by $\mathcal{P}$, which are entered in the fractional polynomial formulation.

### 2.3.2. Time-Varying Effects

Time-varying effects and the concern about them have a long tradition, as the PH assumptions can be seen as the strongest assumption in the Cox model. Cox (1972) proposed inclusion of time-varying effects $g(t)$ using artificial time-dependent covariates. He proposed to use $g(t) = \alpha \cdot \log(t)$ as time-dependent effect and model the interaction of $g(t)$ and covariate $u$, which is assumed to have a time-varying effect:

$$\alpha \cdot \log(t) \cdot u = \alpha \cdot \tilde{u}(t), \tag{2.10}$$

where $\tilde{u}(t)$ is now a known time-dependent covariate.

Again the fractional polynomial approach can be applied to model the functions more flexibly than just applying one pre-specified transformation. Sauerbrei et al. (2007) describe an extension of a multivariable fractional polynomial approach with a special focus on detecting time-varying effects. First, a model with time-fixed effects for the full time period is determined based on the classical multivariate fractional polynomial approach. Then the time period is restricted to allow short-term effects to be entered in the model if they are significant. This model serves as starting model for the next step. For each selected covariate, together with the selected fractional polynomial, time-varying effects, based on fractional polynomials of time, are added in a forward selection procedure if they are significant. Thus we have a strategy in several steps: First, variables with possible influence are selected. Second, for continuous covariates the functional form is determined. Third, time-varying effects are analyzed to allow departures from the PH assumption. The drawbacks remain the same as in Section 2.3.1, i.e., the procedure highly depends on the possible fractional polynomials.

Applying the artificial covariate approach is not restricted to known functions as in Cox's (1972) proposal or parametric functions as in the multivariable fractional polynomial approach for time-varying effects. Being a function of time, $g(t)$ can be flexibly modeled using again the same smoothing techniques as for nonparametric effects, as for example, smoothing splines, regression splines and P-splines. Thus, one gets nonparametric and flexible time-varying effects. Again, P-splines will be our choice to model $g(t)$ in the following.

**Cox-Type Additive Model** Including both, time-varying and smooth effects in the model, we get a Cox-type additive model as introduced by Kneib & Fahrmeir (2007). The hazard rate then can be written as

$$\lambda_i(t) = \exp(\eta_i(t)) \tag{2.11}$$

with the additive predictor

$$\eta_i(t) = g_0(t) + \sum_{l=1}^{L} g_l(t)u_{il} + \sum_{j=1}^{J} f_j(v_{ij}) + \boldsymbol{x}_i'\boldsymbol{\beta}\,, \tag{2.12}$$

where

- $g_0(t)$ is the log-baseline hazard,
- $g_l(t)$ are the time-varying effects of covariates $u_{il}$ , $l = 1, \ldots, L$,
- $f_j(v_{ij})$ are the smooth effects of covariates $v_{ij}$ , $j = 1, \ldots, J$, and
- $\boldsymbol{x}_i'\boldsymbol{\beta}$ are linear effects as in the classical Cox model (2.6).

As we include the log-baseline hazard $g_0 = \log(\lambda_0(t))$ in the additive predictor (2.12) there is no need to additionally specify the classical baseline hazard $\lambda_0(t)$

in (2.11). How to model and especially estimate the functional forms $g_l(\cdot)$, $l = 0, \ldots, L$, and $f_j(\cdot)$, $j = 1, \ldots, J$, in (2.12) is discussed in the following chapter.

# 3. Estimation in Survival Models

This chapter deals with the technical backgrounds of Cox-type additive models. In Section 3.1.1 we introduce the concept of P-splines with a special focus on survival models and derive their degrees of freedom in this context. Then we give some insights in the estimation procedures (Sec. 3.2) applied for the classical Cox model and the flexible Cox-type additive model, as specified in Section 3.1.3. In the last part of this chapter (Sec. 3.3) we discuss the issues of variable selection and model choice in flexible survival models. A two-stage stepwise procedure is introduced, which tries to solve this problem for survival models in a classical estimation framework. More on model choice and variable selection for flexible survival models can be found in Chapter 5, where a boosting algorithm called $\text{Cox}_{\text{flex}}$Boost is derived. This approach allows to combine estimation of the model with variable selection and model choice.

## 3.1. Cox-type Additive Models

We already defined the Cox-type additive model in Section 2.3 in a basic way. Now, possibilities to model and to estimate the time-varying effects, expressed as functions $g_l(t)$, and the smooth effects, denoted by functions $f_j(v_{ij})$, are introduced.

Both functions can be modeled as splines of covariates $t$ or $v_{ij}$, respectively. Modeling time-varying effects as product of a spline function $g_l(t)$ and a covariate $\boldsymbol{u}_l$ (see Gray 1992) provides a flexible extension to the classical approach proposed by Cox (cf. Sec. 2.3.2).

### 3.1.1. P-Splines

Classical modeling approaches for flexible models apply smoothing splines or regression splines (Gray 1992) to introduce non-linear functions. The latter can, for example, be expressed based on truncated power series basis (see e.g., Fahrmeir & Tutz 2001). A numerically superior alternative are B-splines (de Boor 1978, Dierckx 1993). An introduction to B-splines can be found in Eilers & Marx (1996), which builds the basis of the following section.

B-splines consist of polynomial pieces of degree $q$, which are specified between given knots. They are constructed such that they are connected in each knot in a special way. A B-spline of degree one, for example, consists of two linear pieces (i.e., polynomials of degree one) joint at one knot (see Figure 3.1). B-splines of degree two consist of three quadratic pieces joint at two knots. They have a support of four knots (see Figure 3.1).

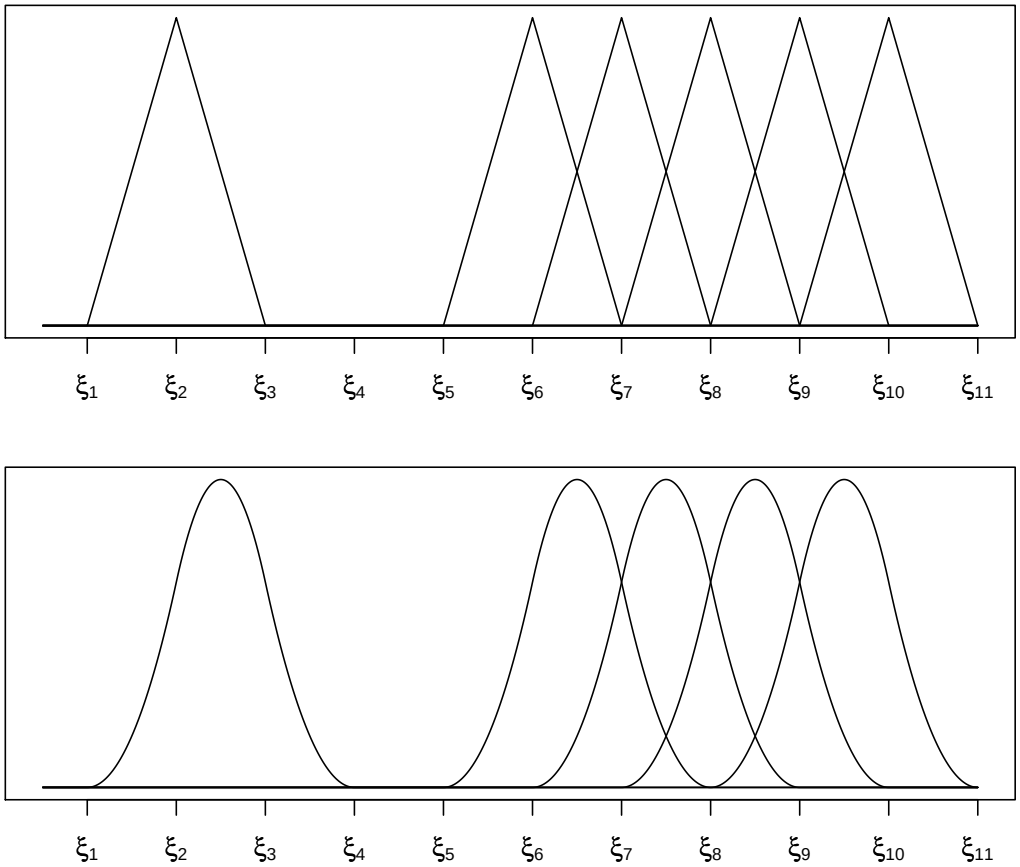


Figure 3.1.: B-splines of degree one (upper panel) and of degree two (lower panel); in each panel one isolated B-spline is depicted.

The general properties of a B-spline basis function of degree $q$ can be summarized as follows:

- it consists of $q + 1$ polynomial pieces of degree $q$;
- the polynomial pieces are joint at $q$ inner knots;
- derivatives up to order $q - 1$ are continuous at the joining points;
- one B-spline basis function is positive on a domain spanned by $q+2$ knots,

elsewhere it is zero;

- it overlaps with $2q$ polynomial pieces of its neighbors, except at the boundaries;
- at a given $x$, $q+1$ B-splines are non-zero.

A B-spline with degree zero is a constant (greater than zero) on an interval between two neighboring knots and zero elsewhere. B-splines of higher degrees can be computed recursively (based on lower degree B-splines) following a relatively simple algorithm (de Boor 1978). For equidistant knots, as used in the following, the algorithm can be further simplified. In the statistical software package `R` (R Development Core Team 2007) the function `bs()` for calculating B-Spline basis functions can be found in the package `splines`.

In the following, let $B_k(x;q)$ denote the $k$-th B-spline basis function for a (given) degree $q$ at value $x$. For simplicity the degree $q$ from the B-spline notation is dropped if not needed. Let the range $x_{\min}$ to $x_{\max}$ be divided into intervals by $s$ equidistant knots. To ensure that each interval in the range of data is covered by $q+1$ B-spline basis functions of degree $q$, one needs $2q$ additional boundary knots, $q$ on each side of the interval. Thus, one has $s+2q$ knots with altogether $K = s+q-1$ B-splines. A mathematical function can then be expressed as

$$f(x) = \sum_{k=1}^{K} \beta_k B_k(x) \tag{3.1}$$

with knots $\xi_1 < \ldots < \xi_K$. The index $k$ connects a spline basis to a knot. Here we define $B_k$ to start at knot $\xi_k$. For example, the leftmost B-spline basis function in both panels in Figure 3.1 is defined as $B_1$. From (3.1) it follows that the function $f$ is approximated as a weighted sum of known functions. Thus, $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_K)'$ can be regarded as a normal parameter vector in a regression problem with the design matrix

$$\boldsymbol{B} = (B_1(\boldsymbol{x}), \ldots, B_K(\boldsymbol{x})), \tag{3.2}$$

and column vectors $B_k(\boldsymbol{x}) = (B_k(x_1), \ldots, B_k(x_n))',\ k = 1, \ldots, K$.

The basis functions strongly depend on the number and location of knots and thus, the resulting function $f$ as well. A lot of research on the impact of the number (and positioning) of knots has been done. Especially, if you take to many knots overfitting may occur (Eilers & Marx 1996). Friedman & Silverman (1989) and Kooperberg & Stone (1992), for example, proposed automated schemes for choosing the knots. Nevertheless, these are numerically demanding and no good scheme for all purposes exists.

Another solution to prevent overfitting is penalization. Therefore we choose a relatively large number of knots and penalize too flexible models. P-splines

(Eilers & Marx 1996) build a computationally very effective way of specifying the penalty. To control smoothness, the differences of coefficients of adjacent B-splines are penalized. Classical penalties are based on first or second order differences. Using first order differences

$$\Delta\beta_k = \beta_k - \beta_{k-1} \tag{3.3}$$

can be seen as penalizing deviation from a constant fit. For second order differences

$$\Delta^2\beta_k = \Delta(\Delta\beta_k) = \beta_k - 2\beta_{k-1} + \beta_{k-2} \tag{3.4}$$

deviations from a linear fit are penalized. Higher order penalties $\Delta^d$ can be constructed and interpreted analogously. One can show that for a penalty of order $d$, in the limit ($\lambda \to \infty$), a $d-1$ polynomial remains unpenalized if the degree of the B-splines is greater or equal to $d$ (Eilers & Marx 1996). In the following, we will only apply second order difference penalties but the results are not restricted to these.

We get the P-spline penalty term for differences of order $d$ as

$$\text{pen}^{(d)}(\boldsymbol{\beta}) = \frac{\lambda}{2}\sum_{k=d+1}^{K}(\Delta^d\beta_k)^2, \tag{3.5}$$

where $\lambda$ is the smoothing parameter, determining the amount of smoothness. The larger $\lambda$ gets, the more smoothing is achieved. For $\lambda = 0$ the penalty completely vanishes; hence, no smoothing is performed. The penalty (3.5) can be written in matrix notation as

$$\text{pen}^{(d)}(\boldsymbol{\beta}) = \frac{\lambda}{2}\boldsymbol{\beta}'\boldsymbol{K}^{(d)}\boldsymbol{\beta} \tag{3.6}$$

with $\boldsymbol{K}^{(d)}$ being the cross product of the difference matrix that means, $\boldsymbol{K}^{(d)} = (\boldsymbol{D}^{(d)})'\boldsymbol{D}^{(d)}$. For example, for second order differences we get

$$\boldsymbol{D}^{(2)} = \begin{pmatrix} 1 & -2 & 1 & 0 & \dots & 0 \\ 0 & 1 & -2 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & 1 & -2 & 1 \end{pmatrix}. \tag{3.7}$$

Inference is then based on penalized least squares estimation or in a more general case on the penalized likelihood criterion. For survival data, the latter case is relevant. The penalized full likelihood and further details on estimation in survival models are discussed in Section 3.2.

As stated above, the number of knots should be relatively large when using penalized splines. However, if we choose the degrees of freedom (df) to be small,

we do not loose much flexibility when the number of knots is also small. Gray (1992) suggest that there is not much gain of using more than 10 to 20 knots. So with 3 degrees of freedom they use 10 knots in their paper. Gray (1992) points out that due to the sparseness resulting from censoring, estimations of time-varying effects tend to be unstable in the right tail.

Of course, in a model with more smooth effects, for each of them a separate B-spline representation (3.1) with according penalty (3.6) is defined. The number and placement of knots may potentially be different for each representation as well as each penalty function may be different. The smoothing parameter $\lambda$ is a very natural quantity to adopt as one might want different degrees of smoothing for different covariates. Furthermore, the matrix $K$ depends again on the number of knots and one could also use difference matrices with other difference orders $d$.

### 3.1.2. Degrees of Freedom

Defining degrees of freedom df instead of a smoothing parameter $\lambda$ for flexible terms is by far more intuitive. Furthermore, one can specify df such that the resulting spline is comparable to other splines or even classical parametric terms (compare Section 4.3.4).

A definition for the equivalent degrees of freedom of P-splines in survival models is given by Gray (1992). Equivalent degrees of freedom represent the *effective* number of parameters in the model and replace the classical parameter count since the penalty reduces the flexibility given by the number of parameters and thus reduces the degrees of freedom of the resulting model. Gray (1992) derives the degrees of freedom for the construction of tests for general linear hypotheses $\boldsymbol{C}\boldsymbol{\beta} = \mathbf{0}$, where $\boldsymbol{C}$ has full rank. For the global null hypothesis, $\boldsymbol{\beta} = \mathbf{0}$, $\boldsymbol{C}$ is the identity matrix and we get the overall model degrees of freedom:

**Definition 3.1** *Let $\boldsymbol{F}$ be the observed Fisher matrix and $\boldsymbol{F}_{\text{pen}}$ be the observed, penalized Fisher matrix. The degrees of freedom for a flexible survival model* (2.11) *with additive predictor* (2.12) *then can be defined as*

$$\text{df} := \text{trace}\left[\boldsymbol{F} \cdot \boldsymbol{F}_{\text{pen}}^{-1}\right]. \tag{3.8}$$

The definitions for the observed Fisher matrix and the observed, penalized Fisher matrix are derived in Section 3.2. Note that the model degrees of freedom as defined here are slightly different from those defined by Hastie & Tibshirani (1990a), where $\text{df} := \text{trace}(2\boldsymbol{S} - \boldsymbol{S}^2)$ with smoother matrix $\boldsymbol{S}$. However, correspondence can be seen with the definition of degrees of freedom in Buja et al. (1989), where $\text{df} := \text{trace}(\boldsymbol{S})$. The latter is the popular definition from the spline smoothing literature (e.g., Silverman 1985).

As $\boldsymbol{F}_{\text{pen}}$ can be regarded as a function of the smoothing parameter $\lambda$, the degrees of freedom can also be writen as a function of $\lambda$:

$$\text{df}(\lambda) = \text{trace}\left[\boldsymbol{F} \cdot \boldsymbol{F}_{\text{pen}}^{-1}(\lambda)\right] = \text{trace}\left[\boldsymbol{F}(\boldsymbol{F} + \lambda\boldsymbol{K})^{-1}\right]. \tag{3.9}$$

This shows the clear link between the smoothing parameter $\lambda$ and the degrees of freedom.

The above definition of df can also be motivated by comparison with the degrees of freedom for generalized linear models (GLMs). Here the degrees of freedom are defined as

$$\begin{aligned} \text{df} = \text{trace}(\boldsymbol{H}) &= \text{trace}[\boldsymbol{X}(\boldsymbol{X}'\boldsymbol{W}\boldsymbol{X} + \lambda\boldsymbol{K})^{-1}\boldsymbol{X}'\boldsymbol{W}] = \\ &= \text{trace}[\underbrace{\boldsymbol{X}'\boldsymbol{W}\boldsymbol{X}}_{\boldsymbol{F}}(\underbrace{\boldsymbol{X}'\boldsymbol{W}\boldsymbol{X} + \lambda\boldsymbol{K}}_{\boldsymbol{F}_{\text{pen}}})^{-1}] \end{aligned} \tag{3.10}$$

with the hat matrix $\boldsymbol{H}$, the design matrix $\boldsymbol{X}$, the weight matrix $\boldsymbol{W}$, and $\lambda$ and $\boldsymbol{K}$ as before. Thus, we see the same structure for the degrees of freedom as in (3.8). With $\lambda = 0$, i.e., without penalty, we get the usual definition of degrees of freedom $\text{df} = \text{trace}(\boldsymbol{I}) = \text{rank}(\boldsymbol{X}) = p$.

### 3.1.3. Model Specification

Using P-splines in survival models is not restricted to the parametric part of the Cox model (2.6) but one can apply them to model the baseline hazard rate and time-varying effects. This leads to Cox-type additive models (Kneib & Fahrmeir 2007) as already introduced in Section 2.3

$$\lambda_i(t) = \exp\left(g_0(t) + \sum_{l=1}^{L} g_l(t)u_{il} + \sum_{j=1}^{J} f_j(v_{ij}) + \boldsymbol{x}_i'\boldsymbol{\beta}\right).$$

Having a P-spline representation for the baseline hazard $\exp(g_0(t))$, the full likelihood is available and thus can be used for inference.

## 3.2. Likelihood Inference

Classical estimation in Cox models is based on the partial likelihood (Cox 1972). Therneau & Grambsch (2000) give a good overview of the inference in this context. They state that a partial likelihood is generally not a likelihood in a strong sense of being proportional to the probability of an observed data set. Nonetheless, it can be used for likelihood-based inference.

As mentioned above, we have got the full likelihood available and therefore do not need to use the partial likelihood. In the following, let $\boldsymbol{\eta}_i = \log(\lambda_i(t)) =$

$\widetilde{\boldsymbol{x}}_i'\boldsymbol{\beta}$ denote the linear predictor, including flexible terms expressed as B-spline bases (Dierckx 1993), as already discussed in Section 3.1.1. The column vector $\widetilde{\boldsymbol{x}}_i$, $i = 1, \ldots, n$ contains the original covariates $\boldsymbol{x}_i$ for the $i$-th observation or transformations of the covariates (e.g., B-spline basis functions computed for covariate $x_{ij}$). In matrix notation this can be written as $\boldsymbol{\eta} = \boldsymbol{X}\boldsymbol{\beta}$, where $\boldsymbol{X} = (\widetilde{\boldsymbol{x}}_1, \ldots, \widetilde{\boldsymbol{x}}_n)'$ is the design matrix for $n$ observations. We recall that $\lambda_i(t) = \exp(\eta_i(t))$ (see Sec. 2.3). The observed survival time $t_i$ and the indicator for non-censoring $\delta_i$ are defined as in Section 2.1.

The (unpenalized) log-likelihood can be expressed as

$$l(\boldsymbol{\beta}) = \sum_{i=1}^{n}(\delta_i\eta_i - \int_0^{t_i}\lambda_i(t)dt) = \boldsymbol{\delta}'\boldsymbol{\eta} - \mathbb{1}'\boldsymbol{\Lambda} \tag{3.11}$$

where $\boldsymbol{\Lambda} = (\Lambda_1(t_1), \ldots, \Lambda_n(t_n))'$ is a vector of the cumulative hazard rates $\Lambda_i(t_i) = \int_0^{t_i}\lambda_i(t)dt$. The score vector is given by the first derivative of the log-likelihood $l(\boldsymbol{\beta})$

$$\boldsymbol{s}(\boldsymbol{\beta}) = \frac{\partial}{\partial\boldsymbol{\beta}}l(\boldsymbol{\beta}) = \boldsymbol{\delta}'\boldsymbol{X} - \sum_{i=1}^{n}\int_0^{t_i}\boldsymbol{x}_i(u)\lambda_i(u)du, \tag{3.12}$$

where $\boldsymbol{x}_i(u)$ depicts that $\boldsymbol{x}_i$ *may* contain time-depending covariates. This can, for example, be the case when time-varying effects are used, as these are expressed as artificial time-dependent covariates (cf. Sec. 2.3.2). The observed Fisher matrix is then calculated as the negative second derivative of the log-likelihood:

$$\boldsymbol{F}(\boldsymbol{\beta}) = -\frac{\partial^2}{\partial\boldsymbol{\beta}\partial\boldsymbol{\beta}'}l(\boldsymbol{\beta}) = \sum_{i=1}^{n}\int_0^{t_i}\boldsymbol{x}_i(u)\boldsymbol{x}_i'(u)\lambda_i(u)du. \tag{3.13}$$

The corresponding penalized counterparts for Cox-type additive models (2.12) with P-splines are derived in the following. Let the smoothing parameters of the smooth functions $g_l(t), l = 0, \ldots, L$ and $f_j(v_j), j = 1, \ldots, J$ be denoted by $\lambda_l, l = 0, \ldots, L + J$, where $\lambda_l$ is possibly different for each function. The cross products $\boldsymbol{K}_l, l = 0, \ldots, L + J$ of the difference matrices correspond to those in (3.6). The parameter vector $\boldsymbol{\beta} = (\boldsymbol{\beta}'_{\text{pen},0}, \ldots, \boldsymbol{\beta}'_{\text{pen},L+J}, \boldsymbol{\beta}'_{\text{unpen}})'$ forms a column vector that consist of the penalized and unpenalized coefficients, where each of the penalized coefficient vectors $\boldsymbol{\beta}'_{\text{pen},l}$ is a vector of coefficients as in (3.1). The penalized log-likelihood then can be written as the difference of the log-likelihood $l(\boldsymbol{\beta})$ and the penalty

$$\begin{aligned} l_{\text{pen}}(\boldsymbol{\beta}) &= l(\boldsymbol{\beta}) - \sum_{l=0}^{L+J}\frac{\lambda_l}{2}\boldsymbol{\beta}'_{\text{pen},l}\boldsymbol{K}_l\boldsymbol{\beta}_{\text{pen},l} \\ &= l(\boldsymbol{\beta}) - \frac{1}{2}\boldsymbol{\beta}'\mathcal{K}\boldsymbol{\beta}, \end{aligned} \tag{3.14}$$

where $\mathcal{K} = \text{diag}(\lambda_0 \boldsymbol{K}_0, \ldots, \lambda_{L+J} \boldsymbol{K}_{L+J}, \boldsymbol{0}, \ldots, \boldsymbol{0})$ is a block diagonal matrix representing the penalization. Note that parameters for linear effects remain unpenalized in the model, only coefficients corresponding to smooth terms are penalized. The penalized score function $\boldsymbol{s}_{\text{pen}}$ is derived as

$$\boldsymbol{s}_{\text{pen}}(\boldsymbol{\beta}) = \frac{\partial}{\partial \boldsymbol{\beta}} l_{\text{pen}}(\boldsymbol{\beta}) = \boldsymbol{s}(\boldsymbol{\beta}) - \mathcal{K}_l \boldsymbol{\beta} \tag{3.15}$$

and the penalized Fisher matrix $\boldsymbol{F}_{\text{pen}}$ is

$$\boldsymbol{F}_{\text{pen}}(\boldsymbol{\beta}) = -\frac{\partial^2}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}'} l_{\text{pen}}(\boldsymbol{\beta}) = \boldsymbol{F}(\boldsymbol{\beta}) + \mathcal{K}. \tag{3.16}$$

With these formulations at hand one can estimate the parameters using Fisher scoring or any other numerical optimization method. Some more details of estimations methods in the boosting context are given in Appendix B.1.

## 3.3. Variable Selection and Model Choice in Flexible Survival Models

As already discussed in Section 2.2.2, variable selection and model choice are important issues for deriving a model. First, we have to think about model prespecifications. This needs good collaboration and communication between the statistician and project partners. Here, considerations from background content count as well as technical aspects. Second, we need tools that help us assessing the functional forms and to check for deviations from the PH assumption. Here, some approaches have been developed so far. One of these is the multivariable fractional polynomial approach for time-varying effects (MFPT) of Sauerbrei et al. (2007) as already mentioned in Section 2.3.2. In the following we want to illustrate an approach with similarities to the well-known forward stepwise selection. Later, approaches based on boosting methods shall be discussed in Section 5, which built the main focus of this work.

### 3.3.1. Introduction and Definitions

Hofner et al. (2008) propose a two-stage stepwise selection strategy to choose both the relevant covariates and the corresponding modeling alternatives within the choice set of possible covariates simultaneously. For categorical covariates, competing modeling approaches are linear effects and time-varying effects, whereas nonparametric modeling provides a further alternative in the case of continuous covariates. Before continuing with the description of the procedure, we introduce in short mixed model based inference and define an appropriate information criterion for model comparison.

### Mixed Model Based Inference

Estimation of additive hazard regression models is based on the penalized log-likelihood criterion (3.14). The smoothing parameters $\lambda_l$ are crucial quantities in obtaining penalized likelihood estimates. A full inferential procedure that provides both estimates for the regression coefficients and the smoothing parameters can be derived based on a mixed model interpretation of additive hazard regression (Kneib & Fahrmeir 2007). We first have to note that the penalty term associated with a penalized spline can be equivalently interpreted as a special random effects distribution. Setting the smoothing parameter $\lambda = \frac{1}{\tau^2}$ in (3.6) leads to

$$\text{pen}(\boldsymbol{\beta}) = \frac{1}{2\tau^2}\boldsymbol{\beta}'\boldsymbol{K}\boldsymbol{\beta}. \tag{3.17}$$

Comparing this penalty to a Gaussian distribution with density

$$p(\boldsymbol{\beta}|\tau^2) \propto \exp\left(-\frac{1}{2\tau^2}\boldsymbol{\beta}'\boldsymbol{K}\boldsymbol{\beta}\right) \tag{3.18}$$

reveals that the penalty essentially equals the negative log-density of a Gaussian random effects distribution. Within that distribution, the smoothing parameter $\tau^2$ turns into a random effects variance.

A difficulty arising with the random effects distribution (3.18) is its partial impropriety arising from the rank-deficiency of the penalty matrix $\boldsymbol{K}$. This rank-deficiency reflects the fact that a $(d-1)$-th order polynomial remains unpenalized when applying a $d$-th order difference penalty as discussed in Section 3.1.1. Since standard mixed models require proper random effects distributions, a reparameterization into fixed effects representing the unpenalized part and random effects representing the penalized part has to be applied to all vectors of regression coefficients associated with nonparametric or time-varying effects (see Sec. 4.3.4 for more details). Afterwards, the model (2.12) can be written as

$$\eta_i(t_i) = \boldsymbol{w}(t_i)'\boldsymbol{\gamma} + \boldsymbol{z}(t_i)'\boldsymbol{\nu} \tag{3.19}$$

where $\boldsymbol{\gamma}$ comprises the collection of all fixed effects corresponding to covariates $\boldsymbol{w}(t_i)$, $\boldsymbol{\nu}$ contains random effects corresponding to covariates $\boldsymbol{z}(t_i)$, and all random effects variances are collected in the vector $\boldsymbol{\theta}$ of length $L + J$. In terms of the mixed model representation (3.19), the likelihood and log-likelihood will be denoted as $L(\boldsymbol{\gamma}, \boldsymbol{\nu})$ and $l(\boldsymbol{\gamma}, \boldsymbol{\nu})$ but actually coincide with the quantities derived in Section 3.2.

The advantage of the mixed model representation is the availability of algorithms for the joint determination of the random effects and of their variances. The approach is based on penalized likelihood estimation for the random effects and marginal likelihood estimation for the variances. The latter employs a Laplace approximation to the marginal likelihood, yielding a simple Newton-Raphson

algorithm (see Kneib & Fahrmeir (2007) for details). The mixed model based estimation procedure is implemented in the software package BayesX, freely available from `http://www.stat.uni-muenchen.de/~bayesx`.

### Information Criterion

Based on the mixed model estimation scheme from the previous section, it is possible to obtain estimates in a model with fixed model structure, i.e., given the modeling specification for the different covariates. However, in practice several competing modeling strategies exist for different types of covariates. The two-stage stepwise procedure, given in the following, provides a means of model choice. To determine which modeling alternative should be used in the model a suitable criterion for model comparison is needed.

Since estimation is based on mixed model methodology, it seems plausible to base model choice on Akaikes information criterion (AIC) for mixed models. However, two different versions of the AIC are available: The marginal AIC is based on the marginal likelihood of the mixed model representation (3.19) with the random effects integrated out, i.e.,

$$\text{AIC}_m = -2\log[\int L(\boldsymbol{\gamma}, \boldsymbol{\nu}) d\boldsymbol{\nu}] + 2[\dim(\boldsymbol{\gamma}) + \dim(\boldsymbol{\theta})]. \tag{3.20}$$

It consists of the log-marginal likelihood as model fit criterion and the number of fixed effects $\dim(\boldsymbol{\gamma})$ plus the number of smoothing variances $\dim(\boldsymbol{\theta})$ as a measure of model complexity. In contrast, the conditional AIC

$$\text{AIC}_c = -2l(\boldsymbol{\gamma}, \boldsymbol{\nu}) + 2\,\text{df} \tag{3.21}$$

is based on the conditional log-likelihood $l(\boldsymbol{\gamma}, \boldsymbol{\nu})$ in combination with the effective degrees of freedom df as a complexity measure. The degrees of freedom are defined as in (3.8) and serve as an effective number of parameters in the model. The degrees of freedom replace the usual parameter count since the effective dimensionality reduction induced by the random effects distribution has to be taken into account.

The random effects represent the penalized part of the nonparametric function and integrating them out corresponds to marginalizing over parts of the function. Since the nonparametric functions are of major interest in the analyses, a conditional model choice measure seems recommended. Moreover, the conditional AIC coincides with the classical AIC from the smoothing literature as outlined, for example, in Hastie & Tibshirani (1990b).

### 3.3.2. Two-Stage Stepwise Procedure

Based on the conditional AIC, the following two-stage stepwise selection strategy is proposed:

**Starting Model:** Define a starting model. Typically this will be the empty model containing only the baseline hazard rate. An alternative possibility to derive a non-empty starting model with preset variables is described in Section 6.2.1.
**Initial Choice Set:** Define an initial choice set of covariates not already included in the starting model.

[i] **Modeling Alternatives:** For each covariate in the choice set, define a set of modeling alternatives, for example, linear effect vs. time-varying effect in case of categorical variables, or linear effect vs. nonparametric effect vs. time-varying effect in case of continuous covariates.

[ii] **Estimation of Models:** For each of the covariates in the choice set and for each modeling possibility, estimate the hazard regression model obtained from the current model by adding the covariate in the respective modeling possibility and store the conditional AIC.

[iii] **Selection Step with Stopping Criterion:** If the minimal $\text{AIC}_c$ obtained in step [ii] is smaller than the $\text{AIC}_c$ of the current model, replace the current model with the best-fitting model from step [ii], delete the corresponding covariate from the choice set and go to step [iv]. Otherwise terminate the algorithm.

[iv] **Backward Deletion:** Perform a backward deletion step on the current model, i.e. estimate all hazard regression models obtained from the current model by dropping one covariate at a time. If an $\text{AIC}_c$ reduction can be achieved, make the reduced model with minimal $\text{AIC}_c$ the working model and add the deleted variable again to the choice set. Continue with step [i].

The selection procedure is called two-stage since it differentiates between inclusion of variables on the first stage and different modeling possibilities for the covariates on the second stage. It proceeds in a stepwise fashion, where each forward step for inclusion of additional terms is followed by a backward deletion step.

A "toy example" that illustrates the application of the two-stage stepwise procedure can be found in Table 3.1. We start with an empty model containing only the baseline hazard rate and three variables in the *initial choice set* with either two or three *modeling alternatives* (step [i]). For each variable and each modeling possibility, the model is fitted and the $\text{AIC}_c$ is calculated (step [ii]). "Apache II score" modeled as smooth term has the minimal $\text{AIC}_c$ and thus is

added to the *starting model*. In the next iteration, only two variables are left in the choice set. "Palliative operation for malignant disease" is added as linear term as it is the minimizer of $AIC_c$. In the last step of this example, age is chosen as linear term. The inclusion of the variable, in each step, improves the $AIC_c$ of the previous step (i.e., $AIC_c$ *decreases*) (step [iii]). Between step 1 and 2 no *backward deletion step* (step [iv]) is needed, as only one variable is included in the model so far. The backward step after step 2 was performed in the usual manner but did not lead to a better model.

Of course, the proposed selection scheme can be modified at some points if recommended by the application at hand. For example, one might think of starting with a full model instead of the empty model but this approach will suffer from two drawbacks: Firstly, it is not clear which model should be the full model. In particular, for continuous covariates it is unclear whether a full model should contain nonparametric or time-varying effects. Secondly, the full model would typically be overly complex. This would lead to a higher computational burden compared to the proposed strategy. Moreover, it may often be impossible to identify the full model from given data if, for example, the percentage of censoring is high or the number of possible covariates is large.

We have to mention that no formal test for time-varying coefficients is performed. Inclusion of time-varying terms (and smooth terms as well) is due to an $AIC_c$ based argumentation. We show an application of the two stage stepwise procedure in Section 6.2.1. For more details on the applications and methodology see Hofner et al. (2008) and Moubarak et al. (2008).

In the next chapters we want to discuss another approach with built in variable selection and model choice called boosting.

| Variable (**stage 1**) | Modeling Alternative (**stage 2**) | $AIC_c$ in step 1 | 2 | 3 |
|---|---|---|---|---|
| Apache II score (continuous) | linear | 3188.09 | – | – |
| | smooth | **3186.21** | – | – |
| | time-varying | 3188.37 | – | – |
| palliative operation for malignant disease (categorical) | linear | 3530.43 | **3176.31** | – |
| | time-varying | 3532.26 | 3177.98 | – |
| age (continuous) | linear | 3524.45 | 3178.18 | **3168.55** |
| | smooth | 3525.74 | 3178.37 | 3168.58 |
| | time-varying | 4073.94 | 3697.34 | 3685.98 |

backward deletion: not applicable

backward deletion: no improvement

Table 3.1.: **Toy example** for the two-stage stepwise procedure (simplified version of the first three selection steps of the starting model from our application, cf. Chapter 6).

# 4. Boosting Algorithms

Boosting was originally introduced in the field of machine learning for improved prediction of binary outcomes (Schapire 1990, Freund & Schapire 1996, 1997). Later, Breiman (1998, 1999) and Friedman et al. (2000) linked the original AdaBoost to statistical estimation schemes as they showed that AdaBoost and other boosting procedures can be seen as functional gradient descent algorithms in function space. Friedman et al. (2000) show a clear connection of boosting to forward stagewise additive modeling – which can be seen as the breakthrough of boosting from a statisticians point of view – and give some insight on how and why boosting works. A good overview of developments in boosting from a statistical perspective is given in Bühlmann & Hothorn (2007).

The two special cases AdaBoost and $L_2$Boosting (see e.g., Bühlmann & Yu 2003) are particularly descriptive and allow to get some ideas of what is going on in boosting. AdaBoost refits weighted data in such a way that misclassified observations get higher weights. The weights stem from the goodness of fit. Thus, the algorithm concentrates on observations that could not be properly fitted in previous iterations (Ridgeway 1999). $L_2$Boosting can be viewed as refitting of residuals. Again, the focus is on improving the "bad predictions" in later iterations, i.e., those observations with high residuals.

In this chapter, we give a short overview on the present state of boosting and highlight some special boosting procedures. We will present functional gradient descent boosting and the related likelihood-based boosting. Definitions of degrees of freedom and information criteria such as the Akaike information criterion (AIC) are addressed as well as the different possibilities of choosing and defining base-learners. Component-wise boosting as a method for variable selection is introduced. A model choice scheme that employs component-wise boosting in a special way will also be considered.

## 4.1. Forward Stagewise Additive Modeling

In many statistical settings one wants to get estimates of a function $f^*(\boldsymbol{x})$. Semi-parametric estimation leads to estimates of the form $f(\boldsymbol{x}) = \sum_{m=1}^{M} \alpha_m g(\boldsymbol{x}; \boldsymbol{\beta}_m)$. The function $g(\boldsymbol{x}; \boldsymbol{\beta}_m)$ is a simple parameterized function of input variables $\boldsymbol{x}$, determined by parameters $\boldsymbol{\beta}_m = (\beta_{m1}, \beta_{m2}, \ldots)$. The parameters $\alpha_m$ are just

multipliers. In the boosting context, we call $g(\boldsymbol{x}; \boldsymbol{\beta}_m)$ a weak learner or a base-learner. A classical choice for the base-learner is a small regression tree. In our case, $g(\boldsymbol{x}; \boldsymbol{\beta}_m)$ is chosen to be a B-spline (3.1) with pre-specified, equidistant knots and regression coefficients $\boldsymbol{\beta}_m$ combined with the difference penalty (3.6), i.e., a P-spline base-learner (see Sec. 4.3.2). The penalty needs to be added to the estimation algorithms in an appropriate form. Minimizing the negative log-likelihood, for example, is replaced by minimizing the negative, penalized log-likelihood.

Friedman et al. (2000) and Friedman (2001) introduce forward stagewise additive modeling. They state, that many learning techniques aim to minimize the expected value of a loss function $\rho$ (e.g., squared error loss or likelihood-based loss functions), i.e.,

$$f^*(\boldsymbol{x}) = \underset{f(\cdot)}{\operatorname{argmin}} \, \mathbb{E}_{Y,X}[\rho(y, f(x))]. \tag{4.1}$$

This can be achieved by minimizing the loss function averaged over the training data, i.e., by minimizing the empirical risk

$$(\hat{\alpha}_m, \hat{\boldsymbol{\beta}}_m)_{m=1}^M = \underset{(\alpha_m, \boldsymbol{\beta}_m)_{m=1}^M}{\operatorname{argmin}} \sum_{i=1}^n \rho \left( y_i, \sum_{m=1}^M \alpha_m g(\boldsymbol{x}; \boldsymbol{\beta}_m) \right). \tag{4.2}$$

This optimization problem can be very high-dimensional, as we have $M$ pairs of parameters where one parameter is again a vector of, classically many, coefficients. Thus, optimizing (4.2) is, in most cases, computationally demanding or even infeasible and hence, one tries to approximate (4.2) with a "greedy stagewise approximation". Stagewise additive modeling is an iterative approach, where in each iteration $m$, $m = 1, \ldots, M$, new functions $g(\boldsymbol{x}; \boldsymbol{\beta}_m)$ are added without changing the parameters $(\alpha_1, \ldots, \alpha_{m-1})$ and coefficients $(\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_{m-1})$ of the base-learners already added to the model. One estimates the coefficients $\boldsymbol{\beta}_m$ and parameters $\alpha_m$ by

$$(\hat{\alpha}_m, \hat{\boldsymbol{\beta}}_m) = \underset{(\alpha, \boldsymbol{\beta})}{\operatorname{argmin}} \sum_{i=1}^n \rho \left( y_i, \, \hat{f}^{[m-1]}(\boldsymbol{x}_i) + \alpha \, g(\boldsymbol{x}_i; \boldsymbol{\beta}) \right). \tag{4.3}$$

Afterwards, the function estimates are updated by

$$\hat{f}^{[m]}(\boldsymbol{x}) = \hat{f}^{[m-1]}(\boldsymbol{x}) + \hat{\alpha}_m g(\boldsymbol{x}; \hat{\boldsymbol{\beta}}_m). \tag{4.4}$$

The estimation of $\alpha_m$ and $\boldsymbol{\beta}_m$ is thus based on the loss function

$$\rho(y_i, \hat{f}^{[m-1]}(\boldsymbol{x}_i) + \alpha \, g(\boldsymbol{x}_i; \boldsymbol{\beta})) \tag{4.5}$$

with a fixed function $\hat{f}^{[m-1]}(\boldsymbol{x})$ from the previous iteration.

In the machine learning community this generic approach is called boosting (Friedman 2001). For a quadratic loss function $\rho$ (i.e., $L_2$-loss), boostingok reduces to refitting the residuals of the previous iteration. A detailed discussion of $L_2$Boosting can be found in Section 4.2.3.

# 4.2. FGD Boosting

## 4.2.1. Generic FGD

Starting from the forward stagewise additive model, Friedman et al. (2000) and Friedman (2001) derived the general framework for functional gradient descent (FGD) boosting. Given the estimate $\hat{f}^{[m-1]}$ of the previous iteration, the term $\hat{\alpha}_m g(\boldsymbol{x}; \hat{\boldsymbol{\beta}}_m)$ in equations (4.3) and (4.4) can be seen as the best greedy step towards the estimate of $f^*(x)$. This optimization is performed under the constraint that the step direction $g(\boldsymbol{x}; \boldsymbol{\beta})$ is a member of a pre-specified parameterized class of functions available for $g$. Thus, estimation can be seen as a steepest-descent step and consequently be rewritten using the data-based negative gradient

$$U_i^{[m]} = - \left. \frac{\partial \rho(y_i, f)}{\partial f} \right|_{f=\hat{f}^{[m-1]}(\boldsymbol{x}_i)} \tag{4.6}$$

with the negative gradient vector $\boldsymbol{U}^{[m]} = (U_1^{[m]}, \ldots, U_n^{[m]})'$ evaluated at the function of the previous iteration $\hat{f}^{[m-1]}(\boldsymbol{x}_i)$. This gives the steepest-descent direction in the $n$ dimensional data space at the point $\hat{f}^{[m-1]}(x)$, which is *only* defined at the observed data points. Generalization to other points in space can, for example, be achieved by choosing the function $g(\boldsymbol{x}; \hat{\boldsymbol{\beta}}_m)$ most parallel to $\boldsymbol{U}^{[m]}$. This function can be estimated by minimizing the squared error loss:

$$\hat{\boldsymbol{\beta}}_m = \underset{\alpha, \boldsymbol{\beta}}{\operatorname{argmin}} \sum_{i=1}^{n} \Big( U_i^{[m]} - \alpha g(\boldsymbol{x}_i; \boldsymbol{\beta}) \Big)^2 . \tag{4.7}$$

Thus, the unconstrained negative gradient $\boldsymbol{U}^{[m]}$ can be replaced by the constrained estimate $g(\boldsymbol{x}; \hat{\boldsymbol{\beta}}_m)$. This leads to the least-squares problem (4.7) instead of the stagewise estimation scheme (4.3), which results in a much more effective estimation algorithm, especially if the latter estimate is difficult to obtain.

The first connection of boosting and steepest-descent algorithms was shown by Breiman (1998, 1999) who derived that the AdaBoost algorithm can be seen as steepest-descent algorithm in function space and thus is a functional gradient descent algorithm.

When we use a likelihood-based loss function, for example the negative log-likelihood, and directly minimize (4.5) this leads to likelihood-based boosting as introduced by Tutz & Binder (2006) (see Sec. 4.4). Applying (4.7) instead can be seen as generic functional gradient descent (FGD) boosting as described in Friedman (2001). The algorithm for the latter is given in the following paragraph.

### Generic FGD Algorithm

(i) **Initialization:** Set the iteration index $m := 0$. Initialize the function estimate $\hat{f}^{[0]}(\cdot)$ with an offset value, typically

$$\hat{f}^{[0]}(\cdot) \equiv \underset{c}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \rho(y_i, c)$$

or

$$\hat{f}^{[0]}(\cdot) \equiv 0.$$

(ii) **Negative gradient:** Increase $m$ by 1. Compute the negative gradient of the loss function $\rho(y_i, f)$ evaluated at the function values of the previous iteration $\hat{f}^{[m-1]}(\boldsymbol{x}_i)$:

$$U_i^{[m]} = - \left. \frac{\partial \rho(y_i, f)}{\partial f} \right|_{f=\hat{f}^{[m-1]}(\boldsymbol{x}_i)}, \; i = 1, \ldots, n$$

(iii) **Estimation:** Fit the negative gradient vector $\boldsymbol{U}^{[m]} = (U_1^{[m]}, \ldots, U_n^{[m]})$ to $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ by a real-valued base procedure (classically by least squares or penalized least squares estimation):

$$(\boldsymbol{x}_i, U_i)_{i=1}^n \overset{\text{base procedure}}{\longrightarrow} \hat{g}^{[m]}(\cdot)$$

The base-learner $\hat{g}^{[m]}(\cdot) = g(\cdot; \hat{\boldsymbol{\beta}}_m)$ is a member of a parameterized class and can be regarded as an approximation of the negative gradient vector.

(iv) **Update:** Compute the update

$$\hat{f}^{[m]}(\cdot) = \hat{f}^{[m-1]}(\cdot) + \nu \cdot \hat{g}^{[m]}(\cdot)$$

with step-length factor $0 < \nu \leq 1$.

(v) **Stopping rule:** Continue iterating steps (ii) to (iv) until $m = m_{\text{stop}}$ for a given stopping iteration $m_{\text{stop}}$.

The stopping iteration $m_{\text{stop}}$ is usually determined based on an information criterion (e.g., AIC) or via cross-validation. It can be seen as the major tuning parameter in the generic FGD algorithm (see Sec. 4.5).

The add-on package `mboost` (Hothorn et al. 2007) for the statistical software package `R` (R Development Core Team 2007) implements boosting as a generic functional gradient descent (FGD) algorithm for arbitrary loss functions as described above.

### 4.2.2. Step-Length

The step-length factor $\nu$ is of minor importance in the above algorithm, as long as it is small enough, e.g., $\nu = 0.1$. A smaller value of $\nu$ typically requires a larger number of boosting iterations, i.e., more computing time. Sufficiently small values of $\nu$ (such as 0.1) have empirically found to be *potentially* better and almost never worse, regarding the predictive accuracy (Friedman 2001). In the generic FGD boosting approach introduced by Friedman (2001) an additional line search between steps (iii) and (iv) for the optimal step-length is performed. Bühlmann & Hothorn (2007) argument that this seems unnecessary for achieving a good estimator $\hat{f}^{[m_{\text{stop}}]}$, but the computational time is potentially increased.

Friedman (2001) states that choosing a small step-length factor can be seen as incremental shrinkage, as each update in step (iv) is scaled by the learning rate $\nu$. Decreasing the learning rate improves the prediction performance strongly. The reason for this is less clear, as shrinkage in each boosting iteration produces a very complex result. In the best case, global shrinkage of the entire model provides only minor improvements, far away from the dramatic effect of incremental shrinkage. The good performance of incremental shrinkage perhaps becomes a bit clearer, if we think, for example, of a group of highly correlated predictors. If we choose just a fraction of the fit of the best-fitting of these predictors in each iteration, the other predictors have the chance to enter the model in subsequent steps. Thus, the fit of the model may be improved. If we would include the best-fitting predictor completely in the first iteration, none of the other predictors of this group might be able to improve this fit substantially and hence, no more variables (of this group) would enter the model. Therefore, in this case global shrinkage of a model where only one variable of the highly correlated group is selected cannot improve the model in the same way as incremental shrinkage could do.

### 4.2.3. $L_2$Boosting

$L_2$Boosting is the classical candidate for regression problems with many possible predictors. The generic loss function is specified as squared error loss $\rho(y, f) = \frac{|y-f|^2}{2}$. Thus, the following algorithm can be derived from the generic FGD algorithm (Sec. 4.2.1):

#### $L_2$Boosting Algorithm

(i) **Initialization:** Set the iteration index $m := 0$. Initialize the function estimate $\hat{f}^{[0]}(\cdot)$ with an offset value. The default is $\hat{f}^{[0]}(\cdot) \equiv \overline{y}$.

(ii) **Negative gradient:** Increase $m$ by 1. Compute the residuals $U_i^{[m]} = y_i - \hat{f}^{[m-1]}(\boldsymbol{x}_i)$ for $i = 1, \ldots, n$.

(iii) **Estimation:** Fit the negative gradient vector $\boldsymbol{U}^{[m]} = (U_1^{[m]}, \ldots, U_n^{[m]})$ to $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ by a real-valued base procedure (classically by (penalized) least squares estimation):

$$(\boldsymbol{x}_i, U_i)_{i=1}^n \overset{\text{base procedure}}{\longrightarrow} \hat{g}^{[m]}(\cdot)$$

(iv) **Update:** Compute the update

$$\hat{f}^{[m]}(\cdot) = \hat{f}^{[m-1]}(\cdot) + \nu \cdot \hat{g}^{[m]}(\cdot)$$

with a step-length factor $0 < \nu \leq 1$ (cf. Sec. 4.2.2) that means, proceed along the negative gradient vector.

(v) **Stopping rule:** Continue iterating steps (ii) to (iv) until $m = m_{\text{stop}}$ for a given stopping iteration $m_{\text{stop}}$.

Note that the negative gradient vector becomes the residual vector. Thus, $L_2$Boosting can be regarded as the refitting of residuals. We can gain several insights from this simple case. Bühlmann & Hothorn (2007) show that overfitting occurs if one does not stop the boosting algorithm at an early stage. Smoothing splines with low-order degrees of freedom adapt higher order smoothness with continuing boosting iterations. But at the same time, overfitting is much slower than with one single smoothing spline with increasing degrees of freedom (Bühlmann & Yu 2003).

### 4.2.4. FGD Boosting for Survival Models

For survival models with censored outcome, the negative gradient of the Cox partial likelihood can be used to fit proportional hazard models with boosting algorithms (Ridgeway 1999). Another approach is given in Hothorn et al. (2006). Here a weighted least squares framework is applied with weights from the inverse probability of censoring. Therefore, we recall the definitions and notations from Section 2.1. The observed survival times are denoted as $t_i$, while $t_i^*$ are the realizations of the true survival times. The corresponding indicator for non-censoring is $\delta_i$ and $\boldsymbol{x}_i$ is the observed data. The restrictive assumption that $c_i$ is conditionally independent of $t_i$ given $\boldsymbol{x}_i$ must hold. For the *true* survival times a squared error loss

$$\rho(y^*, f) = |y^* - f|^2 \tag{4.8}$$

is considered, where $y^* = \log(t_i^*)$. For the *observed* data $y = \log(t)$ Hothorn et al. (2006) propose the weighted version

$$\rho_{obs}(y, f; \delta, \boldsymbol{x}) = (y - f)^2 \frac{\delta}{G(t|\boldsymbol{x})}, \tag{4.9}$$

where

$$G(c|\boldsymbol{x}) = P(C > c|\boldsymbol{X} = \boldsymbol{x}). \tag{4.10}$$

So the observed data loss is weighted with the inverse probability for censoring $\frac{\delta}{G(t|\boldsymbol{x})}$. For model fitting, $G(\cdot|\boldsymbol{x})$ is estimated, for example, with the Kaplan-Meier estimator and $L_2$Boosting with the weighted squared error loss is applied. Thus, one can use various base procedures as long as they allow for weighted least squares fitting. Degrees of freedom for the boosted model and the AIC can be derived analogously to Section 4.5.

Estimation in the accelerated failure time framework, i.e., for parametric distributions of the survival time, is considered by Schmid & Hothorn (2008). The negative log-likelihood of the survival distribution is used as loss function in the FGD framework. A simultaneous estimation step for the scale parameter of the model is introduced. They show that the additional estimation of the scale parameter does not effect the variable selection properties of boosting.

## 4.3. Choosing Base-Learners

### 4.3.1. Weak Learner – The Low-Variance Principle

As already briefly stated, different types of base-learners exist. A base-learner can, for example, be a tree, a linear, or even a smooth function such as a smoothing spline or P-spline. The latter will be discussed in more detail in the next section. After choosing the appropriate functional form for a base-learner, the question arises, how complex a non-linear base-learner should be. Bühlmann & Hothorn (2007) give a general answer to this question: They recommend to choose a base-learner with the desired structure and low variance. The possibly enlarged estimation bias that is due to the low variance is accepted, as it vanishes with increasing boosting iterations. For smoothing spline or P-spline base-learners this would imply to choose the degrees of freedom small, for example df = 4. The step-size factor $\nu$ can be seen as a shrinkage factor (see Sec. 4.2.2). Hence, it also implies a reduced variance and a potentially larger estimation bias.

Even if the bias of a single base-learner is large, due to the iterative nature of boosting, the bias reduces over the iterations. Bühlmann & Yu (2003) showed

that smoothing splines with low degrees of freedom are even capable of adapting higher degrees of freedom of the true, underlying model.

Some remarks on the learning capacity of a base-learner and connections to its weakness are given in Section 4.5.1.

### 4.3.2. P-Spline Base-Learners

P-splines as base-learners were first considered in the boosting context by Tutz & Binder (2006) (see Sec. 4.4.1). A thorough investigation of replacing smoothing splines, as proposed in the boosting context by Bühlmann & Yu (2003), with the computationally much more efficient P-splines was done by Schmid & Hothorn (2007). They showed, in the framework of $L_2$Boosting that the number of knots in boosting, as in other modeling approaches, has very little influence on the prediction performance, as long as it is sufficiently large ($20 - 50$ knots). Moreover they showed that the degrees of freedom for P-splines should be small (e.g., df $= 4$) to obtain a weak learner (cf. Sec. 4.5.1). The choice of the step length factor $\nu$, for small values of $\nu$, is of minor importance, as a smaller value leads to almost the same predictive power but it requires an increased number of boosting iterations.

Note that for P-spline base-learners, as well as for smoothing splines or other base-learners with penalty, the fitting criteria are transformed to penalized ones. For least squares estimation thus, we get penalized least squares and in the context of likelihood-based boosting, maximum likelihood estimation becomes penalized maximum likelihood estimation. For P-splines the penalty matrix (3.6) is applied in both cases.

### 4.3.3. Component-Wise Boosting

Component-wise boosting as presented, for example, in Bühlmann & Yu (2003), is a useful extension to classical boosting procedures that incorporates variable selection into boosting. Especially for high-dimensional data sets this is a big advantage. Component-wise boosting uses different base-learners, (at least) one for every potential predictor variable $\boldsymbol{x}_j \in \mathbb{R}^n, j = 1, ..., p$ . In each boosting iteration not only one base-learner is fitted to the negative gradient vector $\boldsymbol{U}^{[m]}$ but all base-learners $g_j(\boldsymbol{x}_j, \boldsymbol{\beta}_j)$ are estimated *separately*. Only the best fitting base-learner, with respect to some criterion, is then selected. For FGD boosting, classically the base-learner $j^*$ that minimizes the residual sum of squares is selected:

$$j^* = \underset{1 \leq j \leq p}{\operatorname{argmin}} \sum_{i=1}^{n} (U_i - g_j(x_{ij}, \hat{\boldsymbol{\beta}}_j))^2. \tag{4.11}$$

Thus, we get either a linear or an additive model including variable selection if the base-learners are linear or smooth functions (such as P-splines), respectively. As discussed later in Section 4.5, it is important to stop the boosting algorithm before overfitting occurs. Here the importance of choosing an appropriate stopping iteration is backed by another argument: Variable selection will, in general, only be carried out if the number of iterations and thus the number of potentially selected variables is small enough.

Later, in Section 4.4.1, we also introduce component-wise likelihood-based boosting. Note that the selection criterion then is a likelihood based criterion as, e.g., the log-likelihood or the deviance.

As Schmid & Hothorn (2007) state, component-wise boosting can even be computed if the number of observations $n$ is smaller than the number of possible predictors $p$. More than $n$ variables may enter the model as variable selection is built into the modeling process and only one base-learner is fitted at a time. So component-wise boosting is even capable to choose and fit more than $n$ different base-learners and thus is superior to procedures as stepwise linear regression wich can maximally include $n$ variables.

### 4.3.4. Model Choice

In the following, we do not restrict to one single type of base-learners in component-wise boosting. We allow different, competing base-learners for one covariate. Thus, with the selection of a base-learner we select a variable in a special functional form. So, we get variable selection as in the classical component-wise boosting approach *and* in addition model choice is performed (Kneib et al. 2007).

We start with a generic model

$$\mathbb{E}(y_i|\boldsymbol{x}_i) = h(\eta_i(\boldsymbol{x}_i)),\ i = 1, \ldots, n, \tag{4.12}$$

with response function $h$ and an additive predictor of the form

$$\eta_i(\boldsymbol{x}_i) = \beta_0 + \sum_{j=1}^{J} f_j(\boldsymbol{x_i}), \tag{4.13}$$

where the functions $f_j(\boldsymbol{x}_i)$ are a generic representation of different types of covariate effects. To come back to the Cox-type additive model as introduced in Sections 2.3 and 3.1.3 and to make the model formulation more concrete, consider the following examples of functions $f_j(\boldsymbol{x}_i)$: The functions can represent

- linear effects: $f_j(\boldsymbol{x}_i) = f_{\text{linear}}(\tilde{x}_i) = \tilde{x}_i\beta$, where $\tilde{x}_i \in \boldsymbol{x}_i$.

- smooth effects: $f_j(\boldsymbol{x}_i) = f_{\text{smooth}}(\tilde{x}_i)$, where $\tilde{x}_i \in \boldsymbol{x}_i$ is a continuous covariate and $f_{\text{smooth}}$ is a smooth function.
- time-varying effects: $f_j(\boldsymbol{x}_i) = f_{\text{smooth}}(t) \cdot \tilde{x}_i$, where $(\tilde{x}_i, t) \in \boldsymbol{x}_i$. The covariate $\tilde{x}_i$ can be either continuous or categorical, $t$ represents the observed survival time. $f_{\text{smooth}}$ is again a smooth function.

We see that a covariate $\tilde{x}_i$ can enter the model in up to three different ways. The effect can be either linear, smooth (in the case of a continuous covariate $\tilde{x}_i$) or time-varying. Hence, the question arises, how each variable should enter the model. One solution is, to specify a separate base-learner for each suitable modeling possibility. Component-wise boosting then chooses between covariates and modeling possibilities at the same time, if the boosting algorithm is stopped after an appropriate number of iterations.

The effects can be expressed in the same way as in the Cox-type additive model. Thus, linear effects just enter the model as linear-base-learners, smooth effects can be added using P-spline base-learners and time-varying effects are derived as an interaction between a P-spline of time and the covariate $\tilde{x}_i$.

To make the different base-learners comparable in terms of complexity, one could try to define equal degrees of freedom for each term. Increasing the smoothing parameter $\lambda$ leads to decreasing degrees of freedom. But as stated in Section 3.1.1, Eilers & Marx (1996) showed that a polynomial of order $d-1$ remains unpenalized by a $d$-th order difference penalty if the degree of the B-spline basis is large enough. Thus, we cannot make the degrees of freedom arbitrary small. As normally we are using B-splines of degree 3 or higher, the degrees of freedom for difference penalties of order 2 or higher remain greater than one. Hence, making such smooth effects comparable with single linear effects seems impossible.

Kneib et al. (2007) propose a modified parameterization of the P-splines, which we already briefly sketched in Section 3.3. Therefore, with a continuous covariate $x$, the smooth function $f_{\text{smooth}}(x)$ is split into a parametric part consisting of the unpenalized polynomial of order $d-1$ and the deviation from this polynomial $f_{\text{centered}}(x)$:

$$f_{\text{smooth}}(x) = \underbrace{\beta_0 + \beta_1 x + \ldots + \beta_{d-1} x^{d-1}}_{\text{unpenalized, parametric part}} + \underbrace{f_{\text{centered}}(x)}_{\text{deviation from polynomial}} \tag{4.14}$$

For the parametric part, separate linear base-learners are added for each term. The deviation from the polynomial $f_{\text{centered}}$ can be included as smooth effect with exactly one degree of freedom. Thus, we have the possibility to check, if $x$ has any influence at all (i.e., none of the base-learners depending on $x$ is selected). If $x$ is influential, we have the additional possibility to check whether we need a nonparametric part to describe the influence.

Varying coefficient terms (Hastie & Tibshirani 1993), as time-varying effects can be reparameterized in the same manner, i.e.,

$$f_{\text{smooth}}(t) \cdot x = \underbrace{\beta_0 \cdot x + \beta_1 t \cdot x + \ldots + \beta_{d-1} t^{d-1} \cdot x}_{\text{unpenalized, parametric part}} + \underbrace{f_{\text{centered}}(t) \cdot x}_{\text{deviation from polynomial}}, \tag{4.15}$$

where $t$ is the time and $x$ is an arbitrary covariate.

Technically, this model decomposition is achieved by decomposing the vector of regression coefficients $\boldsymbol{\beta}$ into $(\widetilde{\boldsymbol{\beta}}_{\text{unpen}}, \widetilde{\boldsymbol{\beta}}_{\text{pen}})'$, i.e., into an unpenalized and a penalized part. This can be achieved based on a spectral decomposition of the penalty matrix. Details in the context of geoadditive regression models can be found in Fahrmeir et al. (2004).

One should add that the clear separation and straight forward interpretation of the resulting selections and effects get lost if one adds the decomposition of $f_{\text{smooth}}(x)$ and at the same time the decomposition of $f_{\text{smooth}}(t) \cdot x$ to the model. Thus, we could get linear terms, polynomial terms, and smooth terms for $x$ as well as interactions of $x$ with a linearly, polynomially, and smoothly added $t$. With this many possible base-learners, interpretation is at least tricky.

## 4.4. Likelihood-Based Boosting

With stagewise additive modeling as a "starting point" for boosting, leading to generic functional gradient descent boosting on the one hand and likelihood-based boosting on the other hand, one has a clear linkage between those two concepts. Still, the two concepts are not the same. Likelihood-based boosting for all simple exponential families was introduced by Tutz & Binder (2006). They derive a single framework for generalized additive models with all kinds of link functions and distributions.

Let $\boldsymbol{x}_i, i = 1, \ldots, n$ be again the input variables and $y_i$ the corresponding dependent variable. The conditional distribution of $y_i|\boldsymbol{x}_i$ is assumed to follow a simple exponential family

$$f(y_i|\boldsymbol{x}_i) = \exp\left\{\frac{y_i\theta_i - b(\theta_i)}{\phi} + c(y_i, \phi)\right\}, \tag{4.16}$$

where $\theta_i$ is the canonical parameter and $\phi$ is the dispersion parameter. The classical linear predictor $\eta_i = \boldsymbol{x}_i'\boldsymbol{\beta}$ is replaced by a more general predictor. In each boosting step $m$, a base-learner $g(\boldsymbol{x}_i; \boldsymbol{\beta}_m)$ is fitted. This could be again a P-spline base-learner where $\boldsymbol{\beta}_m$ determines the weights of the spline functions and the location and number of knots.

The log-likelihood to be maximized is then given by

$$l(\boldsymbol{\beta}) = \sum_{i=1}^{n} \frac{y_i \theta_i - b(\theta_i)}{\phi} + c(y_i, \phi), \tag{4.17}$$

where the canonical parameter $\theta_i$ is a function of $\eta_i$. The response function $h$ connects the conditional expectation $\mathbb{E}(y_i|\boldsymbol{x}_i)$ with the predictor such that

$$\mathbb{E}(y_i|\boldsymbol{x}_i) = \mu_i = h(\eta_i). \tag{4.18}$$

#### Likelihood-Based Boosting Algorithm

(i) **Initialization:** Set the iteration index $m := 0$. Initialize the additive predictor $\hat{\eta}^{[0]}$ with the maximizer of the log-likelihood of the intercept model

$$\mu^{[0]} = h(\eta^{[0]}(\cdot))$$

as offset value, where $\eta^{[0]}(\cdot) \equiv c$, i.e., a model with only a constant:

$$\hat{\eta}^{[0]}(\cdot) \equiv \underset{c}{\operatorname{argmax}}\ l(c)$$

(ii) **Estimation:** Increase $m$ by 1. Fit the model

$$\mu^{[m]} = h\left(\hat{\eta}^{[m-1]}(\cdot) + g(\cdot)\right)$$

with base-learner $g(\cdot) = g(\cdot; \boldsymbol{\beta})$. The estimate $\hat{\boldsymbol{\beta}}$ is determined by one-step Fisher scoring (see below) where $\hat{\eta}^{[m-1]}(\cdot)$ is treated as an offset.

(iii) **Update:** Compute the update

$$\hat{\eta}^{[m]}(\cdot) = \hat{\eta}^{[m-1]}(\cdot) + \hat{g}(\cdot),$$

where $\hat{g}(\cdot) = g(\cdot; \hat{\boldsymbol{\beta}})$.

(iv) **Stopping rule:** Continue iterating steps (ii) to (iii) until $m = m_{\text{stop}}$.

The estimation of coefficients in each boosting iteration is performed by numerical optimization. A classical approach for likelihood optimization is the Fisher scoring algorithm. Coefficients $\boldsymbol{\beta}$ are estimated iteratively starting with an initial value $\boldsymbol{\beta}^{(0)}$ by

$$\hat{\boldsymbol{\beta}}^{(n+1)} = \hat{\boldsymbol{\beta}}^{(n)} + \boldsymbol{F}^{-1}(\hat{\boldsymbol{\beta}}^{(n)})\boldsymbol{s}(\hat{\boldsymbol{\beta}}^{(n)}), \tag{4.19}$$

with Fisher matrix $\boldsymbol{F}(\cdot)$ and score vector $\boldsymbol{s}(\cdot)$. Convergence of the algorithm is reached when $l(\hat{\boldsymbol{\beta}}^{(n+1)}) - l(\hat{\boldsymbol{\beta}}^{(n)})$ is sufficiently small. As boosting is based on weak learners (cf. Sec. 4.3), total convergence of the Fisher scoring algorithm

is not needed. Hence, Tutz & Binder (2006) propose to use a one-step Fisher scoring estimate, i.e., just update (4.19) once. Using a one-step Fisher scoring estimate can be seen as analogon to the multiplication of the step-length factor $\nu$ in generic FGD boosting. Thus, a step-length factor is not needed here. To simplify the estimation even further, Tutz & Binder choose the initial value $\boldsymbol{\beta}^{(0)} = 0$ in (4.19). Hence, the estimation problem in boosting iteration $m$ becomes

$$\hat{\boldsymbol{\beta}}_{\text{new}} = \left(\boldsymbol{F}^{[m-1]}\right)^{-1} \boldsymbol{s}^{[m-1]}, \tag{4.20}$$

where $\boldsymbol{F}^{[m]}$ is the Fisher matrix evaluated at the estimation after the previous iteration $\hat{\eta}^{[m-1]}(\cdot)$ and analogously $\boldsymbol{s}^{[m-1]}$ is the score function evaluated at the estimation after the previous iteration. For penalized base-learners such as P-spline base-learners (cf. Sec. 4.3.2) we get a penalized estimation scheme, where the Fisher matrix and the score vector are replaced by their penalized counterparts.

## 4.4.1. Component-Wise GAMBoost

Two special cases of likelihood-based boosting are given by Tutz & Binder (2006). Firstly, they consider a component-wise boosting algorithm for generalized additive models which they call GAMBoost with P-splines. Secondly, GAMBoost with penalized stumps is taken into account. In the following we will restrict to the former case of GAMBoost. The amount of smoothness that a (P-spline) base-learner implies is chosen by specifying a single smoothing parameter $\lambda$ for all base-learners. To obtain a weak learner, $\lambda$ should be chosen large. Tutz & Binder (2006) recommend to select $\lambda$ from a coarse grid such that the number of boosting iterations $m_{\text{stop}}$ is greater or equal to 50. They conclude that the different amount of smoothing needed for each variable is automatically adapted, as we have the variable selection step (see below). Variables that are selected more often can potentially adapt higher order smoothness than those that are selected only very seldom.

### Component-Wise GAMBoost with P-Splines (Algorithm)

(i) **Initialization:** Set the iteration index $m := 0$.

a) Initialize the function estimates

$$\hat{f}_j^{[0]}(\cdot) \equiv 0, j = 1, \ldots, J.$$

b) Initialize the additive predictor $\hat{\eta}^{[0]}$ with the maximizer of the log-likelihood of the intercept model

$$\mu^{[0]} = h(\eta^{[0]}(\cdot))$$

as offset value, where $\eta^{[0]}(\cdot) \equiv c$, i.e., a model with only a constant:

$$\hat{\eta}^{[0]}(\cdot) \equiv \underset{c}{\operatorname{argmax}}\, l(c).$$

(ii) **Estimation:** Increase $m$ by 1. For all P-spline base-learners

$$g_j(\cdot) = g(\cdot; \boldsymbol{\beta}_j),\ j = 1, \ldots, J$$

fit the model

$$\mu_j^{[m]} = h\left(\hat{\eta}^{[m-1]}(\cdot) + g_j(\cdot)\right).$$

The estimate $\hat{\boldsymbol{\beta}}_j$ is determined by one-step Fisher scoring with penalized Fisher matrix and penalized score function, where $\hat{\eta}^{[m-1]}(\cdot)$ is treated as an offset.

(iii) **Selection:** Choose the base-learner $\hat{g}_{j^*}$ with

$$j^* = \underset{j}{\operatorname{argmax}} \left[\operatorname{Dev}\left(\hat{\eta}^{[m-1]}(\cdot)\right) - \operatorname{Dev}\left(\hat{\eta}^{[m-1]}(\cdot) + \hat{g}_j(\cdot)\right)\right],$$

where $\hat{g}_j(\cdot) = g(\cdot; \hat{\boldsymbol{\beta}}_j)$.

(iv) **Update:**

a) Compute the update for the function estimate of the selected base-learner

$$\hat{f}_{j^*}^{[m]}(\cdot) = \hat{f}_{j^*}^{[m-1]}(\cdot) + \hat{g}_{j^*}(\cdot)$$

and set $\hat{f}_j^{[m]}(\cdot) = \hat{f}_j^{[m-1]}(\cdot)$ otherwise (i.e., for $j \neq j^*$).

b) Compute the update for the additive predictor

$$\hat{\eta}^{[m]}(\cdot) = \hat{\eta}^{[m-1]}(\cdot) + \hat{g}_{j^*}(\cdot).$$

(v) **Stopping rule:** Continue iterating steps (ii) to (iv) until $m = m_{\text{stop}}$.

The deviance in step iii is defined as

$$\operatorname{Dev}(\hat{\eta}(\boldsymbol{x}_i)) = -2\phi \sum_{i=1}^{n} \Big[ l_i(\underbrace{h(\hat{\eta}(\boldsymbol{x}_i))}_{\hat{\mu}_i}) - l_i(y_i) \Big], \tag{4.21}$$

where $\phi$ is the dispersion parameter from (4.16), $h$ is the response function, $l_i(\hat{\mu}_i)$ is the log-likelihood of observation $i$ and $l_i(y_i)$ is the log-likelihood, where $\hat{\mu}_i$ is replaced by $y_i$, i.e., the maximized log-likelihood (Fahrmeir & Tutz 2001).

The maximal difference in the deviances, or analogously, as $\operatorname{Dev}\left(\hat{\eta}^{[m-1]}(\cdot)\right)$ is just a constant for all $j$, the minimal deviance $\operatorname{Dev}\left(\hat{\eta}^{[m-1]}(\cdot) + \hat{g}_j(\cdot)\right)$, is chosen as criterion for the selection of the appropriate base-learner in each boosting step (Tutz & Binder 2006). Note that the deviance depends on the form of the considered distribution.

### 4.4.2. Likelihood-Based Boosting for Survival Models

Binder & Schumacher (2008) introduce a likelihood-based boosting algorithm for censored time-to-event data, where they explicitly consider mandatory covariates. These are covariates of clinically high importance that should be included in the model by all means. They therefore use an offset-based update mechanism where mandatory covariates can be included unpenalized whereas other covariates enter the model penalized if they are chosen. This approach proves to be useful, in particular in high-dimensional settings.

## 4.5. Stopping Criteria

One way of regularization for boosting is given by using weak learners as basis functions. Weak learners can be achieved by choosing a small step-length $\nu$ to make the base-learner $g(\cdot, \boldsymbol{\beta}_j)$ a "weaker" learner or by directly specifying the base-learner such that it is "weak enough". The latter approach is, for example, chosen in likelihood-based boosting as described in Tutz & Binder (2006), where the base-learner is obtained by one-step Fisher scoring. Another way of regularization is given by restricting the number of components in the model by choosing a relatively small stopping iteration $m_{\text{stop}}$ (compare 4.2.2). There are two ways to determine $m_{\text{stop}}$. Firstly, we can use information criteria as the AIC, the corrected AIC (Hurvich et al. 1998) or the gMDL criterion (Hansen & Yu 2001). Secondly, cross-validation (CV) can be applied. The latter is often very time consuming, especially for high-dimensional data sets. Both approaches, the AIC and the CV approach, try to mimic the performance of the model on new data taking overfitting due to over-parameterization into account. Another possibility, especially useful for simulation studies, is to draw an independent validation data set from the distribution that was used to sample the learning data and optimize a stopping criterion on the validation data set. Thus, we evaluate the performance on new data which is exactly the case we try to emulate (when we use AIC, CV, etc.). An example would be, to fit the model on the learning sample and to define the stopping iteration as the one that maximizes the unpenalized likelihood (i.e., the empirical risk for likelihood-based boosting) on the validation sample. This can be a good choice, as we do not need to split the original data set, i.e., we do not reduce the learning sample. Moreover, evaluating the criterion (e.g., the likelihood) on the validation sample is typically very fast and easy and reflects exactly the quantity we are interested in.

In all cases it is favorable to fit a larger number of iterations and determine the appropriate stopping iteration $m_{\text{stop}}$ afterward, to avoid local minima (see e.g., Kneib et al. 2007).

### 4.5.1. Definition of Hat Matrices

For $L_2$Boost, Bühlmann & Yu (2003) give a definition of the hat matrix of the boosting procedure as follows: Let $x_i, i = 1, \ldots, n$ be a one dimensional predictor and $y_i$ be the corresponding continuous response. Thus, a base-learner can be represented by a hat matrix (or a smoother matrix) $\boldsymbol{\mathcal{S}} : \mathbb{R}^n \to \mathbb{R}^n$. The hat matrix $\boldsymbol{\mathcal{S}}$ maps the response variable $\boldsymbol{y}$ to the fitted values $\hat{f}(\boldsymbol{x}) = (\hat{f}(x_1), \ldots, \hat{f}(x_n))'$. Using for simplicity $\hat{f}^{[0]} \equiv 0$, they show that the hat matrix in boosting iteration $m$ equals

$$\boldsymbol{\mathcal{B}}_m = \boldsymbol{\mathcal{B}}_{m-1} + \nu\boldsymbol{\mathcal{S}}(\boldsymbol{I} - \boldsymbol{\mathcal{B}}_{m-1}) = \boldsymbol{I} - (\boldsymbol{I} - \nu\boldsymbol{\mathcal{S}})^m. \tag{4.22}$$

From (4.22) one can conclude that if $0 < \|\boldsymbol{I} - \nu\boldsymbol{\mathcal{S}}\| < 1$ for a suitable norm, the base-learner has a "learning capacity" in the sense that the residual vector is "smaller" than the response vector $\boldsymbol{y}$. If the base-learner has a learning capacity, $\boldsymbol{\mathcal{B}}_m$ converges to the identity $\boldsymbol{I}$ as $m \to \infty$. Hence, $\hat{\boldsymbol{y}} = \boldsymbol{\mathcal{B}}_m\boldsymbol{y}$ converges to the fully saturated model $\boldsymbol{y}$, where all values are interpolated exactly (Bühlmann & Hothorn 2007). This shows explicitly that one has to stop early in order to prevent overfitting.

Schmid & Hothorn (2007) state that a base-learner is weak, if $\|\boldsymbol{I} - \nu\boldsymbol{\mathcal{S}}\|$ is close to one. Thus, a larger number of boosting iterations $m_{\text{stop}}$ is required to obtain a fit that is optimal with respect to some criterion (e.g., the AIC). They conclude that $\|\boldsymbol{I} - \nu\boldsymbol{\mathcal{S}}\|$ is a measure for the weakness of a base-learner. Furthermore, they show that when one uses the scaled Frobenius norm $\|\boldsymbol{I} - \nu\boldsymbol{\mathcal{S}}\|_F := n^{-1}\,\text{trace}\,[(\boldsymbol{I} - \nu\boldsymbol{\mathcal{S}})'(\boldsymbol{I} - \nu\boldsymbol{\mathcal{S}})]$ to measure the weakness of the corresponding smoothing spline or P-spline base-learner, for $0 < \nu < 1$, $\|\boldsymbol{I} - \nu\boldsymbol{\mathcal{S}}\|_F$ is strictly increasing in the smoothing parameter $\lambda$ and converges to 1 as $\lambda \to \infty$. An increase of $\lambda$ implies a decrease in df. Thus, choosing the degrees of freedom df small in order to gain a weak learner as proposed, for example, by Bühlmann & Hothorn (2007) (cf. Sec. 4.3.1) is supported by this finding.

Component-wise boosting uses different base-learners in each boosting iteration but only one of them is chosen adaptively (cf. Sec. 4.3.3). Bühlmann & Hothorn (2007) state that for component-wise $L_2$Boosting with linear least squares estimation, the hat matrix of a single base-learner is given by

$$\boldsymbol{\mathcal{S}}^{(j)} = \frac{\boldsymbol{x}^{(j)}(\boldsymbol{x}^{(j)})'}{\|\boldsymbol{x}^{(j)}\|^2}, \tag{4.23}$$

where the superscript $(j), j = 1, \ldots, p$ denotes that *only* the $j$-th predictor variable is used. $\|\boldsymbol{x}\|^2$ is the Euclidean norm $\boldsymbol{x}'\boldsymbol{x}$. Let $j_m^*$ denote the base-learner that is added in iteration $m$. Thus, the hat matrix of the base-learner in iteration $m$ is given by

$$\boldsymbol{\mathcal{S}}^{(j_m^*)} : (U_1^{[m]}, \ldots, U_n^{[m]})' \mapsto (\hat{U}_1^{[m]}, \ldots, \hat{U}_n^{[m]})'. \tag{4.24}$$

Bühlmann & Hothorn (2007) then derive the hat matrix of component-wise $L_2$Boosting in the $m$-th iteration:

$$\boldsymbol{\mathcal{B}}_m = \boldsymbol{\mathcal{B}}_{m-1} + \nu \cdot \boldsymbol{\mathcal{S}}^{(j_m^*)}(\boldsymbol{I} - \boldsymbol{\mathcal{B}}_{m-1}) = \boldsymbol{I} - \prod_{r=1}^{m}(\boldsymbol{I} - \nu \cdot \boldsymbol{\mathcal{S}}^{(j_r^*)}). \tag{4.25}$$

Bühlmann & Hothorn (2007) emphasize, that $\boldsymbol{\mathcal{B}}_m$ is depending on the response variable $\boldsymbol{y}$ via the selected base-learners $j_r^*, r = 1, \ldots, m$ and thus can only be seen as approximative hat matrix.

In likelihood-based boosting with component-wise base-learners, Tutz & Binder (2006) derive a hat matrix in iteration $m$ of the form

$$\boldsymbol{\mathcal{B}}_m = \sum_{j=0}^{m} \boldsymbol{M}_j \prod_{i=0}^{j-1}(\boldsymbol{I} - \boldsymbol{M}_i), \tag{4.26}$$

where the matrices $\boldsymbol{M}_j$ are given in the appendix of Tutz & Binder (2006). In the case of the same smoother matrix $\boldsymbol{\mathcal{S}}$ in each iteration (i.e., without component-wise base-learners) the hat matrix simplifies to

$$\boldsymbol{\mathcal{B}}_m = \sum_{j=0}^{m} \boldsymbol{\mathcal{S}}(\boldsymbol{I} - \boldsymbol{\mathcal{S}})^{j-1} = \boldsymbol{I} - (\boldsymbol{I} - \boldsymbol{\mathcal{S}})^{m+1} \tag{4.27}$$

which is, as Tutz & Binder (2006) state, equivalent to the form (4.22).

### 4.5.2. Degrees of Freedom

In all these cases, the (approximative) hat matrix can be used to define degrees of freedom. We follow the definition of effective degrees of freedom (Hastie & Tibshirani 1990b), which are defined as trace of the hat matrix. Thus, we get

$$\mathrm{df}(m) = \mathrm{trace}(\boldsymbol{\mathcal{B}}_m). \tag{4.28}$$

### 4.5.3. Definition of AIC

Using the above definitions of hat matrices (4.22), (4.25), (4.26) and (4.26), and degrees of freedom (4.28) we can define Akaike's information criterion for boosting iteration $m$ as

$$\mathrm{AIC}(m) = \mathrm{Dev}(\hat{f}^{[m]}) + 2 \cdot \mathrm{df}(m), \tag{4.29}$$

where $\mathrm{Dev}(\hat{f}^{[m]})$ is the deviance of the model in the $m$-th boosting iteration (Tutz & Binder 2006). We have to mention that the classical definition of the AIC

$$\mathrm{AIC}(m) = -2 \cdot l(\hat{f}^{[m]}) + 2 \cdot \mathrm{df}(m) \tag{4.30}$$

could also be used, as the deviance (4.21) can be written as

$$\text{Dev}(\hat{f}^{[m]}) = \phi\Big[-2 \cdot l(\hat{f}^{[m]}) + 2 \cdot l(\boldsymbol{y})\Big]. \tag{4.31}$$

We see that the two definitions of the AIC are just rescaled versions of each other. Hence, they are equivalent as they are just used for comparative purposes (within the analysis).

Moreover, Bühlmann (2006) proposes the use of a corrected version of the AIC (Hurvich et al. 1998) for Gaussian data:

$$\text{AIC}_{\text{corr}}(m) = \log(\hat{\sigma}^2) + \frac{1 + \text{df}(m)/n}{1 - (\text{df}(m) + 2)/n}, \tag{4.32}$$

where $n$ is the number of observations, the degrees of freedom are defined as in (4.28) and

$$\hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - (\boldsymbol{\mathcal{B}}_m \boldsymbol{y})_i)^2. \tag{4.33}$$

An estimate for the optimal number of boosting iterations $\widehat{m}_{\text{stop,opt}}$ is then given by

$$\widehat{m}_{\text{stop,opt}} = \underset{1 \leq m \leq m_{\text{stop}}}{\text{argmin}} \ \text{AIC}(m), \tag{4.34}$$

where $m_{\text{stop}}$ is the stopping iteration in the boosting algorithm, i.e., the upper bound for the candidate number of boosting iterations and $\text{AIC}(m)$ can be either the classical (4.30) or the bias corrected AIC (4.32) (Bühlmann 2006).

Using the AIC to determine the stopping iteration has shown to be superior to cross-validation in many settings. There are two reasons for this: First, it is computationally far less demanding. Compared to five-fold cross-validation, for example, the computational time to calculate the stopping iteration based on the AIC is almost five times less. Second, as for example Tutz & Binder (2006) showed, the MSE of models selected with five-fold cross-validation tends to be higher than the one obtained by AIC-based models. However, the AIC also has some drawbacks: The hat matrix is needed to compute the AIC, which is possibly difficult to derive. Furthermore, for large data sets the hat matrix becomes very large as it is an $n \times n$-matrix. Hence, it may be superior in the setting with many observations to use cross-validation to obtain an appropriate stopping iteration.

# 5. Boosting in Survival Models with Time-Varying Effects

Based on the theory from the preceding sections, we want to device an estimation procedure for Cox-type models with additive structure and possibly time-varying effects. Variable selection and model choice play another major role in this setting. To combine all tasks, component-wise boosting methods are applied in the following. Several boosting methods for additive survival models have already been proposed (cf. Sec. 4.2.4 and Sec. 4.4.2) but none is able to deal with time-varying effects.

The first part of this chapter (Sec. 5.2) deals with a functional gradient descent boosting approach, which shows serious problems and thus will not be regarded in the simulation studies nor be applied to the data set of surgical patients with severe sepsis. In Section 5.3, we develop a likelihood-based boosting approach, which we call $\text{Cox}_{\text{flex}}$Boost. We end the chapter with a discussion of the changes, compared to Tutz & Binder's (2006) approach, which we included in our algorithm. Computational details and a toy example that illustrates the usage of the software are presented in Appendix B.

## 5.1. Basic Considerations

Estimation of models can be done with respect to many different criteria. In the boosting context, minimization of a loss function (FGD boosting) or maximization of a likelihood-based criterion (likelihood-based boosting) is usually applied. In both cases, we base the estimation on the *full* log-likelihood (3.11) as introduced in Section 3.2. For FGD boosting, the negative log-likelihood is used as loss function. Likelihood-based boosting directly aims to maximize the log-likelihood. We only consider linear base-learners and P-spline base-learners in the following. For the latter base-learners, the optimization criterion is altered to the (negative) penalized log-likelihood as given in (3.14). Per default, the inner knots of the P-splines are equally spaced covering the range of the corresponding covariate. We only use 20 (inner) knots, as increasing the number computationally is quite demanding and empirically little is gained regarding the prediction performance (cf. Sec. 3.1.1). As we want to include variable selection, a component-wise boosting approach is used in the following. When

model choice is also desired, component-wise boosting with the decomposition of smooth functions as described in Section 4.3.4 is applied.

## 5.2. FGD Boosting for Survival Data

Functional gradient descent boosting for survival models with time-varying effects applies the generic FGD approach (Sec. 4.2.1) and uses the negative, full log-likelihood as loss function

$$\rho(y, f) = -\delta f(t, \boldsymbol{x}) + \int_0^t \exp(f(u, \boldsymbol{x}))du, \tag{5.1}$$

with the indicator for non-censoring $\delta$, the observed survival time $t$, the combination of both $y = (t, \delta)$ and $\boldsymbol{x}$ the observed covariates. The generic function $f(t, \boldsymbol{x})$ is equivalent to the additive predictor $\eta(t)$ as in the Cox-type additive model (2.11). Following the generic representation (4.13) of the additive predictor we see that $f(t, \boldsymbol{x})$ is *potentially* dependent on time $t$ and covariates $\boldsymbol{x}$, with smooth effects for some components of $\boldsymbol{x}$, time-varying effects for others and linear effects for the remainder.

### 5.2.1. Problems and Considerations

For functional gradient descent boosting we need the negative derivative of the loss function

$$-\frac{\partial}{\partial f}\rho(y, f) \tag{5.2}$$

evaluated at the function estimate of $\hat{f}^{[m-1]}$ of the previous iteration. This means, we compute the functional derivative of $\rho$. Then we plug in $\hat{f}^{[m-1]}$ which itself is again dependent on $\boldsymbol{x}$ and in our case of $t$. Hence, we have to use a functional derivative such as the Hadamard derivative (see e.g., van der Vaart 1998):

**Definition 5.1** *A function $\rho : \mathbb{D}_\rho \mapsto \mathbb{E}$, defined on a subset $\mathbb{D}_\rho$ of a normed space $\mathbb{D}$ that contains $f$, is called Hadamard differentiable at $f$ if a continuous, linear map $\rho'_f : \mathbb{D} \mapsto \mathbb{E}$ exists such that*

$$\lim_{s \searrow 0} \left\| \frac{\rho(f + sh_s) - \rho(f)}{s} - \rho'_f(h) \right\|_{\mathbb{E}} = 0, \, \forall h_s \to h, s \in \mathbb{R}. \tag{5.3}$$

In contrast to the Gateaux derivative, the directions $h_s$ of differentiation are not fixed but may change with changing $s$ but eventually they converge. Applying

(5.3), one can show (see Appendix A) that the Hadamard derivative $\rho'_f(h)$ is equivalent to

$$\rho'_f(h) = -\delta h(\boldsymbol{x}, t) + \int_0^t h(\boldsymbol{x}, u) \exp(f(u, \boldsymbol{x}))du, \tag{5.4}$$

where $h(\boldsymbol{x}, t)$ denotes the direction depending on $\boldsymbol{x}$ and $t$. The question in which direction the derivative should be computed cannot be answered so easily. A reasonable but not compulsory direction is the direction induced by the observed data $(\boldsymbol{x}_i, t_i)$

$$h(\boldsymbol{x}, t) = I_{\{\boldsymbol{x}_i\} \times [0, t_i]}(\boldsymbol{x}, t), \tag{5.5}$$

where $I$ is the indicator function. Bühlmann & Hothorn (2007) also derive the empirical loss function in the direction of the observed data for the formulation of boosting in function space. However, in their case, the observed data only consists of the covariates $\boldsymbol{x}_i$ and no additional direction for time $t$ is needed. In our case the direction becomes

$$h(\boldsymbol{x}, t) = \begin{cases} 1 & \text{if } \boldsymbol{x} = \boldsymbol{x}_i \wedge t \in [0, t_i] \\ 0 & \text{otherwise} \end{cases} . \tag{5.6}$$

This leads to a derivative that is equivalent to the naïve derivative where $f$ is treated as a constant. The negative gradient thus is

$$-\frac{\partial}{\partial f}\rho(y, f) = \delta - \int_0^t \exp(f(u, \boldsymbol{x}))du. \tag{5.7}$$

It shows that (5.7) has the form of a martingale residual (cf. Barlow & Prentice 1988, Therneau et al. 1990). Thus, FGD boosting based on the full likelihood simplifies to refitting martingale residuals. This can be seen as a nice and interpretable analogy to $L_2$Boosting, where the negative gradient vector reduces to the classical residual vector (see Sec. 4.2.3).

Nevertheless, this approach proves to be incapable of estimating time-varying effects. Possible reasons for this are discussed in the following section.

### Residuals in Survival Models

In survival models a vast number of different residuals is known, each with different applications (see e.g., Therneau & Grambsch 2000): The basic martingale residual is used to asses the model directly and for investigation of the functional form. The deviance residual is based on the martingale residual and was developed to reveal individual outliers. To asses individual influence score residuals are proposed.

Schoenfeld residuals (Schoenfeld 1982) were introduced to assess the proportional hazards assumption. The residuals $\boldsymbol{r}_i, i = 1, \ldots, n$ are constructed such

that they do not depend on time and hence can be plotted (for each variable separately) against time $t_i$ to examine the proportional hazards assumption. From these residuals we see that there should be no built-in time-dependency in the residuals that are used to examine proportional hazards (which is very similar to the estimation of time-varying effects). This gives a first hint that martingale residuals are not suitable for assessing time-varying effects.

### Investigation of the Negative Gradient Vector

Let us have a closer look at the negative gradient vector for observation $i$ in the $m$-th boosting iteration

$$U_i^{[m]} = - \left. \frac{\partial \rho(y_i, f)}{\partial f} \right|_{f=\hat{f}^{[m-1]}(t_i, \boldsymbol{x}_i)} = \delta_i - \int_0^{t_i} \exp(\hat{f}(t, \boldsymbol{x}_i)) dt. \tag{5.8}$$

Note that we integrate over $t$ from 0 to $t_i$ in (5.8). Thus, as long as we integrate over a positive function the subtrahend is monotonically increasing in $t_i$, i.e., as $t_i$ increases the integral increases as well. This is trivially true, as the integrand is an exponential function and hence always greater than zero (or equal to zero only if $\hat{f}(t, \boldsymbol{x}_i) = -\infty$). Therefore, from (5.8) follows that $U_i^{[m]}$ is negatively correlated with $t_i$ in each iteration $m$. This holds for time-constant functions $\hat{f}(t, \boldsymbol{x}_i) = \hat{f}(\boldsymbol{x}_i)$ as well as for time-dependent functions $\hat{f}(t, \boldsymbol{x}_i)$. In the latter case, the true dependency on time only plays a negligible role in the observed dependency of $U_i^{[m]}$ on time $t_i$.

### Toy Example

To gain some insight and verify the previous statements, we look at a simplified toy example: We assume that only one variable $\boldsymbol{x} = (x_1, \ldots, x_n)'$ and time $\boldsymbol{t} = (t_1, \ldots, t_n)'$ have an (arbitrary) effect on the survival time. We use component-wise FGD boosting where we utilize the negative gradient vector (5.8) and only use linear base-learners, one for time $\boldsymbol{t}$ and one for $\boldsymbol{x}$, i.e., we assume log-linear dependencies of the hazard rate on $\boldsymbol{t}$ and $\boldsymbol{x}$.

**First Step: Offset** No matter how the offset is chosen, either by minimizing the empirical risk or by setting it to zero a priori, it is just a constant. Thus, further considerations just depend on the value itself. If we choose the offset to

be $\hat{f}^{[0]} \equiv 0$, the negative gradient vector $\boldsymbol{U}^{[1]}$ is initialized as:

$$\begin{aligned} U_i^{[1]} &= \delta_i - \int_0^{t_i} \exp(0) dt = \\ &= \delta_i - \int_0^{t_i} 1 \, dt = \\ &= \delta_i - t_i = \\ &= \begin{cases} -t_i & \text{if } \delta_i = 0 \\ 1 - t_i & \text{if } \delta_i = 1 \end{cases} \end{aligned} \tag{5.9}$$

More general, if one chooses the offset $\hat{f}^{[0]} \equiv c$ the negative gradient becomes

$$U_i^{[1]} = \begin{cases} -t_i \cdot \tilde{c} & \text{if } \delta_i = 0 \\ 1 - t_i \cdot \tilde{c} & \text{if } \delta_i = 1 \end{cases}, \text{ where } \tilde{c} = \exp(c). \tag{5.10}$$

In both cases (5.9) and (5.10), $\boldsymbol{U}^{[1]}$ primarily depends on the time $\boldsymbol{t}$. In the case of an offset equal to zero, all censored observations lie on the negative bisecting (half-) line and all uncensored observations lie on the negative bisector shifted up by one. Having an offset not equal to zero, only the slope of the two (half-) lines is changed from one to $\tilde{c}$. This shows that one has an almost perfect linear dependency of $\boldsymbol{U}^{[1]}$ from $\boldsymbol{t}$ and thus, in component-wise boosting, a (linear) base-learner for $\boldsymbol{t}$ is selected.

**Second Step: Negative Gradient in Subsequent Boosting Iterations (Special Case)** In the following boosting iterations we get almost the same picture. Analytically the negative gradient can be computed quite easily, if one assumes that in *all* previous iterations the linear base-learner for time $\boldsymbol{t}$ was selected. As stated above, in the first iteration this will hold almost certainly. At least in early subsequent iterations, this can also be assumed for the same reasons, as we will show in the following paragraph.

Let the estimated coefficient of the linear base-learner for time $\boldsymbol{t}$ in iteration $m$ be denoted by $\hat{\boldsymbol{\beta}}^{[m]} = (\hat{\beta}_0^{[m]}, \hat{\beta}_1^{[m]})'$, where $\hat{\beta}_0^{[m]}$ is the intercept and $\hat{\beta}_1^{[m]}$ is the slope. We then define

$$\hat{\alpha}^{[m]} := \sum_{j=1}^{m-1} \hat{\beta}_0^{[j]} \quad \text{and} \quad \hat{\gamma}^{[m]} := \sum_{j=1}^{m-1} \hat{\beta}_1^{[j]}. \tag{5.11}$$

For simplicity we assume that the offset $\hat{f}^{[0]} \equiv 0$. Thus, in boosting iteration $m$

we get

$$\begin{aligned} U_i^{[m]} &= \delta_i - \int_0^{t_i} \exp\left\{\hat{\alpha}^{[m]} + \hat{\gamma}^{[m]} \cdot t\right\} dt \\ &= \delta_i - \left[\left(\hat{\gamma}^{[m]}\right)^{-1} \cdot \exp\left\{\hat{\alpha}^{[m]} + \hat{\gamma}^{[m]} \cdot t\right\}\right]_0^{t_i} \\ &= \delta_i - \left(\frac{\exp\left\{\hat{\alpha}^{[m]}\right\}}{\hat{\gamma}^{[m]}} \cdot \left[\exp\left\{\hat{\gamma}^{[m]} \cdot t_i\right\} - 1\right]\right), \end{aligned} \tag{5.12}$$

where $\left[F(t)\right]_0^{t_i} := F(t_i) - F(0)$ for an arbitrary antiderivative $F(t)$. Equation (5.12) shows that the negative gradient essentially is an exponential function dependent on $t_i$. More precisely, we get two negative exponentials scaled with the coefficient $\exp\left\{\hat{\alpha}^{[m]}\right\} / \hat{\gamma}^{[m]}$, where one has an intersection with the ordinate in 0 and the other in 1 for censored and uncensored observations, respectively. Thus, again we have a function strongly dependent on $t_i$ which is decreasing in $t_i$, no matter how the true relation is. This leads to further selections of the linear base-learner of time $t$ with negative coefficient for the slope.

### Simulation Results

To show that these results also empirically hold, we set up a small simulation study. Survival times with a baseline hazard rate

$$\lambda(t, x) = \exp\{0.001 \cdot t + 3 \cdot x\} \tag{5.13}$$

where simulated (for details on survival time simulations see Section 6.1.1). Censoring times where drawn independently from an Expo($\nu$) distribution, with $\nu$ such that one gets roughly 90% uncensored data.

Afterwards, we tried to estimate the model using component-wise FGD boosting where we applied the negative gradient (5.8) and only used linear base-learners. Initially 5000 boosting iterations were performed to see if the problems discussed above might vanish in the long run. Every 10th iteration, we plotted $t$ and $x$ both versus $\boldsymbol{U}^{[m]}$. A selection of these plots is shown in Figure 5.1.

For $U^{[1]}$ we see in the leftmost graphic the two half-lines for the observations as derived above in Equation (5.10). We clearly see that the censoring rate is quite low, as more observations are located on the upper half-line. Thus, estimation of the effect of time (solid line) is mainly influenced by the non-censored observations. Note that the true effect of $t$ is much smaller (i.e., 0.001) than the estimated effect and in addition it should be positive. The second picture (from the left), in which the association of $U^{[1]}$ and $x$ is depicted, shows quite a good estimation of the effect of $x$. The slope roughly is 2. Thus, it is estimated a bit smaller than true effect but the sign and the magnitude are

estimated approximately right. Even so, the sum of squared errors (SSE) for $t$ is much smaller than for $x$. Hence, the wrongly estimated base-learner for $t$ is chosen to enter the model (multiplied by the step-length factor $\nu = 0.1$). In the upper right panel, for $U^{[11]}$, we see the two negative exponentials as presented in (5.12). Again, the estimation of the effect of $t$ is mainly influenced by the uncensored observations. The picture for the influence of $x$ is practically the same as in the first iteration. Again the linear base-learner of time $t$ is selected to enter the model.

In the middle left panel, after 40 iterations, nothing has changed much. The only differences are that the curvature of the negative exponentials is increasing and, more importantly, we now have a situation where the linear base-learner for $x$ is selected, as the SSE of $x$ is smaller. Note that in this simulation the base-learner for $x$ was almost never selected. With $x$ entering the model, the picture changes as we can see in the right panel in the middle ($m = 51$). Now the two clearly separated lines for censored and uncensored observations become blurred. This is due to the influence of $x$ which now becomes visible. Nevertheless, the negative influence of $t$ remains.

The last two graphics in the bottom row depict that, with an increasing number of iterations, the estimated effects decrease. The effect of $t$ is still estimated to be negative. This did not change up until the last iteration ($m = 5000$), where the effect of $t$ was tiny but still slightly negative (not depicted). We can also see that both sums of squared errors are decreasing but the SSE for $t$ is (almost) always a little bit smaller than that of $x$.

In principal, for other baseline hazards we get the same results. Choosing a higher weight for $x$ or reducing the weight for $t$ only leads to a higher selection frequency of $x$. Anyhow, $t$ is always selected with a (wrongly) negative coefficient.

Another problem with martingale residuals as response in least squares estimation (as used in FGD boosting) could be the skewness of the distribution of martingale residuals. Least squares estimation is based on symmetrically and even normally distributed errors. Moreover, due to the skewness, the quadratic loss function might be misleading as selection criterion for the "best" base-learner in each iteration. Thus, both, estimation and selection of base-learners, could be improved when we employ other estimation and selection criteria.

### 5.2.2. Conclusion

Using the generic functional gradient descent approach with a loss function based on the full likelihood for survival models with time-dependent effects seems relatively easy at first sight. However, taking a closer look, some problems arise: First, the derivative of the negative log-likelihood is not just a simple

derivative but it is a functional derivative such as the Hadamard derivative with special directions of derivation. This leads to a negative gradient vector that is equivalent to the martingale residuals. Here we get a nice parallel to $L_2$Boosting, where classical residuals are refitted. Inspecting the martingale residuals in their role as response we find the second problem: Martingale residuals do not seem to allow the estimation of time-varying effects. On the contrary, the built-in negative correlation of the martingale residuals with time makes it impossible to estimate time-varying effects. Even the correct sign of a possible effect cannot be identified, far-off from assessing the right functional form.

As one possible solution to overcome these problems, we can use likelihood-based boosting, where we do not compute and fit the negative gradient but try to maximize the likelihood directly. Using the penalized full log-likelihood (3.14) to fit models has shown to work well in other applications with classical Newton-Raphson estimation procedures (see e.g., Kneib & Fahrmeir 2007). This lets us assume that maximizing this log-likelihood might also work in the boosting framework.

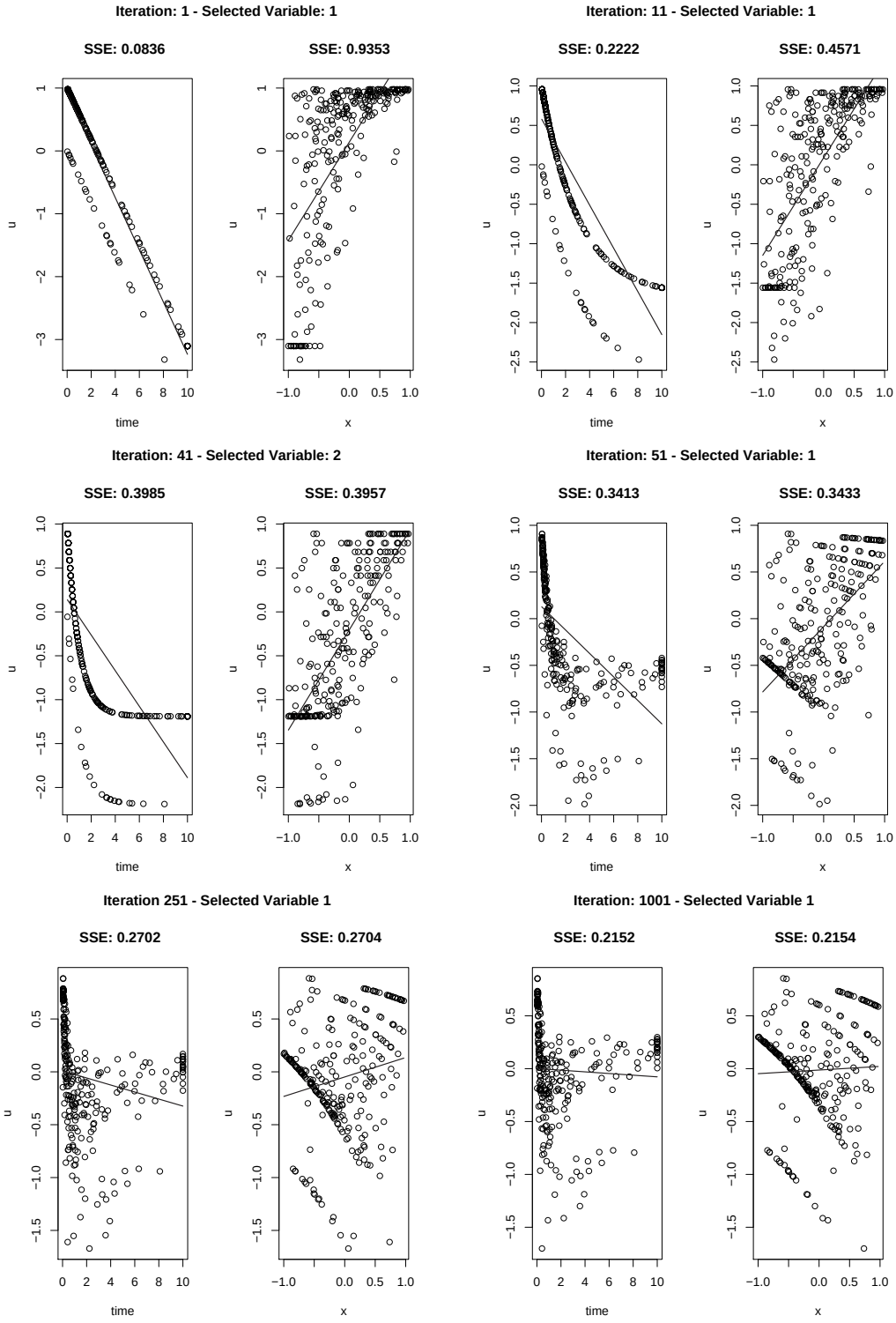


Figure 5.1.: **FGD Boosting:** Negative gradient vectors $\boldsymbol{U}^{[m]}$ plotted against time $t$ and $x$, for $m = 1$ (upper left), $m = 11$ (upper right), $m = 41$ (middle left), $m = 51$ (middle right), $m = 251$ (lower left), and $m = 1001$ (lower right), together with the estimated linear base-learner (solid line) and the sum of squared errors (SSE). Note that the base-learner with the lower SSE is selected.

## 5.3. Likelihood-Based Boosting for Survival Data ($\text{Cox}_{\text{flex}}$Boost)

As in the FGD boosting approach, we base the estimation on the full log-likelihood (3.11). If we are using P-spline base-learners, which will be the standard case, the log-likelihood is replaced by the penalized log-likelihood (3.14). In contrary to FGD boosting, the log-likelihood is not used as basis for the loss function but we directly aim to maximize it. The boosting algorithm, which we will present in the following section, is essentially based on the likelihood-based boosting approach as proposed by Tutz & Binder (2006) (cf. Sec. 4.4). As we specially focus on the inclusion of flexible and time-varying terms in Cox-type additive models, we call the new algorithm $\text{Cox}_{\text{flex}}$Boost.

In the following, we denote the $j$-th base-learner by $g_j(\boldsymbol{x}(t); \boldsymbol{\beta}_j)$. The base-learner can be seen as a generic representation for different types of functions (cf. Sec. 4.3.4). The covariates $\boldsymbol{x}(t)$ include classical covariates and possible time-varying effects expressed as artificial time-dependent covariates or the time $t$ itself. The notation $\boldsymbol{x}(t)$ for the covariates stresses the possible dependence on time. Thus, $g_j(\boldsymbol{x}(t); \boldsymbol{\beta}_j)$ can correspond to a linear function of $\tilde{x} \in \boldsymbol{x}(t)$ or of time $t \in \boldsymbol{x}(t)$ or, more flexible, a smooth function of $\tilde{x}$ or $t$. Moreover, time-varying effects, expressed as varying coefficients as in Section 2.3.2, can be represented by the generic base-learner $g_j(\boldsymbol{x}(t); \boldsymbol{\beta}_j)$, where the effect of time $t$ is either a linear or a flexible function. With this notation at hand, we can derive the $\text{Cox}_{\text{flex}}$Boost algorithm:

### 5.3.1. $\text{Cox}_{\text{flex}}$Boost Algorithm

(i) **Initialization:** Set the iteration index $m := 0$.

a) Initialize the function estimates

$$\hat{f}_j^{[0]}(\cdot) \equiv 0.$$

b) Initialize the additive predictor $\hat{\eta}^{[0]}$ with the maximizer of the log-likelihood of the intercept model, as offset value

$$\hat{\eta}^{[0]}(\cdot) \equiv \underset{c}{\text{argmax}} \sum_{i=1}^{n} \delta_i \cdot c - \exp(c) \cdot t_i,$$

i.e., with the maximum likelihood estimate for a constant log-hazard:

$$\hat{\eta}^{[0]}(\cdot) \equiv \log\left(\frac{\sum_{i=1}^{n} \delta_i}{\sum_{i=1}^{n} t_i}\right)$$

(ii) **Estimation:** Increase $m$ by 1. Fit all (linear and/or P-spline) base-learners

$$\hat{g}_j = g_j(\cdot\,; \hat{\boldsymbol{\beta}}_j),\ \forall j \in \{1, \ldots, J\},$$

determined by penalized maximum likelihood estimation

$$\hat{\boldsymbol{\beta}}_j = \underset{\boldsymbol{\beta}}{\text{argmax}}\, l^{[m]}_{\text{pen},j}(\boldsymbol{\beta}),$$

with the penalized log-likelihood (cf. Eq. (3.14))

$$\begin{aligned} l^{[m]}_{\text{pen},j}(\boldsymbol{\beta}) \;=\; & \sum_{i=1}^{n} \Bigg[ \delta_i \cdot \left( \hat{\eta}_i^{[m-1]}(\boldsymbol{x}_i(t_i)) + g_j(\boldsymbol{x}_i(t_i); \boldsymbol{\beta}) \right) \\ & \qquad - \int_0^{t_i} \exp\left\{ \hat{\eta}_i^{[m-1]}(\boldsymbol{x}_i(\tilde{t})) + g_j(\boldsymbol{x}_i(\tilde{t}); \boldsymbol{\beta}) \right\} d\tilde{t} \Bigg] \\ & - \text{pen}_j(\boldsymbol{\beta}), \end{aligned} \tag{5.14}$$

where $\text{pen}_j(\boldsymbol{\beta})$ is the difference penalty as in (3.6) for the $j$-th base-learner, or $\text{pen}_j(\boldsymbol{\beta}) = 0$ if the corresponding base-learner is unpenalized (i.e., here: a linear base-learner).

(iii) **Selection:** Choose the base-learner $\hat{g}_{j^*}$ that maximizes the *unpenalized* log-likelihood (cf. Eq. (3.11))

$$j^* = \underset{j \in \{1,\ldots,J\}}{\text{argmax}}\, l^{[m]}_j(\hat{\boldsymbol{\beta}}_j),$$

where

$$\begin{aligned} l^{[m]}_j(\hat{\boldsymbol{\beta}}_j) \;=\; & \sum_{i=1}^{n} \Bigg[ \delta_i \cdot \left( \hat{\eta}_i^{[m-1]}(\boldsymbol{x}_i(t_i)) + g_j(\boldsymbol{x}_i(t_i); \hat{\boldsymbol{\beta}}_j) \right) \\ & \qquad - \int_0^{t_i} \exp\left\{ \hat{\eta}_i^{[m-1]}(\boldsymbol{x}_i(\tilde{t})) + g_j(\boldsymbol{x}_i(\tilde{t}); \hat{\boldsymbol{\beta}}_j) \right\} d\tilde{t} \Bigg] \end{aligned} \tag{5.15}$$

(iv) **Update:**

a) Compute the update for the function estimate of the selected base-learner

$$\hat{f}^{[m]}_{j^*}(\cdot) = \hat{f}^{[m-1]}_{j^*}(\cdot) + \nu \cdot \hat{g}_{j^*}(\cdot)$$

and set $\hat{f}^{[m]}_j(\cdot) = \hat{f}^{[m-1]}_j(\cdot)$ otherwise (i.e., for $j \neq j^*$).

b) Compute the update for the additive predictor

$$\hat{\eta}^{[m]}(\cdot) = \hat{\eta}^{[m-1]}(\cdot) + \nu \cdot \hat{g}_{j^*}(\cdot).$$

We classically choose the step-length factor $\nu = 0.1$ but, in general, it is sufficient to choose $\nu \in (0, 1]$ small enough.

(v) **Stopping rule:** Continue iterating steps (ii) to (iv) until $m = m_{\text{stop}}$.

Note that the estimated additive predictor from the previous iteration is treated as an offset in the first part of the formulas (5.14) and (5.15) and it is possibly time-dependent in the integral. The term $\hat{\eta}_i^{[m-1]}(\boldsymbol{x}_i(\tilde{t}))$ in the integral can be interpreted in such a way that the estimated parameters and the time-constant covariates of the base-learners are kept fixed and the time $\tilde{t}$ stays variable. Hence, the integrand in (5.14) is a function depending on the coefficient $\boldsymbol{\beta}$, which we try to estimate, and (possibly) on time $\tilde{t}$. In (5.15), we use the estimates $\hat{\boldsymbol{\beta}}_j$ from step (ii). Thus, we only have a function that (possibly) depends on time $\tilde{t}$.

## 5.3.2. Problems and Considerations

We have to integrate over time $\tilde{t}$ for each base-learner, in each boosting iteration *and* in each step of the optimization method (cf. App. B.1) used to determine $\hat{\boldsymbol{\beta}}_j$. Hence, the estimation step (ii), or more precisely the integrations therein, are the computational bottleneck of the algorithm. Accelerating the integration method can increase the speed of $\text{Cox}_{\text{flex}}$Boost dramatically. We will discuss this in more detail in Appendix B.1.1.

Computationally, the likelihood-based boosting approach has some drawbacks compared to the FGD approach. In the following enumeration, we will discuss some of the problems that arise and present possible solutions.

(a) Tutz & Binder (2006) use a one-step Fisher scoring estimate (4.20) for each base-learner in each boosting iteration. Instead of this estimate, we use a full maximum likelihood estimate and apply a step-length factor $\nu$ as proposed in the FGD boosting literature (e.g., Bühlmann & Hothorn 2007). This can possibly be computationally more intensive but we get an estimate that is "weakened" or "shrunken" with the same relative amount $\nu$ for all elements of the coefficient vector of the base-learner. Different amounts of shrinkage for one-step Fischer scoring may especially occur when competing base-learners with different numbers of parameters are used (e.g., linear base-learners versus P-spline base-learners). This might result in a biased selection of model terms if we use the model choice scheme from Section 4.3.4.

(b) Tutz & Binder (2006) specify the smoothness of the base-learners using the smoothing parameter $\lambda$. They propose to choose $\lambda$ very large in order to obtain a weak learner. Only one single smoothing parameter is used for all base-learners which is chosen relatively crude (see Sec. 4.4.1). However, we believe that specifying the degrees of freedom df to determine the amount smoothness of each base-learner (separately) is much more intu-

itive. Especially when model choice, as introduced in Section 4.3.4, should be integrated in the boosting algorithm, we need to be able to define each base-learner in such a way that its complexity (in terms of df) is comparable to that of other model terms.

In FGD boosting approaches it is relatively easy to use initial degrees of freedom $\widetilde{df}_j$ to compute the corresponding smoothing parameter $\lambda_j$. As the base-learners are fitted by (penalized) least squares estimation to the negative gradient vector, the smoother matrix (dependent on $\lambda_j$) always has the form

$$\boldsymbol{\mathcal{S}}^{(j)}(\lambda_j) = \boldsymbol{X}_j[\boldsymbol{X}_j'\boldsymbol{X}_j + \lambda_j \boldsymbol{K}_j]^{-1}\boldsymbol{X}_j' \tag{5.16}$$

where $\boldsymbol{X}_j$ denotes the design matrix for the $j$-th base-learner and $\boldsymbol{K}_j$ is the corresponding penalty matrix. Thus, we can use the equivalent degrees of freedom

$$\text{df}(\lambda_j) = \text{trace}(\boldsymbol{\mathcal{S}}^{(j)}(\lambda_j)) \tag{5.17}$$

to solve

$$\text{df}(\lambda_j) - \widetilde{df}_j \stackrel{!}{=} 0 \tag{5.18}$$

for $\lambda_j$ with a pre-specified value of $\widetilde{df}_j$. For a given penalty matrix (i.e., $\boldsymbol{K}_j$ fixed), the smoother matrix (5.16) and thus the degrees of freedom (5.17) in the FGD boosting algorithm only depend on the design matrix $\boldsymbol{X}_j$ (and on $\lambda_j$).

For survival models in likelihood based-boosting, least squares estimation is replaced by Fisher scoring (or similar likelihood optimization methods). Therefore, the equivalent degrees of freedom are computed as in Equation (3.8). The connection of the degrees of freedom $\text{df}_j$ of the $j$-th base-learner and the smoothing parameter $\lambda_j$ can be derived from Equation (3.9):

$$\text{df}(\lambda_j) = \text{trace}\left[\boldsymbol{F}_j^{[m]}(\boldsymbol{F}_j^{[m]} + \lambda_j \boldsymbol{K})^{-1}\right]. \tag{5.19}$$

The Fisher matrix of the base-learner $j$ in the $m$-th boosting iteration $\boldsymbol{F}_j^{[m]}$ (see Eq. (3.13)) depends on the design matrix and, at the same time, on the hazard rate $\lambda(\cdot) = \exp(\hat{\eta}^{[m-1]}(\cdot) + g_j(\cdot; \boldsymbol{\beta}_j))$. The additive predictor of the previous iteration $\hat{\eta}^{[m-1]}(\cdot)$ is treated as an offset, i.e., with fixed coefficients, whereas the base-learner $g_j(\cdot; \boldsymbol{\beta}_j)$ is a function of the covariates and the coefficients $\boldsymbol{\beta}_j$. Hence, the estimated degrees of freedom (5.19) do not only depend on the design matrix, the order of the penalty and the smoothing parameter $\lambda_j$ but also on the coefficients $\boldsymbol{\beta}_j^{[m]}$. Thus, calculating the smoothing parameter $\lambda_j$ as in (5.18) is only possible for given (estimated) coefficients $\hat{\boldsymbol{\beta}}_j^{[m]}$ but estimating the coefficients is only possible if $\lambda_j$ is known. At the moment, this problem is solved by using (5.18) where $\text{df}(\lambda_j)$ is estimated as in (5.16) to find an approximative solution. This allows us to specify the smoothing parameter $\lambda_j$ (at least crudely) using

initial degrees of freedom $\widetilde{df}_j$. As we will show in Section 6.1.6, this is – to a certain extent – a reasonable approximation.

A possible refinement could be to estimate an initial smoothing parameter $\hat{\lambda}_j$ using the hat matrix (5.16) from the least squares estimation as just described. With this value, initial estimations for the coefficients can be computed which in turn can be used to compute a new, better approximation for $\lambda_j$ using the appropriate degrees of freedom (5.19). This smoothing parameter $\lambda_j$ can now be used in the first boosting iteration.

(c) Another problem that emerges for likelihood-based boosting is that the specification of a constant smoothing parameter $\lambda_j$ for the base-learner $g_j(\cdot, \boldsymbol{\beta}_j)$ does not correspond to a fixed amount of smoothness for this base-learner. With an increasing number of iterations $m$, the degrees of freedom $\text{df}_j^{[m]}$ for $g_j(\cdot, \boldsymbol{\beta}_j)$ change, as we will show in our simulation studies (see Sec. 6.1.6). However, this effect is not very strong. Over numerous boosting iterations $m$, only minor changes of the estimated degrees of freedom $\text{df}_j^{[m]}$ of the $j$-th base-learner are observed. Thus, again we propose to use the above approximation (see (b)) to specify the *initial* degrees of freedom $\text{df}_j^{[0]}$ approximately equal to $\widetilde{df}_j$ and ignore the (small) changes with increasing iterations.

Again we could think of a correction. We could readjust the smoothing parameter $\lambda_j$ in each (or each $k$-th) iteration such that we get again the desired degrees of freedom. This would lead to an increased computational burden. As we could observe only minor deviations and as the degrees of freedom (3.8) are just an approximation themselves, readjusting $\lambda_j$ does not seem to be necessary.

(d) A last problem considered here, is the computation of the AIC, which is much more complicated for likelihood-based boosting. As we have shown in Section 4.5.1, for FGD boosting we have one single and simple formula (4.25) to estimate the hat matrix of the model in boosting iteration $m$. This is due to the same estimation procedure, i.e., (penalized) least squares, which is (usually) common to all boosting algorithms in the generic FGD framework (Sec. 4.2.1). For likelihood-based boosting, the hat matrix $\boldsymbol{\mathcal{B}}_m$ in iteration $m$ must be derived anew for each estimation procedure. Tutz & Binder (2006) deduced the hat matrix for the GAMBoost procedure, where all simple exponential families are considered. Deriving an approximative solution for the smoother matrix of $\text{Cox}_{\text{flex}}$Boost, and hence having the AIC available, will be subject to future research. As stated above, the $\text{Cox}_{\text{flex}}$Boost algorithm is computationally quite demanding. Therefore, choosing the stopping iteration by cross-validation is generally infeasible.

In this thesis, as we mainly focus on simulation studies, we use a validation data set, as described in Section 4.5, to compute the (unpenalized) log-likelihood criterion (3.11). An appropriate stopping iteration is deter-

mined as the number of boosting iterations $\widehat{m}_{\text{stop,opt}}$ that maximizes the log-likelihood for the validation data.

We see from item (b) that we can use initial degrees of freedom to get an approximative value for $\lambda_j$. Even if we have a slight misspecification, this is more intuitive than defining the smoothing parameter itself. Moreover, this allows us to use the model choice scheme as proposed by Kneib et al. (2007) (cf. Sec. 4.3.4). As the problem of changing degrees of freedom (c) is not that strong, the different model terms stay roughly comparable even in advanced boosting iterations. In the next chapter, we want to support these statements with simulation studies.

# 6. Simulations and an Application

## 6.1. Simulations

In the simulation study that is presented in the following section, we focus on $\text{Cox}_{\text{flex}}$Boost. The two-stage stepwise procedure is not examined due to the high computational burden and the need of the user to interact after each step of the model choice procedure. The required interaction is a result of the current implementation of the procedure (based on R and BayesX at the same time), which is likely to be changed in the future.

### 6.1.1. Simulating Survival Data

Before we can conduct simulation studies, we need to be able to sample survival data according to a known hazard rate $\lambda(t, \boldsymbol{x})$. "Standard" data, satisfying the proportional hazards assumption, is relatively easy to sample, especially if one sticks to data sampled from known parametric distributions such as the exponential or the Weibull distribution.

As we mainly focus on flexible and time-varying effects, we need a more general framework for survival data simulations. Bender et al. (2005) present an algorithm for simulation of survival times according to Cox proportional hazards models with a wider range of distributions.

Using the notation from Section 2.1, one starts with the distribution function of the Cox model

$$\begin{aligned} F(\tilde{t}, \boldsymbol{x}) &= 1 - \exp\left\{-\Lambda(\tilde{t}, \boldsymbol{x})\right\} = \\ &= 1 - \exp\left\{-\int_0^{\tilde{t}} \lambda(t, \boldsymbol{x}) dt\right\} = \\ &= 1 - \exp\left\{-\int_0^{\tilde{t}} \exp(\eta(t, \boldsymbol{x})) dt\right\}, \end{aligned} \tag{6.1}$$

where $\lambda(\cdot, \boldsymbol{x}) = \exp(\eta(\cdot, \boldsymbol{x}))$ is the hazard rate as in (2.11) with an additive predictor $\eta$ as in (2.12). Note that the cumulative hazard rate $\Lambda(\cdot, \boldsymbol{x})$ includes the baseline hazard rate. It can be shown that a cumulative distribution function

is always uniformly distributed on the interval $[0, 1]$, in short $F \sim U[0, 1]$ (see e.g., Mood et al. 1974). So, with $T$ as a random variable, one can write

$$1 - \exp\{-\Lambda(T, \boldsymbol{x})\} =: U \sim U[0, 1]. \tag{6.2}$$

For symmetry reasons the same property holds for the survivor function $S(\cdot, \boldsymbol{x})$:

$$\begin{aligned} S(\tilde{t}, \boldsymbol{x}) &= 1 - F(\tilde{t}, \boldsymbol{x}) = \\ &= \exp\left\{ - \int_0^{\tilde{t}} \exp(\eta(t, \boldsymbol{x}))dt \right\}. \end{aligned} \tag{6.3}$$

We define $\tilde{U} := \exp\{-\Lambda(T, \boldsymbol{x})\}$, where $\tilde{U} \sim U[0, 1]$. If $\Lambda(\cdot, \boldsymbol{x})$ is invertible one obtains with the inversion method (see e.g., Devroye 1986)

$$T = \Lambda^{-1}(-\log(\tilde{U}), \boldsymbol{x}) \sim F(\cdot, \boldsymbol{x}). \tag{6.4}$$

Therefore, if $\tilde{U}$ is drawn from a $U[0, 1]$ distribution, the survival time $T$ derived from (6.4) is distributed with respect to the distribution function $F(\cdot, \boldsymbol{x})$. Equation (6.4) is equivalent to

$$- \int_0^T \exp(\eta(t, \boldsymbol{x}))dt = \log \tilde{U} \tag{6.5}$$

$$\Leftrightarrow \quad \underbrace{\int_0^T \exp(\eta(t, \boldsymbol{x}))dt}_{=\Lambda(T, \boldsymbol{x})} = -\log \tilde{U} \tag{6.6}$$

$$\Leftrightarrow \quad \underbrace{-\log \tilde{U} - \Lambda(T, \boldsymbol{x})}_{=:h(T)} = 0. \tag{6.7}$$

Hence, we are looking for the root of $h(T)$ for a given realization drawn from a $U[0, 1]$ distribution. This can be calculated in `R` using

```
R> uniroot(h, c(0, t_upper))$root
```

where `h` implements the function $h(T)$ as in (6.7) and `c(0, t_upper)` specifies an interval, which contains the desired root of $h(T)$. Solving (6.7) to obtain samples of $T$ can be seen as an extended version of the algorithm proposed by Bender et al. (2005) where only classical Cox models are addressed. Equation (6.7) allows the use of *arbitrary* hazard rates $\lambda(\cdot, \boldsymbol{x})$ and to draw data from the corresponding distribution. The hazard rate is specified as the exponential of an additive predictor $\eta(\cdot, \boldsymbol{x})$ as in the Cox-type additive model (see Sec. 2.3).

Censoring can be introduced by simulating, for example, exponentially distributed censoring times $C_i$. The censoring distribution, respectively the parameters of the censoring distribution, should be chosen such that one gets the desired censoring rate (i.e., the relative frequency of censoring for all observations).

The realizations $t_i$ of the survival times are derived as

$$t_i = \min(T_i, C_i) \tag{6.8}$$

and the indicator for non-censoring equals

$$\delta_i = I(T_i \leq C_i). \tag{6.9}$$

One should mention that it is important that the hazard rate $\lambda(\cdot, \cdot)$ is specified in the right way: The exponent should not be chosen too large, as otherwise the integral of the cumulative hazard rate might be (numerically) diverging.

### Interpretation of Formulas for Survival Time Simulation

To get some insight into the sampling procedure, i.e., in the structure of the sampling scheme, that we just derived, we use the following simplified case of a classical Cox model.

**Theorem 6.1** *Let $\lambda(t, \boldsymbol{x}) = \lambda_0(t) \exp(\boldsymbol{x}'\boldsymbol{\beta})$, i.e., the case of a classical Cox model without time-varying effects and with a nonparametric baseline hazard rate $\lambda_0(t)$. Then, we can rewrite* (6.6) *as*

$$\log(\Lambda_0(T)) = \boldsymbol{x}'\widetilde{\boldsymbol{\beta}} + \underbrace{\log(-\log \tilde{U})}_{=:\varepsilon}, \tag{6.10}$$

*where $\widetilde{\boldsymbol{\beta}} = -\boldsymbol{\beta}$.*

*Proof.* In the Cox model, the hazard rate $\exp(\eta(t, \boldsymbol{x}))$ is replaced by $\lambda_0(t) \exp(\boldsymbol{x}'\boldsymbol{\beta})$. Thus, from (6.6) we get:

$$\begin{aligned}
& \int_0^T \lambda_0(t) \exp(\boldsymbol{x}'\boldsymbol{\beta}) dt = -\log \tilde{U} \\
\Leftrightarrow \quad & \exp(\boldsymbol{x}'\boldsymbol{\beta}) \int_0^T \lambda_0(t) dt = -\log \tilde{U} \\
\Leftrightarrow \quad & \exp(\boldsymbol{x}'\boldsymbol{\beta}) \Lambda_0(T) = -\log \tilde{U} \\
\Leftrightarrow \quad & \Lambda_0(T) = \exp(\boldsymbol{x}'\boldsymbol{\beta})^{-1} \cdot (-\log \tilde{U}) \\
\Leftrightarrow \quad & \log(\Lambda_0(T)) = -\boldsymbol{x}'\boldsymbol{\beta} + \log(-\log \tilde{U}) \\
\Leftrightarrow \quad & \log(\Lambda_0(T)) = \boldsymbol{x}'(-\boldsymbol{\beta}) + \log(-\log \tilde{U})
\end{aligned}$$

□

Thus, from Equation (6.10) we see that in this simple case we obtain a linear model for the transformed time $\log(\Lambda_0(T))$ with an (additive) error term $\varepsilon$ specified by the distribution of $\tilde{U} \sim U[0, 1]$. In other words this means, we have a parametric model with a parametric error term determined by the samples from $\tilde{U}$ for a (nonparametric) transformation of time, i.e. a special case of a transformation model.

### 6.1.2. Measures of prediction error

A classical measure to assess the quality of the prediction would be the mean squared prediction error. Therefore, the expected survival time given the covariates $\mathbb{E}(T|\boldsymbol{x})$ is needed. An estimate is defined by the model for the hazard rate:

$$\begin{aligned}
\hat{t} &:= \mathbb{E}_{\hat{\lambda}}(T|\boldsymbol{x}) = && (6.11)\\
&= \int_0^\infty t\hat{f}(t|\boldsymbol{x})dt = && (6.12)\\
&= \int_0^\infty \hat{S}(t|\boldsymbol{x})dt = && (6.13)\\
&= \int_0^\infty \exp(-\int_0^t \hat{\lambda}(u|\boldsymbol{x})du)dt. && (6.14)
\end{aligned}$$

Formula (6.13) holds, as $T|\boldsymbol{x}$ is a non-negative, real-valued random variable with $\mathbb{E}(T|\boldsymbol{x}) < \infty$. Using $\hat{t}$ we can, for example, calculate the estimated mean squared prediction error

$$\widehat{\text{MSE}} = \frac{1}{n}\sum_{i=1}^{n}(t_i - \hat{t}_i)^2. \tag{6.15}$$

The problem with this approach is that we need to integrate over the time from zero to infinity. As the model is just estimated on a time span from $t_{\min}$ to $t_{\max}$, where (classically) it holds that $t_{\max} \ll \infty$, we heavily have to extrapolate from the model. Thus, an arbitrarily large error can occur, depending on how well the extrapolated model holds. The problem of extrapolation even increases whenever we use splines to model smooth functions of time. For linear effects, we have a natural way of extrapolation by just moving on with the same slope. Splines instead have no natural way to do so. One could think of carrying forward the last value (i.e., to use a constant prediction for values of time larger than $t_{\max}$), linear extrapolation, or even carrying on the skewness that we have at $t_{\max}$ (e.g., if we have estimated a quadratic function, we extrapolate quadratically). However, no choice is altogether superior to another one.

To overcome these problems, we could use quantile based measures, i.e. compare, for example, the predicted median survival with the median survival from the true model. Another approach tries to estimate the expected Brier score (Brier 1950). A robust estimator with application to survival models (under the assumption that censoring is independent of the covariates) was introduced by Graf et al. (1999). Further refinements and consistent estimators in the case of censoring mechanisms that are conditionally independent from the survival times, given the covariates, were introduced by Gerds & Schumacher (2006).

The Brier score approach is based on the prediction of the event status

$$Y_i(t) = I_{(T_i>t)} = \begin{cases} 1 & \text{if } T_i > t \\ 0 & \text{if } T_i \leq t \end{cases} \tag{6.16}$$

at a fixed time $t$ for the $i$-th observation. Given the covariates $\boldsymbol{x}_i$, $Y_i(t)$ can be estimated by a risk prediction model based on the estimated survival model. The risk prediction model is the probability of surviving until $t$, i.e., the survivor function $S(t|\boldsymbol{x}_i)$ which can be estimated by

$$\hat{S}(t|\boldsymbol{x}) = \exp\left(-\int_0^t \hat{\lambda}(u|\boldsymbol{x})du\right). \tag{6.17}$$

This leads to the true prediction error, which is proportional to the expected Brier score:

$$\text{err}(t;\hat{S}) = \mathbb{E}[(Y(t) - \hat{S}(t|\boldsymbol{x}))^2]. \tag{6.18}$$

The prediction error can be estimated by

$$\widehat{\text{err}}(t;\hat{S}) = \frac{1}{n}\sum_{i=1}^{n}(Y_i(t) - \hat{S}(t|\boldsymbol{x}_i))^2 \cdot \omega(t), \tag{6.19}$$

where weights $\omega(t)$ are needed to incorporate censoring (see Gerds & Schumacher 2006). Binder & Schumacher (2008) used this measure for boosting models in the survival context and applied the integrated prediction error estimate as a measure that summarizes the prediction errors for multiple, fixed time points:

$$\widehat{\text{IErr}}(\tilde{t};\hat{S}) = \int_0^{\tilde{t}} \widehat{\text{err}}(u;\hat{S})du. \tag{6.20}$$

The upper boundary of the integral $\tilde{t}$ is selected in such a way that it is in the range of the given data. Classically, one would choose it from an area, where there is still enough data present to prevent the measure to depend on the possibly instable boundaries. Integration over a transformation of the survivor function as in (6.13) is necessary to compute (6.20) but we can restrict the integral to areas with enough observations and therefore, the problems described above are avoided. Incorporating this strategy to assess the predictive power of the $\text{Cox}_{\text{flex}}$Boost algorithm will be subject to future work.

### 6.1.3. Outline of Simulations

To gain deeper insights in the properties of the proposed $\text{Cox}_{\text{flex}}$Boost procedure, three simulation studies were performed. In all three settings, we generated data sets consisting of 300 observations in the learning sample and 100 observations in the validation sample. The former sample was used to fit the Cox-type

additive model (2.11) with $\text{Cox}_{\text{flex}}$Boost, and the latter to determine the stopping iteration $m_{\text{stop}}$. First, we want to give a short outline of the simulation goals. Afterwards, we present the exact simulation schemes and the results.

In the first setting, data was simulated without any time-varying effects. Even the baseline hazard was chosen constant over time. This corresponds to data from an exponential distribution (given the covariate values $\boldsymbol{x}$). In this setting, the goal was to evaluate the performance of the algorithm with respect to the detection of linear and smooth effects applying the model choice scheme as proposed in Section 4.3.4. Another interesting topic was the investigation of the ability to perform variable selection, both in the model choice scheme and in a variable selection scheme. The latter applies a model that only uses one type of base-learner per covariate (linear base-learners for categorical covariates and P-spline base-learners with 4 degrees of freedom otherwise).

The second and third setting included time-varying effects. In the second setting only the baseline hazard depends on time, whereas in the third setting a categorical covariate has an additional time-varying effect. This corresponds to different baseline hazards in the two groups. In the second setting, we tried to investigate, whether the algorithm chooses time-varying effects even if there are none present. Furthermore, we investigated again the properties of variable selection and wanted to check, if the effects are estimated "correctly".

In the third setting, only the model choice procedure (see Sec. 4.3.4) was used. We investigated whether the time-varying effect was detected correctly and whether it was estimated right. Moreover, it is again of interest, if other effects, like linear and smooth effects, are detected and modeled "correctly".

For the first two simulation schemes we investigated the strength of variable selection with component-wise boosting. Model choice as given in Section 4.3.4 is applied in all three simulation settings. Obviously, model choice includes variable selection, too. The difference is that for variable selection we only considered linear and P-spline base-learners, the latter with 4 degrees of freedom, and for model choice, we centered the functions and used 1 degree of freedom. In the second and third simulation study we also added time-varying base-learners in the model choice framework (see Tables 6.1 to 6.3).

In all three schemes, we recorded the estimated degrees of freedom within each iteration and for each repetition to check their deviation from the initially defined degrees of freedom $\widetilde{\text{df}}_j$ and to see how they change over the iterations (cf. Sec. 5.3.2).

### Details on Simulation Scheme 1

As already stated, we have three different simulation schemes. For the first study we simulated 400 realizations of 15 i.i.d. covariates $X_1, \ldots, X_{15}$ according to

$$\begin{array}{lll} X_1, X_2, & X_7, X_8, X_9 & \overset{i.i.d.}{\sim} U[-1, 1] \\ X_3, X_4, & X_{10}, X_{11}, X_{12} & \overset{i.i.d.}{\sim} N(0, 1) \\ \underbrace{X_5, X_6,}_{\text{effective covariates}} & \underbrace{X_{13}, X_{14}, X_{15}}_{\text{non-effective covariates}} & \overset{i.i.d.}{\sim} B(1, 0.5). \end{array} \tag{6.21}$$

The covariate realizations $\boldsymbol{x}_i = (x_{1,i}, \ldots, x_{15,i}), i = 1, \ldots, 400$, were used to simulate survival times with the hazard rate

$$\begin{aligned} \lambda(t, \boldsymbol{x}) = \exp\Big(2 + \sin(-x_1^2 - 0.6x_1^3) + 1.4x_2^2 \\ + 0.5\sin(1.5x_3) + x_4 - 2x_5 + 0.1x_6\Big) \end{aligned} \tag{6.22}$$

using the inversion method (6.7). Only the covariates $X_1$ to $X_6$ have an effect on the survival time. We call these covariates "effective covariates". Covariates $X_7$ to $X_{15}$ have no effect on the sampled times. Therefore, we use the term "non-effective covariates" for these variables. We have two uniformly distributed, two standard normally distributed and two binary distributed covariates in the model. $X_1$ to $X_3$ have non-linear effects, $X_4$ and the categorical variables $X_5$ and $X_6$ have linear effects. The effects are depicted in Section 6.1.5 (see e.g., Fig. 6.2). The first 300 observations are used to fit the model, i.e., they are used as learning sample. The remaining 100 observations serve as validation sample to determine the stopping iteration. The censoring times $C_i$ are simulated i.i.d. exponentially distributed with rate $\lambda = 1/\bar{t}$, i.e., with $\mathbb{E}(C) = \frac{1}{n}\sum_{i=1}^{n} t_i = \bar{t}$, leading to a non-censoring rate of approximatively 70%.

Table 6.1 gives an overview of the covariates and the way they enter the model. In the first setting, this can be either as linear base-learner or as P-spline base-learner. In the table we denote the base-learners with the names of the `R`-functions in $\text{Cox}_{\text{flex}}$Boost. The function `bols()` creates a linear base-learner and `bbs()` represents a P-spline base-learner. `bolsTime()` and `bbsTime()` are functions for linear and P-spline base-learners of time. Both base-learners of time represent time-varying effects expressed as varying coefficient terms as in Section 2.3.2, i.e., if a covariate (other than time) is associated with these base-learners, it is modeled as an interaction of time (as linear or smooth term) with the respective covariate. For more computational details see Appendix B.2. Furthermore, the degrees of freedom are given in brackets for flexible base-learners. Note that in the model choice schemes we set the degrees of freedom df = 1 and center the function as described in Section 4.3.4.

For both settings, variable selection and model choice (which obviously also includes variable selection), we simulated 200 randomly drawn replicates of the

data set. Each data set was sampled with the hazard rate (6.22). The models were fitted using the option `hardStop = FALSE` in `cfboost()`. This should prevent the boosting procedure to stop too early. Therefore, at the end of the boosting procedure it is checked if the iteration with minimal risk is close to the pre-specified number of boosting iterations $m_{\text{stop}}$. If this is true, $m_{\text{stop}}$ is increased, i.e., the procedure continues until the minimal risk is further away from the boundary or until the number of boosting iterations is increased more than four times. Hence, with an initial $m_{\text{stop}} = 400$ and an increase of 100 every time we did not reach the minimal risk early enough before the boundary, we maximally used 800 boosting iterations.

| **Variable Selection** | | | | | |
|---|---|---|---|---|---|
| | Type | `bols` | `bbs` (df = 4) | `bolsTime` | `bbsTime` (df = 4) |
| $x_1 - x_4$ | continuous | | ✓ | | |
| $x_5 - x_6$ | categorical | ✓ | | | |
| $x_7 - x_{12}$ | continuous | | ✓ | | |
| $x_{13} - x_{15}$ | categorical | ✓ | | | |
| **Model Choice** | | | | | |
| | Type | `bols` | `bbs` (df = 1) | `bolsTime` | `bbsTime` (df = 1) |
| $x_1 - x_4$ | continuous | ✓ | ✓ | | |
| $x_5 - x_6$ | categorical | ✓ | | | |
| $x_7 - x_{12}$ | continuous | ✓ | ✓ | | |
| $x_{13} - x_{15}$ | categorical | ✓ | | | |

Table 6.1.: **Simulation Scheme 1:** Overview of combinations of covariates and base-learners in the variable selection and model choice scheme. Combinations with ✓were used in the model formula.

### Details on Simulation Scheme 2

The covariates in the second simulation scheme were sampled identically to the first simulation study (6.21). Again, 400 realizations of the 15 covariates were randomly drawn per replicate. The survival time was then sampled with the hazard rate

$$\lambda(t, \boldsymbol{x}) = \exp\Big( -1 + \log(t) + \sin(-x_1^2 - 0.6x_1^3) + 1.4x_2^2 + 0.5\sin(1.5x_3) + x_4 - 2x_5 + 0.1x_6 \Big). \tag{6.23}$$

Thus, we see that only the intercept term was changed from 2 to $-1$ and a baseline hazard $\log(t)$ was introduced. As above, we used the first 300 observations as learning sample and the rest was used as validation sample. For variable selection we included all 15 variables as possible base-learners. Categorical covariates were added as linear base-learners, the remainder as P-spline

base-learners with four degrees of freedom. In the model choice scheme we additionally included linear and P-spline base-learners for time-varying effects for all covariates and decomposed the smooth terms of continuous covariates as described in Section 4.3.4. Especially the estimation of time-varying effects is very time-consuming. Hence, we used only the six effective covariates in the model formulation to decrease the computational effort. Non-effective covariates were excluded a priory from the model. Still, we have 24 base-learners (four base-learners for each of the four continuous covariates, three base-learners for the two categorical covariates and two base-learners for the (log) baseline hazard) where 14 are time-varying, which all have to be fitted in each iteration. As a comparison, in the variable selection scheme we only have 16 base-learners (one per covariate), where only one (for the baseline hazard) is time-varying. An overview of the model terms we used for variable selection and model choice is given in Table 6.2.

In the second simulation scheme we used only 50 replicates for the variable selection scheme and 50 replicates for model choice to examine the properties, as the time until convergence of one boosting model was very high. Boosting one model from the model choice scheme (with that many time varying base-learners) took about one day on a Dual-Core AMD Opteron™ server with 2.8 GHz. For both settings we chose the censoring times $C_i$ again to be i.i.d. exponentially distributed with rate $\lambda = \left(\frac{1}{n}\sum_{i=1}^{n} t_i\right)^{-1} = 1/\bar{t}$ leading to a non-censoring rate of approximatively 50%.

| **Variable Selection** | | | | | |
|---|---|---|---|---|---|
| | Type | `bols` | `bbs` (df = 4) | `bolsTime` | `bbsTime` (df = 4) |
| $t$ | time | | | ✓ | |
| $x_1 - x_4$ | continuous | | ✓ | | |
| $x_5 - x_6$ | categorical | ✓ | | | |
| $x_7 - x_{12}$ | continuous | | ✓ | | |
| $x_{13} - x_{15}$ | categorical | ✓ | | | |
| **Model Choice** | | | | | |
| | Type | `bols` | `bbs` (df = 1) | `bolsTime` | `bbsTime` (df = 1) |
| $t$ | time | | ✓ | ✓ | |
| $x_1 - x_4$ | continuous | ✓ | ✓ | ✓ | ✓ |
| $x_5 - x_6$ | categorical | ✓ | | ✓ | ✓ |
| $x_7 - x_{15}$ | not-included | | | | |

Table 6.2.: **Simulation Scheme 2:** Overview of combinations of covariates and base-learners in the variable selection and model choice scheme. Combinations with ✓were used in the model formula.

### Details on Simulation Scheme 3

In the third simulation scheme with a baseline hazard that is depending on time and one time-varying effect, we only performed model choice. Pure variable selection as in the settings above was omitted. To reduce the computational burden, we decided to include only effective covariates as already done in the model choice procedure in simulation 2. Thus, we sampled the six covariates $X_1$ to $X_6$ according to

$$\begin{aligned} X_1, X_2, &\overset{i.i.d.}{\sim} U[-1,1] \\ X_3, X_4, &\overset{i.i.d.}{\sim} N(0,1) \\ \underbrace{X_5, X_6,}_{\text{effective covariates}} &\overset{i.i.d.}{\sim} B(1,0.5). \end{aligned} \tag{6.24}$$

Applying the inversion method we used 400 realizations $\boldsymbol{x}_i = (x_{1,i}, \ldots, x_{6,i}), i = 1, \ldots, 400$ to sample survival times with the hazard rate

$$\begin{aligned} \lambda(t, \boldsymbol{x}) = \exp\Big(&2 + \log(t + 0.2) + \sin(-x_1^2 - 0.6x_1^3) - 0.3x_2 \\ &+ 0.5\sin(1.5x_3) + x_4 - 2x_5 + 2\sqrt{t} \cdot x_6\Big). \end{aligned} \tag{6.25}$$

Like in both previous simulation schemes we simulated the censoring times $C_i \overset{i.i.d.}{\sim} \text{Expo}(1/\bar{t})$. In this case this corresponds to a non-censoring rate of approximatively 50%. The base-learners that were used in the model are given in Table 6.3 and correspond to those from simulation 2 (model choice).

| Model Choice | | | | | |
|---|---|---|---|---|---|
| | Type | `bols` | `bbs` (df = 1) | `bolsTime` | `bbsTime` (df = 1) |
| $t$ | time | | ✓ | ✓ | |
| $x_1 - x_4$ | continuous | ✓ | ✓ | ✓ | ✓ |
| $x_5 - x_6$ | categorical | ✓ | | ✓ | ✓ |

Table 6.3.: **Simulation Scheme 3:** Overview of combinations of covariates and base-learners in the model choice scheme. Combinations with ✓were used in the model formula.

Again, we used the option `hardStop = FALSE` to reduce the run-time where possible and to increase the number of boosting iterations when necessary. The number of simulation replicates was 50. Again, it took about one day to estimate one model replicate as in the model choice procedure in simulation 2.

### 6.1.4. Simulation Results 1: Model Choice and Variable Selection

In the following section, we explore the accuracy of variable selection given by the relative frequency of (correctly) selected base-learners. This means we count the models (simulation replicates) where the base-learner was included and ignore how often and in which boosting iteration(s) the base-learner was selected. In the same manner we examine the accuracy of model choice. One needs to mention that the pure fact of inclusion in the model or even the frequency of the selection of the base-learner of interest in *one* boosting model are a relatively crude measure. For simplicity, think of a linear base-learner that is repeatedly selected but has a very little influence in terms of the estimated coefficients. Thus, we see that one should also take the magnitude of the selected base-learner into account. For linear base-learners (and standardized covariates) the estimated slope seems to be a good measure of the variable importance. For flexible functions (e.g., P-splines) we cannot use the slope to measure variable importance as it does not exist here. One could think of the integral over the standardized $L_2$-norm of the function

$$\frac{1}{x_{\max} - x_{\min}} \int_{x_{\min}}^{x_{\max}} \| f_j^{[m_{\text{stop}}]}(x) \|_2 \, dx \tag{6.26}$$

with covariates $x \in [x_{\min}, x_{\max}]$ as a measure for variable importance. Note that $x$ could also be the time $t$. The problem of this measure is that it highly depends on the quality of the estimated function. Especially at the boundaries of the support we often have only few observations with possibly very inaccurate predictions (cf. Sec. 6.1.5). These few observations might then govern the whole measure. Thus, the measure is not only dependent on the true variable importance but also on "boundary problems". A possible solution could be to restrict the integral to an interval $\mathcal{I} \subset [x_{\min}, x_{\max}]$, where $\mathcal{I}$ represents a region where we have got enough observations for stable function estimates. This could, for example, be based on percentiles, i.e., $\mathcal{I} = [x_{0.05}, x_{0.95}]$, where $x_\alpha$ is the $(\alpha \cdot 100)\%$ percentile.

**Simulation Scheme 1** As long as we have no robust measure for the variable importance at hand we stick to the crude measure: The frequency of selected base-learners as introduced above. Table 6.4 shows the selection frequencies of the base-learners. Only the variables $x_1$ to $x_6$ have an effect on the hazard rate (6.22) and thus, on the survival time. These effective covariates are presented in the upper half of the table. We see that all effective base-learners have a selection frequency close to one or exactly 1 except for the base-learner of $x_6$. The low selection frequency of `bols(`$x_6$`)` has two reasons: First, the base-learner has only 2 degrees of freedom (one for the intercept and one for the slope) whereas the P-spline base-learners have 4 degrees of freedom. Second and more important,

the effect of $x_6$ is very small. It is 20 times smaller than the effect of the other categorical covariate $x_5$ (see (6.22)). Hence, the low selection frequency seems very plausible. With the lower number of df, compared to the P-spline base-learners, it is more surprising that `bols(`$x_5$`)` was selected in all 200 models (rel. freq. = 100%).

| **Effective Covariates** | |
|---|---|
| | rel. freq. of selection |
| bbs($x_1$) | 0.92 |
| bbs($x_2$) | 0.98 |
| bbs($x_3$) | 0.94 |
| bbs($x_4$) | 1.00 |
| bols($x_5$) | 1.00 |
| bols($x_6$) | 0.15 |
| **Non-Effective Covariates** | |
| | rel. freq. of selection |
| bbs($x_7$) | 0.35 |
| bbs($x_8$) | 0.36 |
| bbs($x_9$) | 0.34 |
| bbs($x_{10}$) | 0.40 |
| bbs($x_{11}$) | 0.36 |
| bbs($x_{12}$) | 0.34 |
| bols($x_{13}$) | 0.08 |
| bols($x_{14}$) | 0.07 |
| bols($x_{15}$) | 0.08 |

Table 6.4.: **Variable Selection: Simulation Scheme 1** – Relative frequencies of the selection of the base-learners (in 200 replicates). The upper half shows the base-learners for covariates that have an influence on the hazard rate, the lower half those without influence.

In the lower part of Table 6.4 we expect the selection frequency to be close to zero or at least substantially smaller than for the effective covariates. All selection frequencies are around 30% for smooth base-learners (`bbs()`) and close to zero for linear base-learners (`bols()`). This is less than half the frequency of effective covariates if one compares linear with linear base-learners and smooth with smooth base-learners. The mean number of selected base-learners was 7.395. These included on average 4.995 effective base-learners with only very little variance (where the most of it is accounted to models that chose all six effective base-learners). Only 2.4 non-effective variables were selected on average with a quite large variance. This shows, that (on average) we only use half the variables with a very high fraction of effective variables. Altogether, this reveals that we have a reasonable variable selection scheme which is clearly not an "oracle" (i.e., far away from being perfect).

Applying the model choice scheme in the same data situation yields very similar results. In Table 6.5 we see very high frequencies of selection. The base-learner `bols(`$x_2$`)` with only 29% selections is an exception. If we look at the true influence of $x_2$ (see (6.22)) it shows that this is a good result, as we have a quadratic influence of this covariate. For the other effects, we can see some linear trend. The sine form of $x_3$ has some trend when we realize that we have only very few observations at the boundary (cf. Fig. 6.1) and thus, the slope in the middle of the variable's span dominates estimation. Another positive result is that the selection frequency of the linear effect for $x_6$ is strongly increased. This can be attributed to the decomposition of the base-learners with reduced degrees of freedom for each single base-learner. For $x_4$ (which has in reality a linear effect) the algorithm selected in 23% of the replicates a flexible deviation from linearity. Thus, in some models the (wrong) impression of an underlying non-linear effect of $x_4$ is given. However, compared to the selection frequencies of the other effects, this is only of minor importance. In addition, in Section 6.1.5 we will see that the departures from linearity are only very small. When we look at the non-effective covariates we see that again, the frequencies of selection are much smaller than those of the effective covariates. Again, the categorical covariates gained more importance (w.r.t. the variable selection scheme) but they are still less frequently selected than the base-learners `bols(`$x_5$`)` and `bols(`$x_6$`)` of the effective covariates.

Note that the number of base-learners is now not equal to the number of variables. A variable is selected if *any* of the base-learners of this variable is selected. On average we selected 9.97 variables with 5.465 effective variables and 4.505 non-effective variables. Thus, we realize that in the model choice scheme we tend to select more variables and to select more non-effective variables. Perhaps, this is due to an increased number of possible base-learners. This argument is backed by the finding that we selected 13 out of 25 base-learners. This is again roughly half the total number of base-learners as in the variable selection scheme. However, we selected (on average) about 5.5 non-effective *base-learners* which corresponded on average to 4.5 non-effective *variables*. Thus, almost every non-effective base-learner is based on another variable. Thus, the variable selection scheme tends to lead to sparser models.

| **Effective Covariates** | |
|---|---|
| | rel. freq. of selection |
| bols($x_1$) | 0.80 |
| bbs($x_1$) | 0.97 |
| bols($x_2$) | 0.29 |
| bbs($x_2$) | 1.00 |
| bols($x_3$) | 0.88 |
| bbs($x_3$) | 0.90 |
| bols($x_4$) | 1.00 |
| **bbs($x_4$)** | **0.23** |
| bols($x_5$) | 1.00 |
| bols($x_6$) | 0.48 |
| **Non-Effective Covariates** | |
| | rel. freq. of selection |
| bols($x_7$) | 0.36 |
| bbs($x_7$) | 0.52 |
| bols($x_8$) | 0.36 |
| bbs($x_8$) | 0.46 |
| bols($x_9$) | 0.38 |
| bbs($x_9$) | 0.50 |
| bols($x_{10}$) | 0.34 |
| bbs($x_{10}$) | 0.28 |
| bols($x_{11}$) | 0.32 |
| bbs($x_{11}$) | 0.29 |
| bols($x_{12}$) | 0.37 |
| bbs($x_{12}$) | 0.24 |
| bols($x_{13}$) | 0.41 |
| bols($x_{14}$) | 0.36 |
| bols($x_{15}$) | 0.32 |

Table 6.5.: **Model Choice: Simulation Scheme 1** – Relative frequencies of the selection of the base-learners (in 200 replicates). The upper half shows the base-learners for covariates that have an influence on the hazard rate, the lower half those without influence. Wrongly assigned smooth effects are printed in bold face.

**Simulation Scheme 2** The results from the variable selection scheme (i.e., component-wise boosting) applied on the data sampled with the hazard rate (6.23) are given in table 6.6. Compared to the first simulation scheme the setting was only altered by adding a time-varying baseline hazard rate. The selection frequencies are almost the same as in the first setting (see Tab. 6.4). We have again very high selection frequencies for the effective covariates with the exception of `bols(`$x_6$`)`, which we tried to explain above. Note that the baseline hazard, i.e., `bbsTime(`$t$`)` was chosen in every simulation replicate. The results for the non-effective covariates are also the same: We have only half the selection frequency than for effective covariates but on average still about 2.5 wrongly selected variables in the resulting models.

| **Effective Covariates** | |
|---|---|
| | rel. freq. of selection |
| bbsTime($t$) | 1.00 |
| bbs($x_1$) | 0.82 |
| bbs($x_2$) | 0.96 |
| bbs($x_3$) | 0.92 |
| bbs($x_4$) | 1.00 |
| bols($x_5$) | 1.00 |
| bols($x_6$) | 0.16 |
| **Non-Effective Covariates** | |
| | rel. freq. of selection |
| bbs($x_7$) | 0.40 |
| bbs($x_8$) | 0.38 |
| bbs($x_9$) | 0.34 |
| bbs($x_{10}$) | 0.44 |
| bbs($x_{11}$) | 0.40 |
| bbs($x_{12}$) | 0.46 |
| bols($x_{13}$) | 0.12 |
| bols($x_{14}$) | 0.14 |
| bols($x_{15}$) | 0.08 |

Table 6.6.: **Variable Selection: Simulation Scheme 2** – Relative frequencies of the selection of the base-learners (in 50 replicates). The upper half shows the base-learners for covariates that have an influence on the hazard rate, the lower half those without influence.

Table 6.7 shows the selection frequencies we got in the model choice scheme. We see that all selection frequencies for time-varying terms are very high. Most of them are much higher than the frequencies of non-effective covariates for variable selection. Sometimes, we even have got selection frequencies close to one. Here we see a strong bias in favor to time-varying effects. These effects

are included in the models as time-varying effects deliver some of the flexibility the model terms require. This problem is already discussed in Section 2.2.2. However, in most cases the true effects are also selected to enter the model and the selection frequency of the true effects is typically (slightly) higher than or at least comparable to the selection frequency of the time-varying effects. The flexible base-learner for time $t$ is always chosen to enter the model.

| **Effective Covariates** | |
|---|---|
| | rel. freq. of selection |
| bolsTime($t$) | 0.58 |
| bbsTime($t$) | 1.00 |
| bols($x_1$) | 0.40 |
| bbs($x_1$) | 0.90 |
| **bolsTime($t$, $x_1$)** | **0.68** |
| **bbsTime($t$, $x_1$)** | **0.40** |
| bols($x_2$) | 0.06 |
| bbs($x_2$) | 0.98 |
| **bolsTime($t$, $x_2$)** | **0.50** |
| **bbsTime($t$, $x_2$)** | **0.34** |
| bols($x_3$) | 0.52 |
| bbs($x_3$) | 0.76 |
| **bolsTime($t$, $x_3$)** | **0.70** |
| **bbsTime($t$, $x_3$)** | **0.30** |
| bols($x_4$) | 0.98 |
| **bbs($x_4$)** | **0.18** |
| **bolsTime($t$, $x_4$)** | **1.00** |
| **bbsTime($t$, $x_4$)** | **0.50** |
| bols($x_5$) | 1.00 |
| **bolsTime($t$, $x_5$)** | **0.84** |
| **bbsTime($t$, $x_5$)** | **0.56** |
| bols($x_6$) | 0.32 |
| **bolsTime($t$, $x_6$)** | **0.34** |
| **bbsTime($t$, $x_6$)** | **0.42** |

Table 6.7.: **Model Choice: Simulation Scheme 2** – Relative frequencies of the selection of the base-learners (in 50 replicates). Wrongly assigned smooth or time-varying effects are printed in bold face.

**Simulation Scheme 3** The third simulation with a time-dependent baseline hazard rate and one time-varying effect shows the same problems regarding the selection bias. The time-varying effect of $x_6$ is always discovered and the (log) baseline hazard is almost always selected. Although the time-varying effect of $x_6$ is in truth non-linear, only in roughly half of the models a flexible time-varying

effect is chosen.

Another problem that arises in this context is that we hardly can interpret the resulting effects as we almost always have a mixture of different modeling alternatives for the covariates in the model: $x_1$, for example, is selected as smooth effect (linear and centered smooth effect) *and* as (flexible) time-varying effect at the same time. Thus, the models are not really interpretable and only useful in the context of pure prediction.

| **Effective Covariates** | |
|---|---|
| | rel. freq. of selection |
| bolsTime($t$) | 0.52 |
| bbsTime($t$) | 0.92 |
| bols($x_1$) | 0.38 |
| bbs($x_1$) | 1.00 |
| **bolsTime($t$, $x_1$)** | **0.80** |
| **bbsTime($t$, $x_1$)** | **0.40** |
| bols($x_2$) | 0.34 |
| **bbs($x_2$)** | **0.60** |
| **bolsTime($t$, $x_2$)** | **0.94** |
| **bbsTime($t$, $x_2$)** | **0.26** |
| bols($x_3$) | 0.32 |
| bbs($x_3$) | 0.90 |
| **bolsTime($t$, $x_3$)** | **0.84** |
| **bbsTime($t$, $x_3$)** | **0.32** |
| bols($x_4$) | 0.98 |
| **bbs($x_4$)** | **0.24** |
| **bolsTime($t$, $x_4$)** | **1.00** |
| **bbsTime($t$, $x_4$)** | **0.28** |
| bols($x_5$) | 1.00 |
| **bolsTime($t$, $x_5$)** | **0.80** |
| **bbsTime($t$, $x_5$)** | **0.66** |
| **bols($x_6$)** | **1.00** |
| bolsTime($t$, $x_6$) | 1.00 |
| bbsTime($t$, $x_6$) | 0.44 |

Table 6.8.: **Model Choice: Simulation Scheme 3** – Relative frequencies of the selection of the base-learners (in 50 replicates). Wrongly assigned linear, smooth or time-varying effects are printed in bold face.

### 6.1.5. Simulation Results 2: Estimated Effects

In the following section we only used the first 20 results of each simulation scheme and each modeling approach (variable selection / model choice). The reason for this is that the we need to store the complete list of results returned by `cfboost()` to plot the estimated fits that we will show below. These lists are really large. Storing one such list takes in our case up to 80 MB. For 20 results that makes already about 1.5 GB of disk space.

**Simulation Scheme 1** This time we start with the model choice scheme. In Figure 6.2 we see the estimated effects for the six effective covariates. Note that all function estimates and all true effects are centered such that their mean is equal to zero. This is required, as the "level" of the estimates can be altered: In every base-learner we have parameters for the intercept to allow the overall estimate to reach the right level. Hence, the intercept estimate of a base-learner is not connected with the effect of the corresponding covariate and thus the "level" of a base-learner is generally arbitrary. Actually we only want to compare the form of the estimated effects. For $x_3$ and $x_4$ we did not use the mean of the function on the complete support but only calculated the mean function on the interval $[-1, 1]$. This leads to a better comparable plot of the estimates. The reasons for this will be given in the following paragraph.

Figure 6.2 shows that the effects of $x_1$, $x_3$, $x_4$ and $x_5$ are estimated reasonable well. The estimated effects of $x_6$ seem to have a large bias but if we take the scale into account we see that there is no big deviation. The sine form of $x_3$ is estimated quite sensible in the middle part. Almost all estimations detect the slope from $-1$ to 1 about right. The boundaries instead, are estimated poorly. This is not due to a problem of boosting or $\text{Cox}_{\text{flex}}$Boost in particular but caused by the sparse data at the boundaries as depicted in Figure 6.1. Estimation of the function is only based on a dozen observations at the boundaries. When we look at the simulation scheme (6.21) we spot that $x_3$ is standard normally distributed and thus, we have light tails. This problem could be circumvented in the simulation study by choosing, for example, $x_3$ to be uniformly distributed on $[-3, 3]$. However, as normally distributed data or data with similar distributions is very common in reality we stick to this simulation scheme. For linear effects, the sparse tails do not pose such a big problem as we see from $x_4$. Only in some cases (23%, see Tab. 6.5) we have deviations from linearity. The estimation in the center region is hardly effected and only slight deviations in the areas with less observations can be identified. Hence, linear effects seem to be hardly effected by sparse tails.

The plots of the estimated effects for the non-effective covariates $x_7$ to $x_{15}$ can be found in Figure C.1 (see App. C). Altogether, the effects are more or less oscillating around zero (if present). Again the normally distributed variables

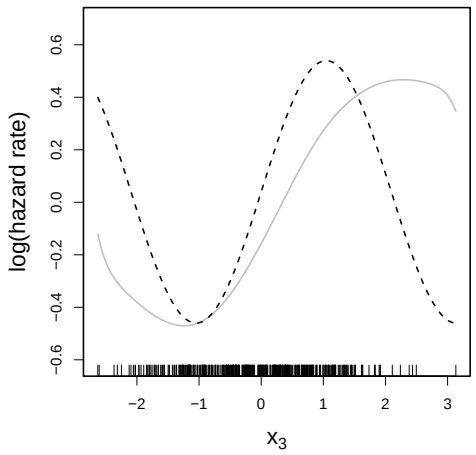


Figure 6.1.: **Model Choice: Simulation Scheme 1** – Estimation of the effect of $x_3$ for one model (gray line) and real effect (dashed line) together with rugs for the observed data. Effect estimates and the real effect are centered.

show a higher variation at the boundaries. Categorical variables, which are seldom selected (see Tab. 6.5), show the smallest deviations from the horizontal line (which corresponds to "no effect").

The figures for the variable selection scheme (see Fig. 6.3) show almost the same results as those of the model selection approach but generally tend to be more instable. For example, we have an almost straight line for the quadratic effect of $x_2$. Such deviations did not occur in the model choice setting. The deviations from a linear fit for $x_4$ are also larger than above. Note that the ideal linear fits of $x_5$ and $x_6$ are due to our model. Both variables are categorical. Thus, we included them only with a linear base-learner. The plots for the non-effective covariates are presented in Figure C.2 (see App. C).

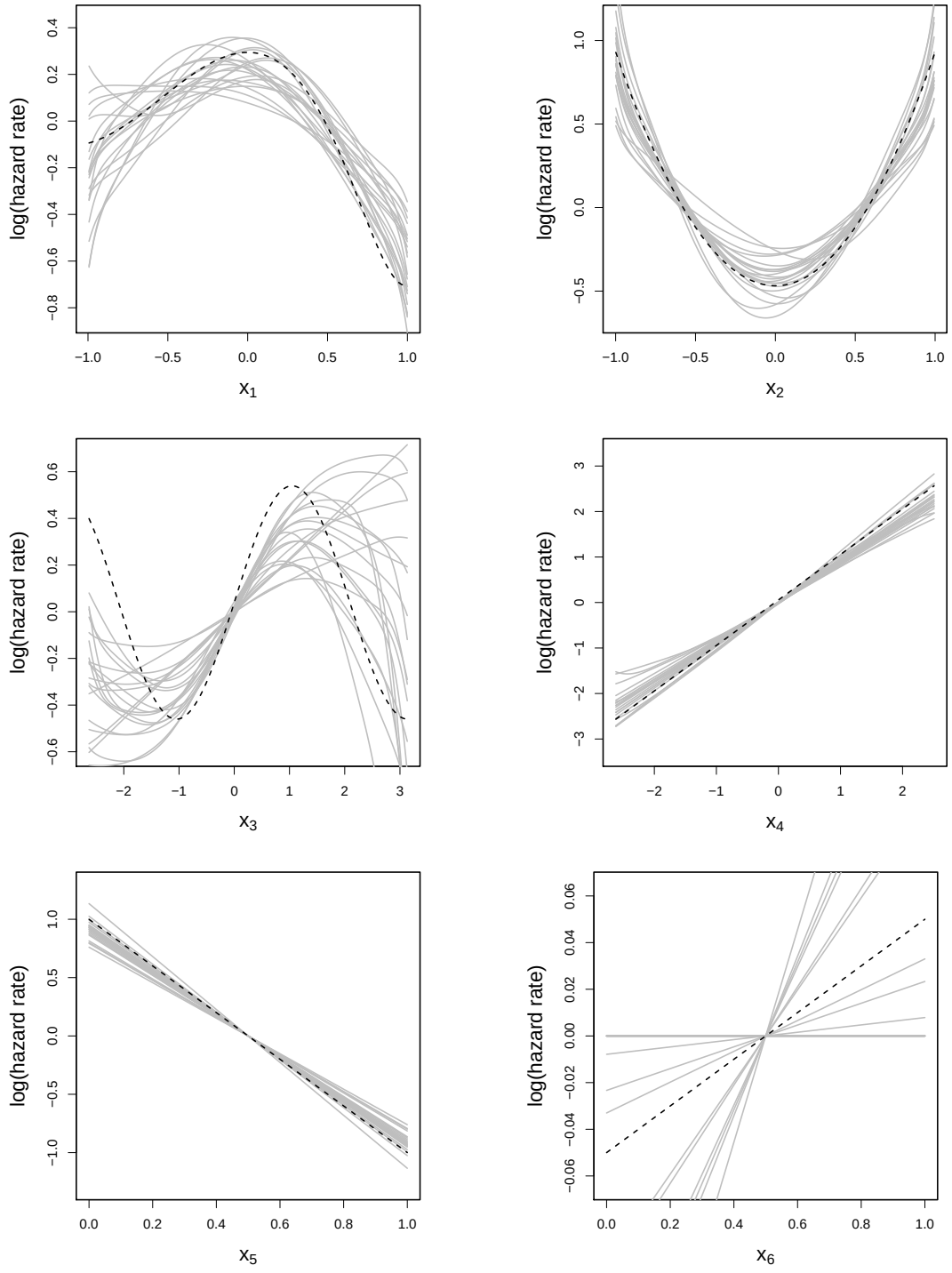


Figure 6.2.: **Model Choice: Simulation Scheme 1** – Estimation of covariate effects from 20 models (gray lines) and real effects (dashed lines). Effect estimates and real effects are centered. For $x_3$ and $x_4$ centering is only based on the "stable" areas $\in [-1, 1]$.

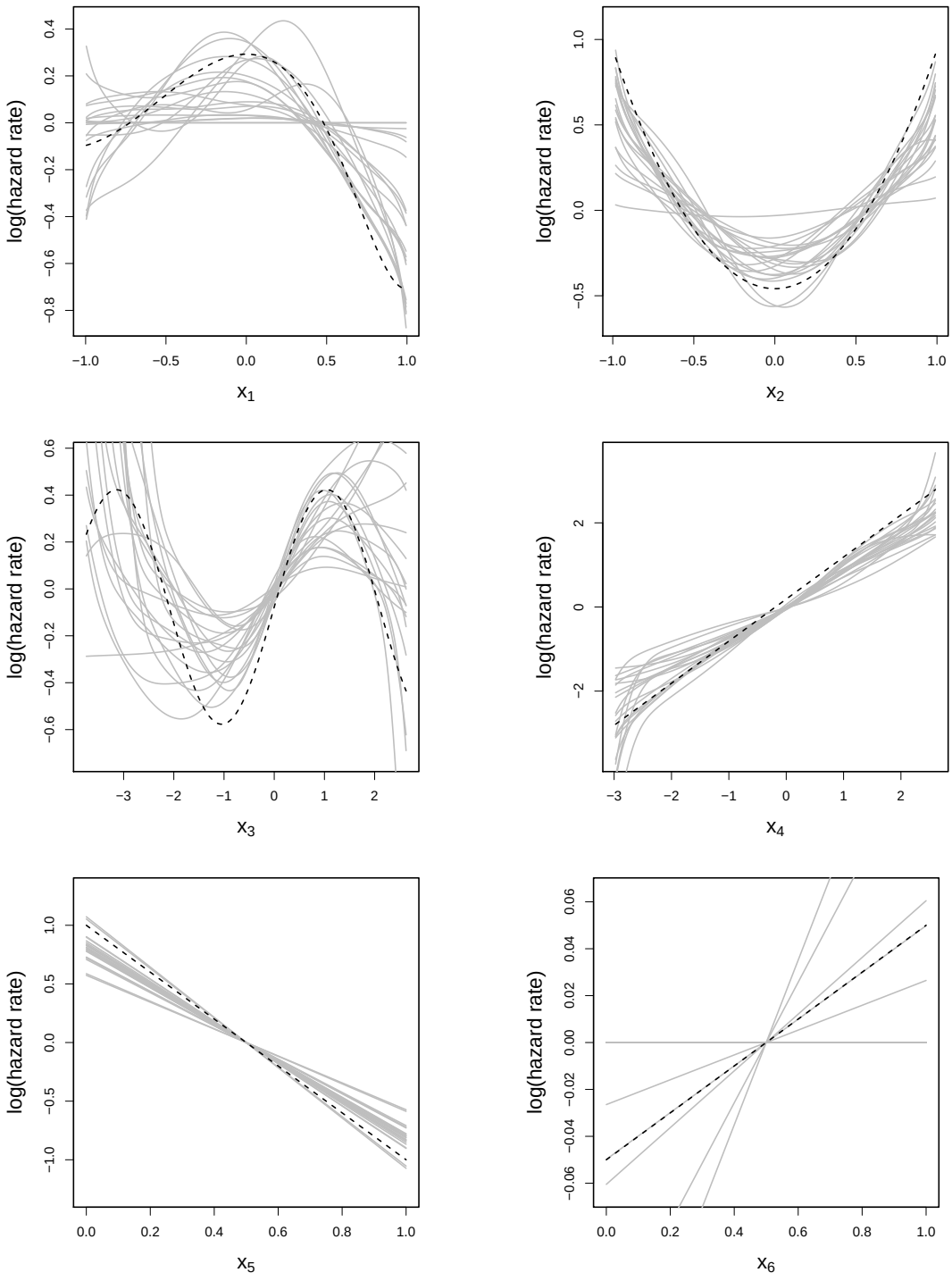


Figure 6.3.: **Variable Selection: Simulation Scheme 1** – Estimation of covariate effects from 20 models (gray lines) and real effects (dashed lines). Effect estimates and real effects are centered. For $x_3$ and $x_4$ centering is only based on the "stable" areas $\in [-1, 1]$.

**Simulation Scheme 2** In the second simulation scheme, a time-dependent baseline hazard is added. In the following section we concentrate on the model choice scheme. In the left panel of Figure 6.4 the estimated effects for time are depicted for 20 models. Until time $t \approx 6$ the curvature of the true effect is fairly well estimated. Thereafter, the quality of the estimation rapidly decreases. This is again due to the sparseness of the data as discussed above for normally distributed data. As it can be seen from the right graphic in Figure 6.4 the survival time has a sparse right tail which leads to instable estimations as already pointed out by Gray (1992). One has to mention that the "wrong offset" of the estimated functions is not caused by a bad prediction for early times (as the graph 6.4 might suggest) but it is caused by the abrupt decrease of the function estimates for later points in time. The underestimation of the function causes a smaller overall mean. Standardizing the estimate with this decreased mean, the level of the true effect in the graphic cannot be reached.

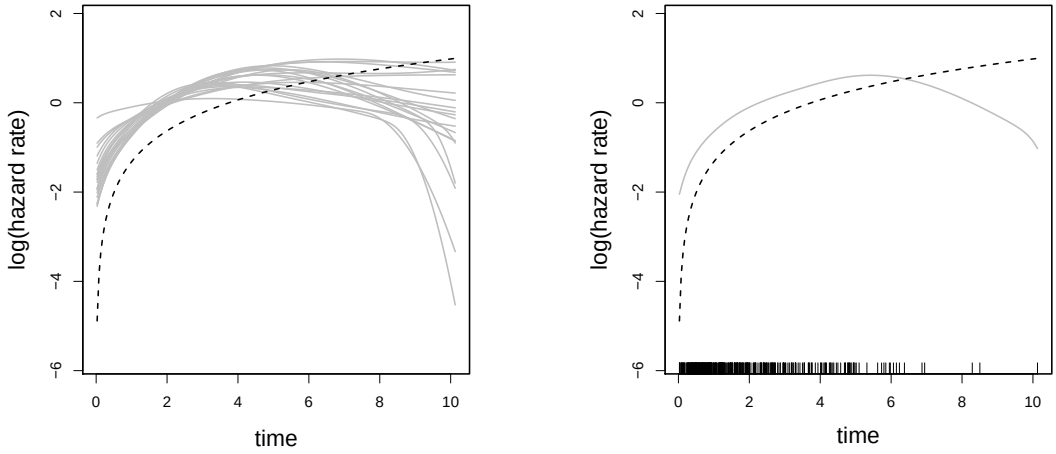


Figure 6.4.: **Model Choice: Simulation Scheme 2** – Left: Estimation of the baseline hazard from 20 models (gray lines) and real effect (dashed line). Right: Estimation of the baseline hazard for one model (gray line) and real effect (dashed line) together with rugs for the observed data. Effect estimates and real effects are centered.

From Figure 6.5 one can see that the estimations are overall fairly good. Again, we see problems for $x_3$ and $x_4$, which are due to sparse tails. In comparison, the effect of $x_6$ is so small that the errors are negligible. Note that we did not use non-effective covariates in the model choice scheme of the second simulation. Estimated effects from the variable selection scheme are given in Appendix C.1. The same consideration as in the model choice setting apply but the results are generally more instable as we have already seen in the first simulation scheme.

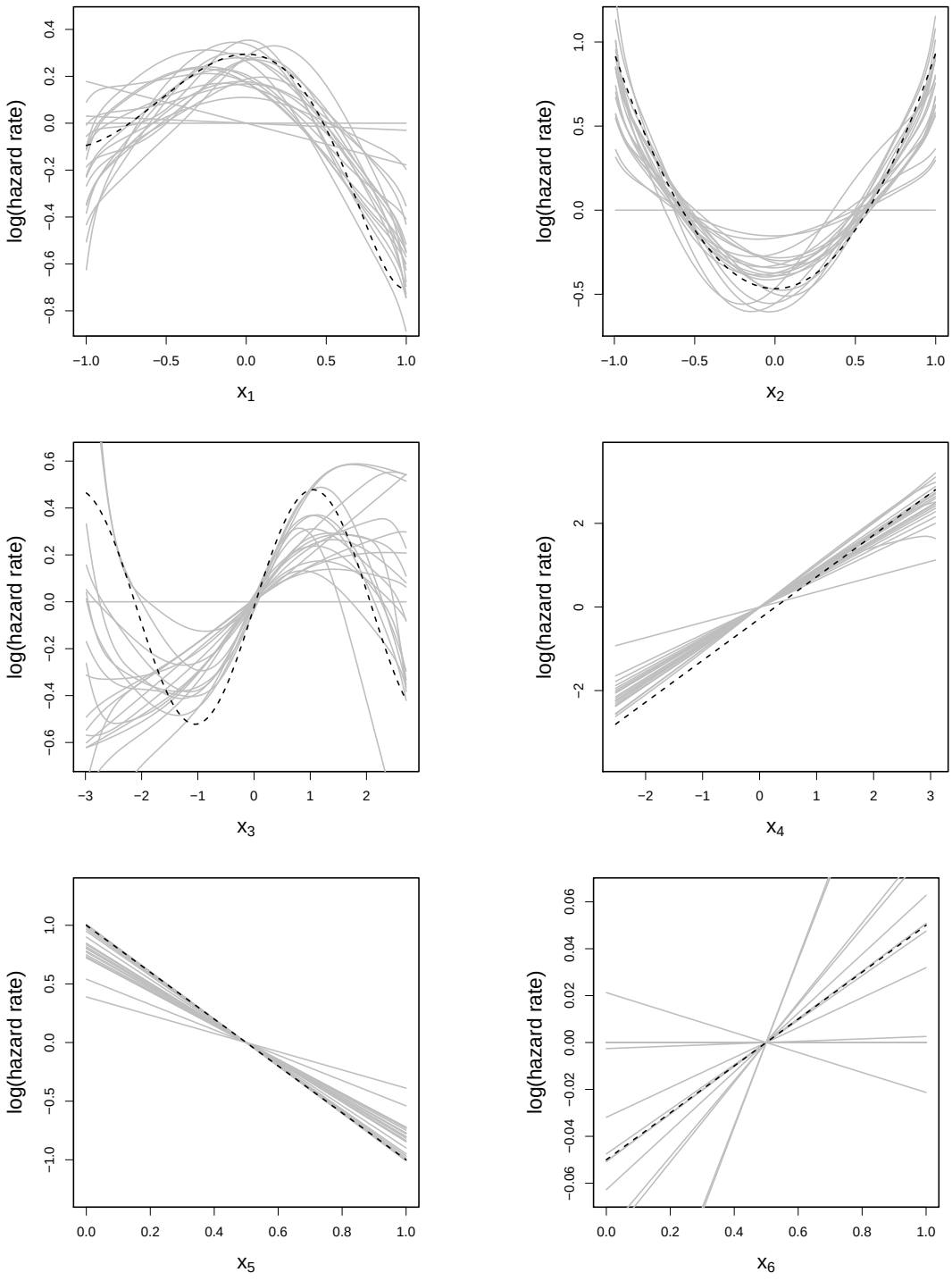


Figure 6.5.: **Model Choice: Simulation Scheme 2** – Estimation of covariate effects from 20 models (gray lines) and real effects (dashed lines). Effect estimates and real effects are centered. For $x_3$ and $x_4$ centering is only based on the "stable" areas $\in [-1, 1]$.

**Simulation Scheme 3** Simulation scheme three has only effective covariates in the model formula. The focus is on the goodness of the estimation of time-varying effects and on the performance of the simultaneous model choice algorithm. In Section 6.1.4 we showed that $\text{Cox}_{\text{flex}}$Boost has real problems to perform model choice. Note that for the plots in Figure 6.7 we did not take into account that the boosting procedure also selected time-varying effects for many covariates. Only the time-fixed effects are depicted except for the baseline hazard (see Fig. 6.6) and the time-varying effect of $x_6$.

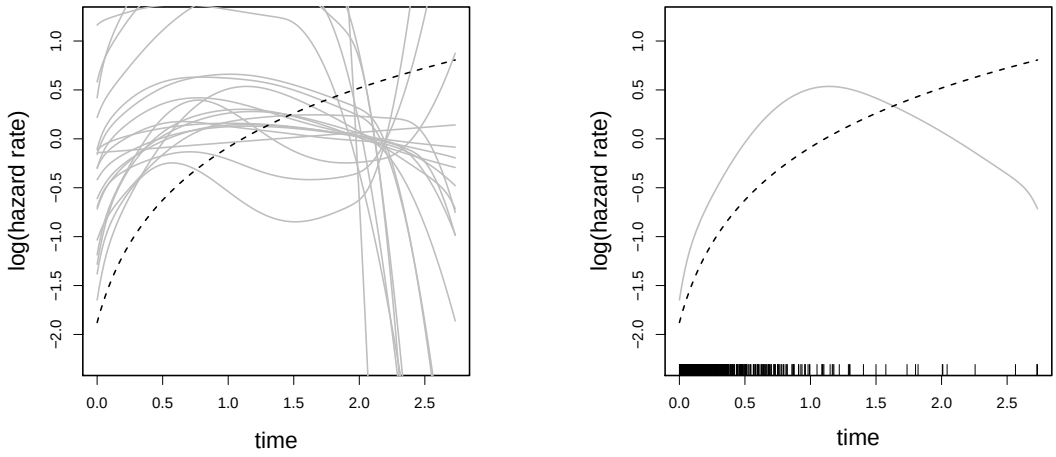


Figure 6.6.: **Model Choice: Simulation Scheme 3** – Left: Estimation of the baseline hazard from 20 models (gray lines) and real effect (dashed line). Right: Estimation of the baseline hazard for one model (gray line) and real effect (dashed line) together with rugs for the observed data. Effect estimates and real effects are centered.

Figure 6.6 shows the same lack of stability in the right tail as the estimated smooth effects of time did before. Here, the effect is even stronger since the distribution of the survival times is heavily right-skewed. The high amount of (early) censoring further increases the skewness.

The estimated effects from Figure 6.7 are in some cases almost as good as in the other simulations but they all tend to be a bit more instable. Especially the estimated effects of the second covariate ($x_2$) show big deviations from the true, linear function. The time-varying effect of $x_6$ (here depicted for $x_6 = 1$) suffers from the same problem as the baseline hazard, which is equal to the time-dependent effect if $x_6 = 0$.

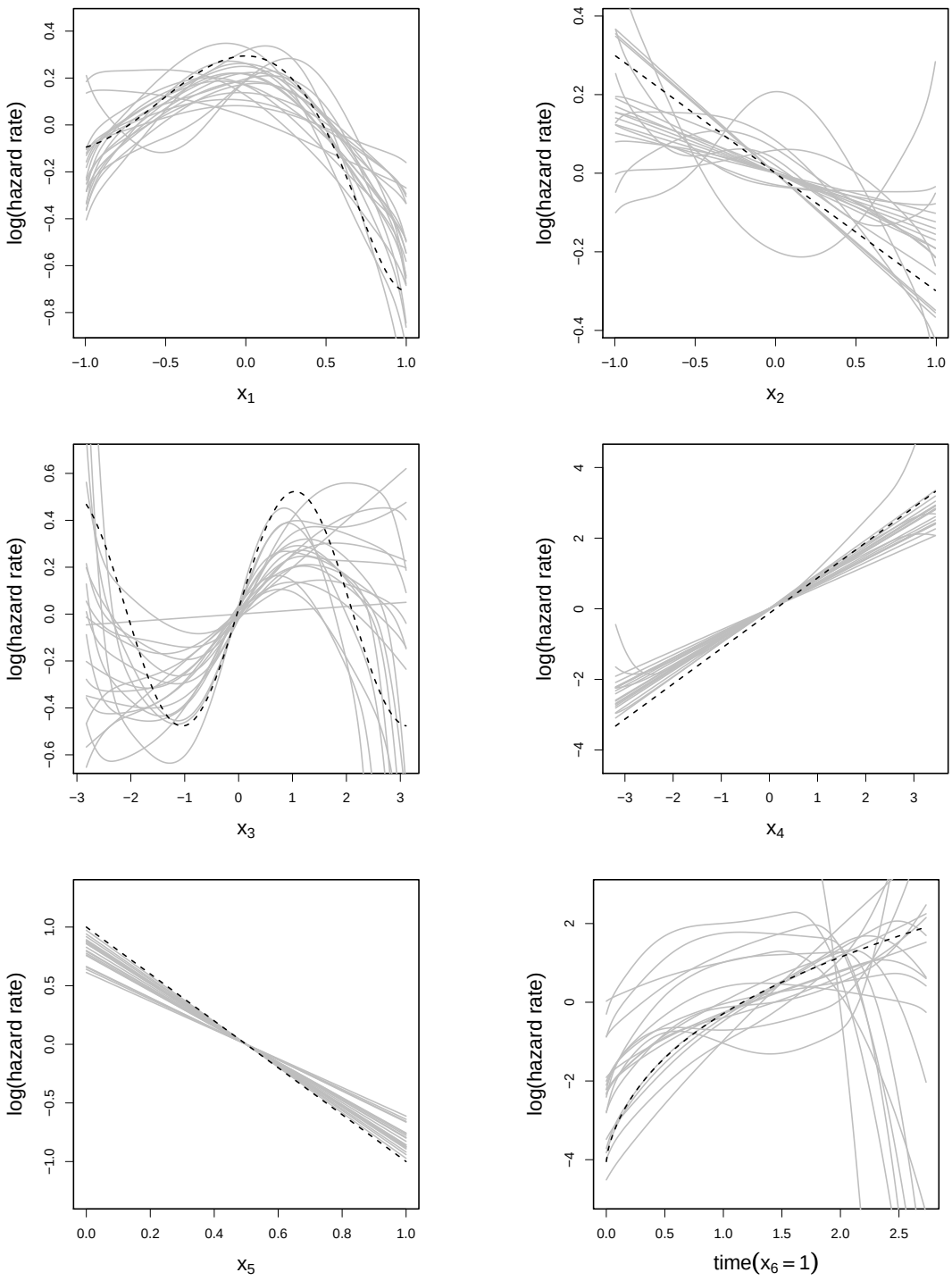


Figure 6.7.: **Model Choice: Simulation Scheme 3** – Estimation of covariate effects from 20 models (gray lines) and real effects (dashed lines). Effect estimates and real effects are centered. For $x_3$ and $x_4$ centering is only based on the "stable" areas $\in [-1, 1]$.

### 6.1.6. Simulation Results 3: Degrees of Freedom

**Initial Degrees of Freedom** First, we want to have a look on the initial degrees of freedom $\text{df}_j^{[0]}$ for all flexible base-learners $j \in \{1, \ldots, J\}$. Later, we investigate the changes of the degrees of freedom for each base-learner. Figure 6.8 shows the estimated degrees of freedom of the variable selection scheme in simulation 1. The initial degrees of freedom $\widetilde{df}_j$ used to specify the smoothing parameters $\lambda_j$ with (5.18) are all equal to four. The depicted degrees of freedom are computed after estimation of the corresponding base-learner using Formula (5.19). All initially estimated degrees of freedom $\text{df}_j^{[0]}$ are smaller than four (dashed line). Furthermore, all base-learners have some variability in the initially estimated degrees of freedom. However, we see a substantially smaller variation for $x_1$, $x_2$ and $x_7$ to $x_9$. These are the variables which are sampled according to a uniform distribution in contrast to the remaining variables which are standard normally distributed. As the latter have a much larger variation is seems that the variability of the (estimated) degrees of freedom $\text{df}_j^{[0]}$ depend on the variability of the covariate of the corresponding base-learner.

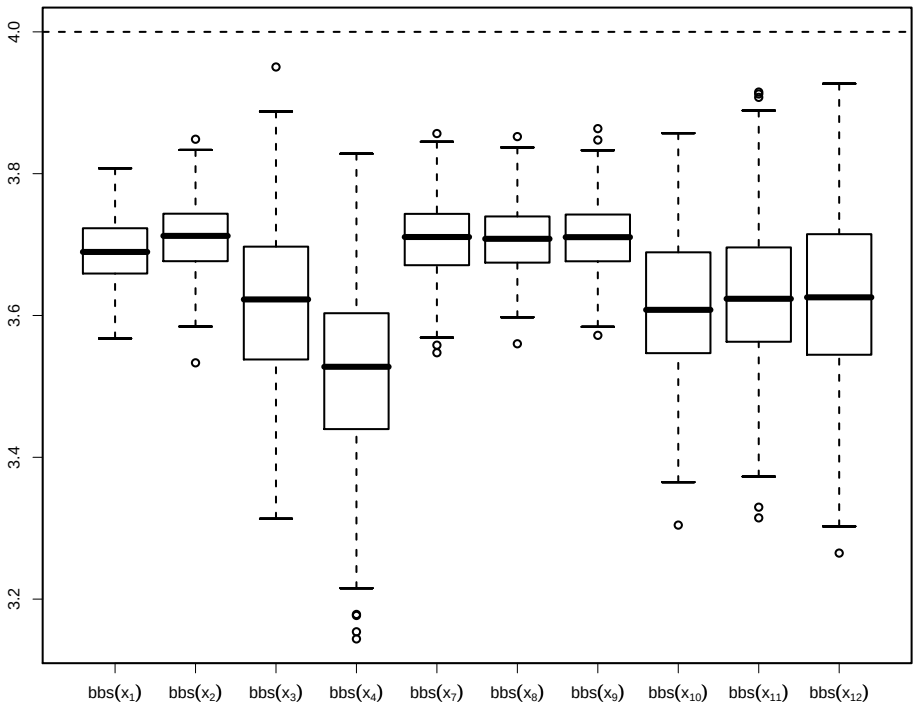


Figure 6.8.: **Variable Selection: Simulation Scheme 1** – Estimated degrees of freedom in the first boosting iteration (in 200 replicates) for all flexible base-learners and initially specified degrees of freedom (dashed line).

Figure 6.9, which depicts the initial degrees of freedom in the model choice scheme, shows that this effect is drastically reduced here. All base-learners show the same variability. The absolute amount of variation is reduced compared to the base-learners from variable selection. However, one needs to be aware of the reduced degrees of freedom that were initially specified to compute the smoothing parameters $\lambda$. In the model choice setting we only use *one* degree of freedom for the flexible part. Thus, the relative variation is much larger than in the variable selection scheme. The decomposition and the reduced degrees of freedom in the model choice approach may also be a reason why the variability of the covariates no longer influences the variance of the degrees of freedom.

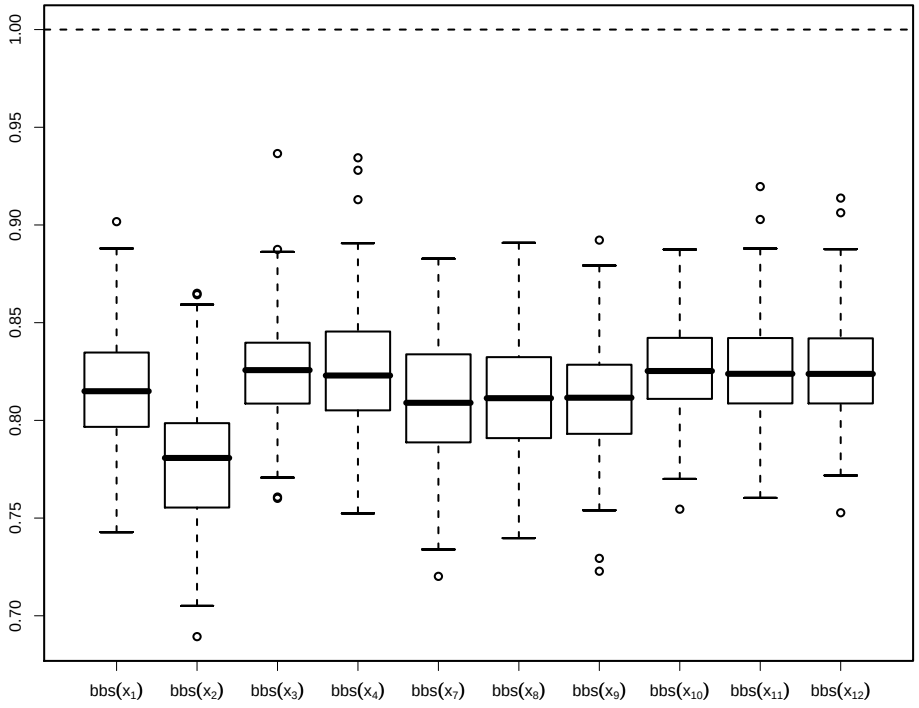


Figure 6.9.: **Model Choice: Simulation Scheme 1** – Estimated degrees of freedom in the first boosting iteration (in 200 replicates) for all flexible base-learners and initially specified degrees of freedom (dashed line).

In the upper part of Figure 6.10 we see the degrees of freedom in the variable selection scheme of simulation 2. The same base-learners as in simulation 1 have a higher variability with respect to their degrees of freedom. The base-learner for time has also highly variable degrees of freedom. This is consistent with the current results as time has again a higher variance than the uniformly distributed covariates.

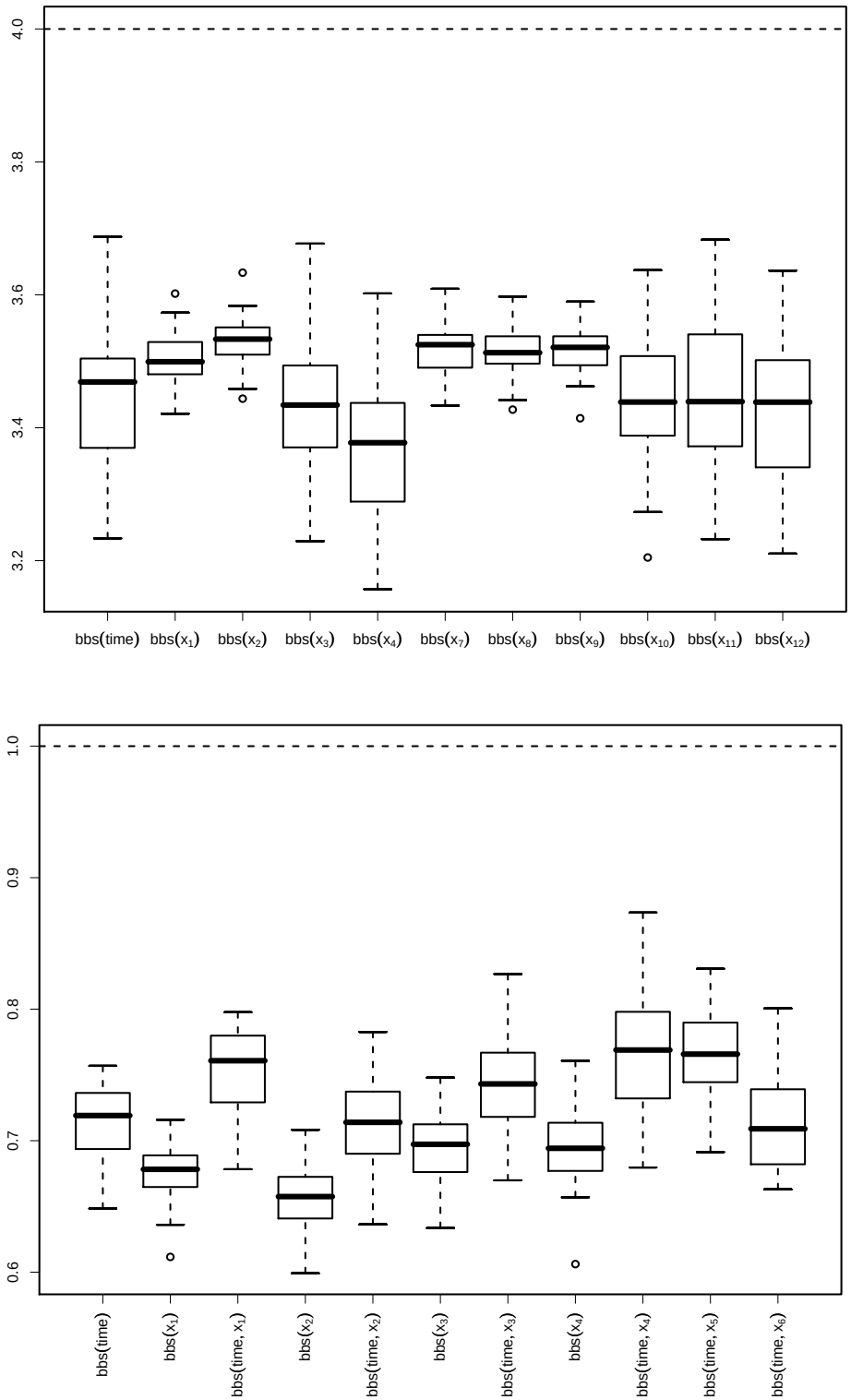


Figure 6.10.: **Simulation Scheme 2** – Estimated degrees of freedom in the first boosting iteration for variable selection (top) and model choice (bottom) (in 50 replicates) for all flexible base-learners together with initially specified degrees of freedom (dashed line).

The lower part of Figure 6.10 shows the initial degrees of freedom resulting from the model choice strategy in the second simulation. Here for all base-learners we have a relatively low variability. The flexible base-learners for $x_1$ to $x_4$ have a decreased median compared to simulation 1. This must be caused by the addition of a time-varying baseline effect to the simulated data. We see that all time-varying base-learners tend to have higher degrees of freedom. The same effect can be seen in Figure 6.11. Again, this seems to be induced by the higher variability of time. Perhaps this tendency of base-learners that depend on time led to the strong selection bias that we observed in Section 6.1.4. A possible solution would be to also standardize time (as covariate). In this case, one has to make sure that the integral is calculated over the right span, i.e. one has to use the real time as upper integral limit and not the standardized time. Moreover, the problem of higher degrees of freedom for time could possibly be fixed by refining the initial degrees of freedom as proposed in Section 5.3.2.

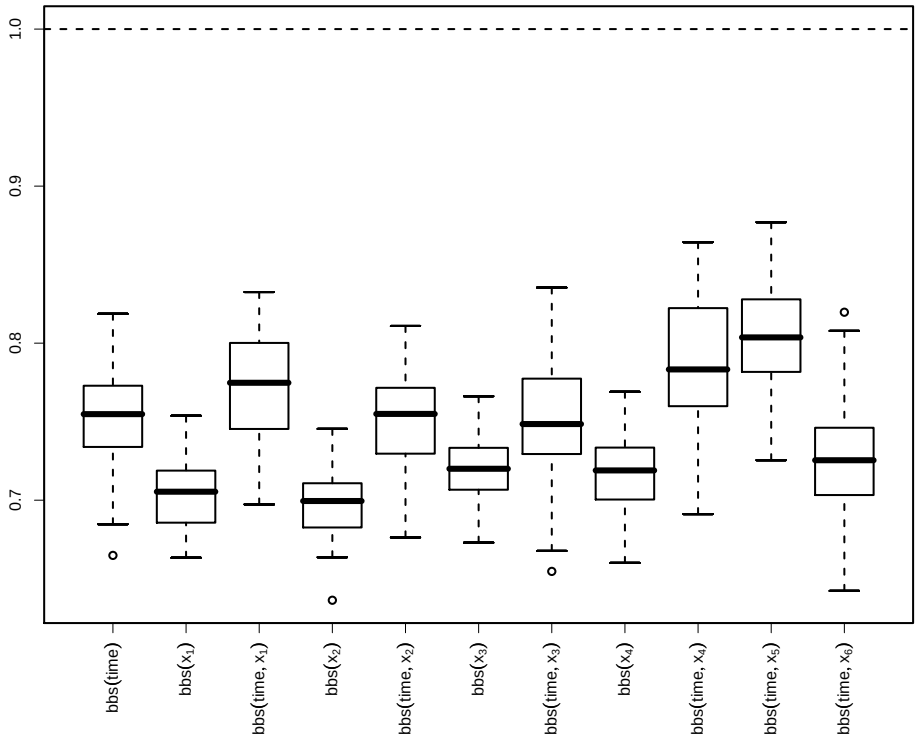


Figure 6.11.: **Model Choice: Simulation Scheme 3** – Estimated degrees of freedom in the first boosting iteration (in 50 replicates) for all flexible base-learners and initially specified degrees of freedom (dashed line).

**Changes of Degrees of Freedom** Next, we want to examine the dependency of the degrees of freedom on the boosting iteration. Therefore, we only chose

a small selection of graphs to illustrate the important findings. Most of the plots can be found in Appendix C.2. From Figure 6.12 we see again that the variability of the degrees of freedom increases with the variance of the covariates of the corresponding base-learners. In the upper plot the degrees of freedom for 200 replicates of the uniformly distributed variable $x_1$ are given. The normally distributed variable $x_3$ (with higher variance) is depicted in the lower graph. Furthermore, it can be noted that the degrees of freedom change only slightly with increasing iterations. The different lengths of the lines representing the degrees of freedom are due to different stopping iterations. Thus we see that there was one very long boosting run with about 700 iterations until the algorithm stopped and lots of shorter runs with about 200 iterations.

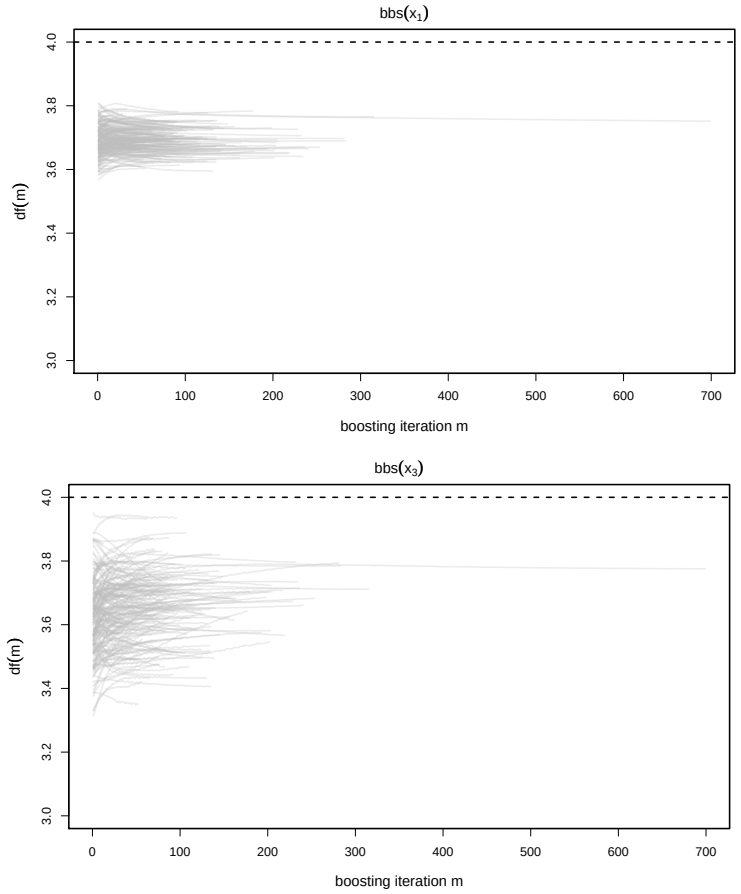


Figure 6.12.: **Variable Selection: Simulation Scheme 1** – Estimated degrees of freedom traced over the boosting steps for the flexible base-learners of $x_1$ and $x_3$ (in 200 replicates) with initially specified degrees of freedom (dashed line).

If we look at the same plots for the corresponding variable selection scheme (see Fig. 6.13) the assumed correlation of the variability with the variance of the

covariates vanishes. The boosting procedure now requires more iterations until convergence. This is due to the decomposition of the effects which leads to an increased number of base-learners that are needed to represent one flexible term. Perhaps it is notable that here we have a decrease in the degrees of freedom in the first few iterations and an increase afterwards. Over time this effect disappears. Thus, like before we cannot see major deviations from the estimated initial degree of freedom with respect to the whole boosting procedure.

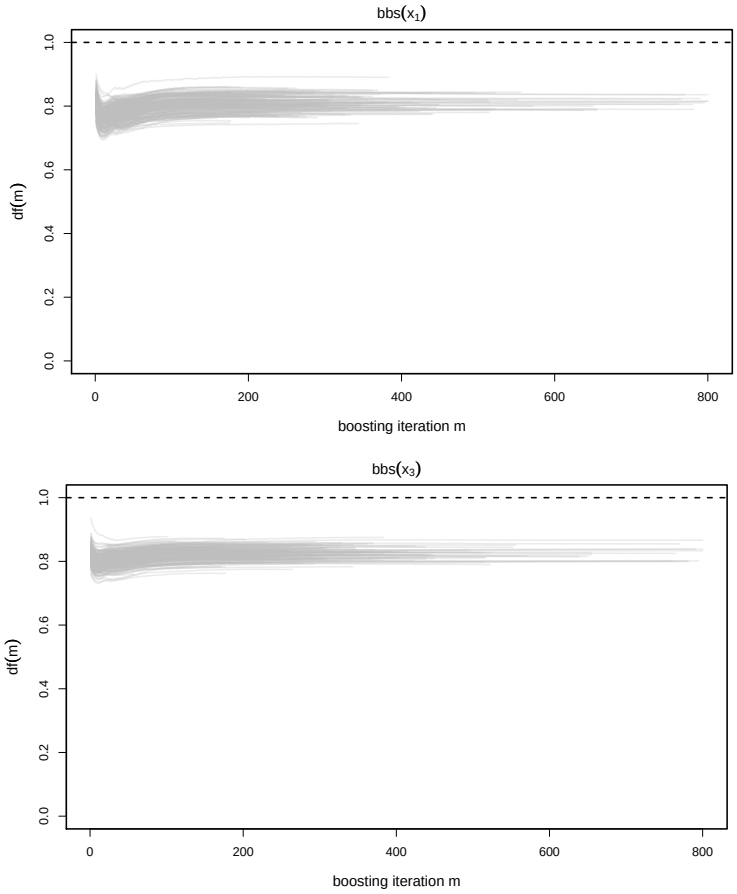


Figure 6.13.: **Model Choice: Simulation Scheme 1** – Estimated degrees of freedom traced over the boosting steps for the flexible base-learners of $x_1$ and $x_3$ (in 200 replicates) with initially specified degrees of freedom (dashed line).

## 6.2. Application: Prognostic Model for Surgical Patients

In the following section, we show an application of the two-stage stepwise procedure that we presented in Section 3.3. The aim was to build a prognostic model for patients with severe sepsis. In the second part of this section, we will apply the proposed $\text{Cox}_{\text{flex}}$Boost algorithm to the data.

Our analysis used data from a local database, which was initiated in 1993 in the Intensive Care Unit, Department of Surgery, Campus Großhadern, LMU Munich, Germany, for local benchmarking and quality control. The documentation period started on March 1st, 1993, and lasted until February 28th, 2005. We obtained relevant covariates reflecting the state of the patient on admission day, and the 90-day survival time for 462 patients with severe sepsis.

### 6.2.1. Application of Two-Stage Stepwise Procedure

#### Starting Model

Based on subject-matter knowledge, six of the covariates included in the original choice set should definitely be included in the final model. We therefore adapted the two-stage selection strategy from Section 3.3.2 to account for this fact. In a first step, we derived a starting model, where only the modeling possibility has to be chosen for each of the fixed covariates. This is implemented by accepting the best-fitting modeling alternative in step [iii] of the algorithm even if it does not reduce the conditional AIC ($\text{AIC}_c$) of the previous model. Application of the modified procedure to our data led to a model containing the effects presented in the upper part of Table 6.9 (in the order of inclusion). As one can see in the last two steps, we did not use a stopping criterion (e.g., increase of $\text{AIC}_c$) for the selection of the starting model. This is due to the fact that we just wanted to perform model choice but not variable selection. We will provide further discussion of the effects of variables in the starting model along the results of the final model in the next section.

#### Prognostic Model

After deciding on the optimal starting model, we derived the final prognostic model by applying the original two-stage procedure to the choice set of remaining covariates. Note that the fixed covariates are of course not subject to deletion in the backward deletion step [iv].

| Step | Covariate | Type of Covariate | Modeling Alternative used | $AIC_c$ |
|---|---|---|---|---|
| S1 | Apache II score | continuous | smooth term | 3186.21 |
| S2 | palliative operation for malignant disease | categorical | linear term | 3176.31 |
| S3 | age (in years) | continuous | linear term | 3168.55 |
| S4 | treatment period (after 2002 = 1) | categorical | linear term | 3159.74 |
| S5 | malignant primary diseases | categorical | linear term | 3161.22 |
| S6 | sex (male = 1) | categorical | linear term | 3163.17 |
| 1 | Horowitz ratio | continuous | smooth term | 3011.41 |
| 2 | hemoglobin concentration (in g/dl) | continuous | smooth term | 2970.52 |
| 3 | fungal infection | categorical | time-varying term | 2942.82 |
| 4 | creatinine concentration (in mg/dl) | continuous | linear term | 2922.19 |
| 5 | need for catecholamine therapy | categorical | linear term | 2913.80 |
| 6 | peritonitis | categorical | time-varying term | 2909.04 |
| 7 | surgery for thoracic disease | categorical | linear term | 2905.37 |
| 8 | need for renal replacement therapy | categorical | linear term | 2904.85 |
| – | direct postoperative admission | categorical | time-varying term | 2906.02 |
| – | need for artificial ventilation | categorical | linear term | 2906.43 |
| – | systolic blood pressure (in mm Hg) | continuous | linear term | 2906.44 |
| – | readmission | categorical | linear term | 2906.55 |
| – | pneumonia | categorical | linear term | 2906.55 |
| – | emergency admission | categorical | linear term | 2906.84 |

Table 6.9.: **Two-Stage Stepwise Procedure:** Variables in the starting model (S1 – S6); Variables in the choice set that were included (1 – 8) and variables in the choice set that were not included (–); Where applicable, step of inclusion, model alternative used at inclusion and conditional AIC at inclusion are given. For variables not included in the model the best-fitting alternative and corresponding $AIC_c$ in the last step of the procedure are given. Unless stated otherwise, categorical variables are dummy-coded with yes = 1.

Table 6.9 shows the full choice set, the model alternative selected in the stepwise procedure, and the conditional AIC obtained in the corresponding inclusion step. The covariate set mainly represents variables indicating renal, pulmonary and cardio-circulatory function, and nature and severity of the underlying disease. While most covariate labels are self-explanatory, some of them require supplementary explanation: "Apache II score" is a measure for the severity of disease determined within the first 24 hours of admission and the variable "Horowitz ratio" ($P_aO_2/F_iO_2$) describes the quality of lung function by referring the arterial partial oxygen pressure to the corresponding inspiratory oxygen concentration.

The effects of the selected smooth terms are plotted in Figure 6.14 and time-varying terms are shown in Figure 6.15. In the latter, one can clearly see the differences in the shapes of the log-baseline hazard rate for the variable "fungal infection" (present vs. absent). In particular, Figure 6.15 also reveals the ability of additive hazard regression models to address the problem of non-proportional hazards.

Table 6.10 shows the linear effects of the variables included in the prognostic model.

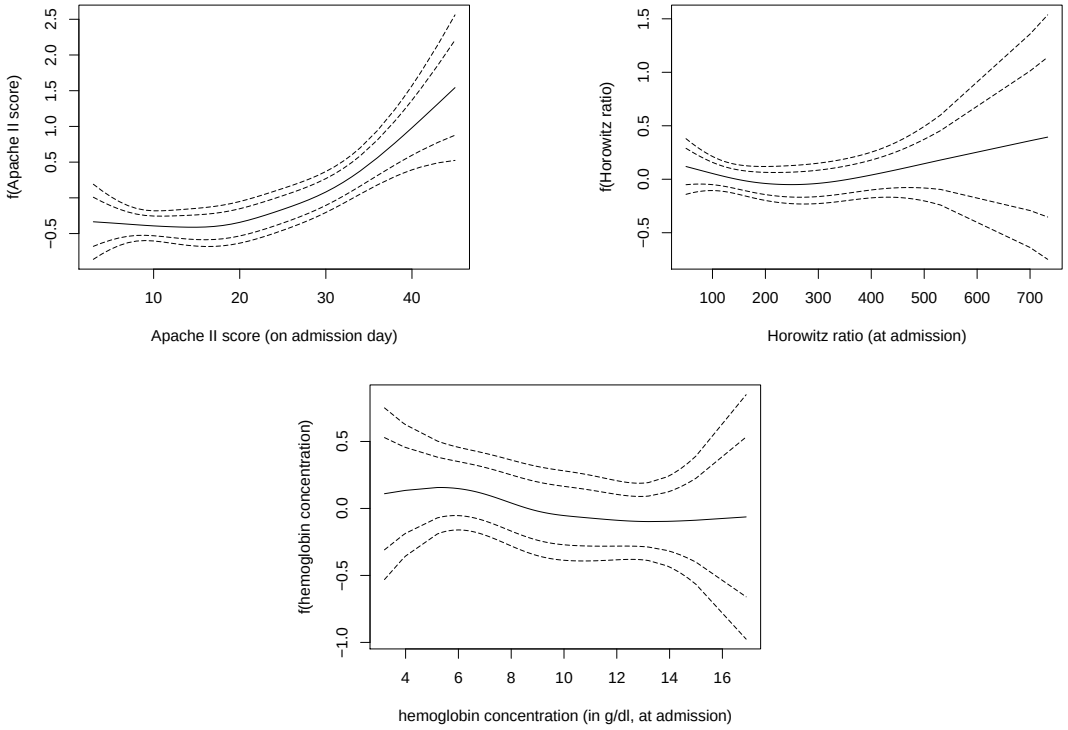


Figure 6.14.: **Two-Stage Stepwise Procedure:** Smooth terms for "Apache II score" (pre-selected in the starting model), "Horowitz ratio", and "hemoglobin concentration" in the prognostic model; Dashed lines are 80% and 95% point-wise confidence intervals.

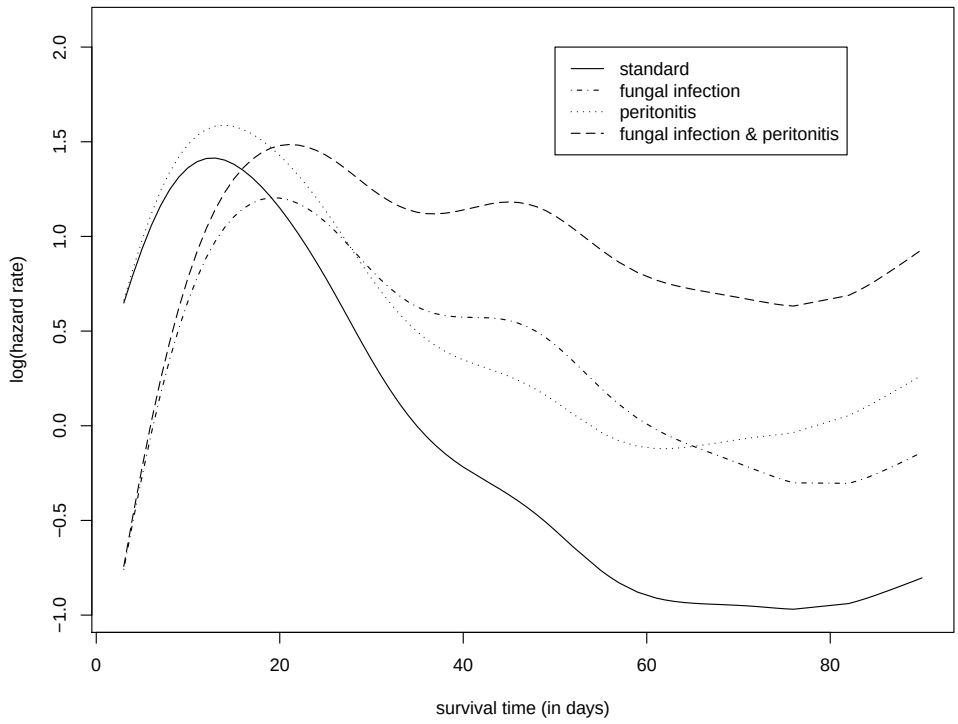


Figure 6.15.: **Two-Stage Stepwise Procedure:** log(baseline hazard rate) in subgroups defined by "fungal infection" (present vs. absent) and "peritonitis" (present vs. absent).

Relevant risk factors ($p < 0.05$) for a shorter survival time were palliative operation for malignant disease, age, creatinine concentration at admission, year of therapy and operation for thoracic diseases. According to 95% pointwise confidence intervals of smooth and time-varying terms, also a high Apache II score on admission day, and the nature and localization of the infection (fungal infection, peritonitis) were associated with an increased mortality. Gender, hemoglobin concentration, the need for renal replacement or catecholamine therapy, the Horowitz ratio or the primary diagnosis of a malignant disease were not associated with mortality.

Our results show that a clinically plausible prognostic model can be constructed on the basis of the suggested algorithm. This model can help the physician in charge to judge the true relevance of various clinical variables for patient outcome, and to adjust his therapeutic concepts according to individual risk profiles.

| Variable | $\beta$ | Std. Dev. | p-value | 95% conf. | interval |
|---|---|---|---|---|---|
| const | -6.361 | 0.5239 | $< 0.001$ | -7.3882 | -5.3342 |
| palliative operation for malignant diseases | 0.684 | 0.2021 | 0.001 | 0.2874 | 1.0798 |
| age (in years) | 0.018 | 0.0051 | $< 0.001$ | 0.0081 | 0.0282 |
| treatment period (after 2002 = 1) | -0.527 | 0.1465 | $< 0.001$ | -0.8139 | -0.2393 |
| malignant primary disease | -0.176 | 0.1478 | 0.233 | -0.4660 | 0.1133 |
| sex (male = 1) | -0.065 | 0.1370 | 0.636 | -0.3333 | 0.2038 |
| creatinine concentration (in mg/dl) | 0.116 | 0.0498 | 0.020 | 0.0186 | 0.2140 |
| need for catecholamine therapy | 0.349 | 0.3282 | 0.288 | -0.2947 | 0.9923 |
| surgery for thoracic diseases | 0.559 | 0.2086 | 0.008 | 0.1497 | 0.9677 |
| need for renal replacement therapy | -0.522 | 0.3534 | 0.140 | -1.2148 | 0.1708 |

Table 6.10.: **Two-Stage Stepwise Procedure:** Linear effects for prognostic model presented in the order of inclusion. Unless stated otherwise, categorical variables are dummy-coded with yes = 1.

### 6.2.2. Application of $\text{Cox}_{\text{flex}}$Boost

To asses the stability of the variable selection and model choice process of component-wise boosting, as implemented in $\text{Cox}_{\text{flex}}$Boost, we used 5 random subsamples each with 362 observations of the severe sepsis data from Großhadern. The remaining 100 observations from each subsample were used to determine the stopping iteration.

Before entering the model, all continuous covariates except time were standardized on intervals $[\frac{x_{\min}}{x_{\max}-x_{\min}}, \frac{x_{\max}}{x_{\max}-x_{\min}}] = [\frac{x_{\min}}{x_{\max}-x_{\min}}, \frac{x_{\min}}{x_{\max}-x_{\min}}+1]$, where $x_{\min}$ and $x_{\max}$ are the minimum and maximum of the respective covariate. This was done by dividing by the range of the covariate:

$$\widetilde{x_i} = \frac{x_i}{x_{\max} - x_{\min}} \tag{6.27}$$

Categorical covariates are dummy coded. Time enters the model unstandardized. Standardizing the covariates further by subtracting the minimum before dividing by the range leads to models that are no longer interpretable. To illustrate this, let $x^* = (x - x_{\min})/(x_{\max} - x_{\min})$ denote the standardized form of $x$ and $\hat{g}^*(t)$ denotes the estimated, standardized time-varying effect:

$$x^*\hat{g}^*(t) = \frac{x - x_{\min}}{x_{\max} - x_{\min}}\hat{g}^*(t) = x\frac{\hat{g}^*(t)}{x_{\max} - x_{\min}} - \hat{g}^*(t)\frac{x_{\min}}{x_{\max} - x_{\min}} \tag{6.28}$$

We see that the subtrahend in (6.28) is a function that only depends on $t$ and not on the actual value of $x$ and thus alters the baseline hazard in an unpredictable way.

As we already have realized in the simulation studies it seems that boosting with model choice is very instable (w.r.t. the selected base-learners) and prefers time-varying base-learners. Table 6.11 shows the selection frequencies of the base-learners.

In contrast to the two-stage stepwise procedure, $\text{Cox}_{\text{flex}}$Boost is not able to handle preset covariates. Such an approach could be included in the boosting framework, for example, by updating a set of mandatory covariates in every iteration (see e.g., Binder & Schumacher 2008). However, as this is not implemented in $\text{Cox}_{\text{flex}}$Boost so far, we did not use mandatory covariates but treated all covariates equal in the model choice procedure. This potentially can affect the inclusion of further covariates heavily. Furthermore, we did not use the complete data set but just subsamples. This again may have an influence on the selection and estimation of base-learners.

In the following paragraph, we compare the variables that were not added to the model when applying the two-stage stepwise (TSS) model with those that were not selected in the $\text{Cox}_{\text{flex}}$Boost model. From Table 6.11 we extracted the

variables that were never or only very seldom selected in the latter approach. Variables printed in **bold** were also not included in the TSS model. Mandatory covariates of the TSS model, which perhaps would not have been added in the TSS model if a stopping criterion would have been applied, are labeled in *italic*.

Covariates that were never selected by $\text{Cox}_{\text{flex}}$Boost are

- "Horowitz ratio",
- "**systolic blood pressure**",
- "**readmission**",
- "**direct postoperative admission**" and
- "**pneumonia**".

Sparsely selected variables, i.e., covariates that were only selected once in 5 models, are

- "*Malignant primary disease*",
- "*sex*",
- "hemoglobin concentration" and
- "renal replacement therapy".

"Surgery for thoracic disease" was only infrequently added (i.e., in 2 out of 5 models). In the two-stage stepwise model "surgery for thoracic disease" and "renal replacement therapy" were added as the last two variables. This could indicate that the inclusion of these covariates is at least arguable.
"Horowitz ratio" and "hemoglobin concentration" were considered to be influential in the TSS model based on the conditional AIC. However, if we look at Figure 6.14 we see that both effects only marginally depart from the zero-line, which would indicate that there is no effect at all. "Need for artificial ventilation" and "emergency admission" were not included in the TSS model. $\text{Cox}_{\text{flex}}$Boost instead selected these variables as time-varying effects. As both variables have just a relatively small linear time-varying effect (see Fig. 6.17) these effects could be artifacts as well. Only 10 out of 20 covariates can be regarded as influential covariates in the boosting model. The TSS procedure selected 14 covariates but 6 of these covariates were mandatory covariates. Thus, a candidate model without a set of compulsory covariates could lead to a sparser final model. We can conclude that both the algorithms TSS and $\text{Cox}_{\text{flex}}$Boost have a comparable strength for variable selection.

| base-learner | rel. freq. of selection |
|---|---|
| bols(time) | 0.60 |
| bbs(time) | 0.00 |

| | |
|---|---|
| bols(Apache II score) | 1.00 |
| bbs(Apache II score) | 0.00 |
| bols(time, Apache II score) | 1.00 |
| bbs(time, Apache II score) | 1.00 |
| bols(palliative operation) | 1.00 |
| bols(time, palliative operation) | 0.00 |
| bbs(time, palliative operation) | 0.20 |
| bols(age) | 1.00 |
| bbs(age) | 0.00 |
| bols(time, age) | 0.20 |
| bbs(time, age) | 0.60 |
| bols(treatment period) | 1.00 |
| bols(time, treatment period) | 0.20 |
| bbs(time, treatment period) | 0.00 |
| bols(malignant primary disease) | 0.00 |
| bols(time, malignant primary disease) | 0.20 |
| bbs(time, malignant primary disease) | 0.00 |
| bols(sex) | 0.00 |
| bols(time, sex) | 0.00 |
| bbs(time, sex) | 0.20 |
| bols(Horowitz ratio) | 0.00 |
| bbs(Horowitz ratio) | 0.00 |
| bols(time, Horowitz ratio) | 0.00 |
| bbs(time, Horowitz ratio) | 0.00 |
| bols(hemoglobin concentration) | 0.00 |
| bbs(hemoglobin concentration) | 0.00 |
| bols(time, hemoglobin concentration) | 0.00 |
| bbs(time, hemoglobin concentration) | 0.20 |
| bols(systolic blood pressure) | 0.00 |
| bbs(systolic blood pressure) | 0.00 |
| bols(time, systolic blood pressure) | 0.00 |
| bbs(time, systolic blood pressure) | 0.00 |
| bols(creatinine concentration) | 0.80 |
| bbs(creatinine concentration) | 0.00 |
| bols(time, creatinine concentration) | 0.80 |
| bbs(time, creatinine concentration) | 0.00 |
| bols(fungal infection) | 0.00 |
| bols(time, fungal infection) | 0.20 |
| bbs(time, fungal infection) | 1.00 |
| bols(emergency admission) | 0.20 |
| bols(time, emergency admission) | 0.80 |
| bbs(time, emergency admission) | 0.00 |

| | |
|---|---|
| bols(readmission) | 0.00 |
| bols(time, readmission) | 0.00 |
| bbs(time, readmission) | 0.00 |
| bols(direct postoperative admission) | 0.00 |
| bols(time, direct postoperative admission) | 0.00 |
| bbs(time, direct postoperative admission) | 0.00 |
| bols(pneumonia) | 0.00 |
| bols(time, pneumonia) | 0.00 |
| bbs(time, pneumonia) | 0.00 |
| bols(peritonitis) | 0.80 |
| bols(time, peritonitis) | 0.00 |
| bbs(time, peritonitis) | 0.00 |
| bols(catecholamine therapy) | 0.40 |
| bols(time, catecholamine therapy) | 1.00 |
| bbs(time, catecholamine therapy) | 0.00 |
| bols(artificial ventilation) | 0.00 |
| bols(time, artificial ventilation) | 1.00 |
| bbs(time, artificial ventilation) | 0.00 |
| bols(renal replacement therapy) | 0.00 |
| bols(time, renal replacement therapy) | 0.20 |
| bbs(time, renal replacement therapy) | 0.00 |
| bols(surgery thoracic disease) | 0.40 |
| bols(time, surgery thoracic disease) | 0.00 |
| bbs(time, surgery thoracic disease) | 0.00 |

Table 6.11.: **Cox$_{\text{flex}}$Boost** with model selection procedure for severe sepsis data in 5 sub-samples: Relative frequency of boosting models that selected the corresponding model term.

The degrees of freedom show the same properties as in the simulations. Figure 6.16 shows the degrees of freedom traced over $m_{\text{stop}}$ boosting iterations. Each plot represents one of the five subsamples and each line in the plot represents one base-learner. We see that the degrees of freedom split up into two subgroups in advanced iterations. Base-learners that were selected at any time in the boosting procedure are depicted in black, those which were not selected are given in gray. We see that all selected base-learners are in the lower group. If a *covariate* was chosen with *any* kind of base-learner all the degrees of freedom of other base-learners of this covariate are also reduced. Thus, all the base-learners corresponding to this covariate fall into the lower group. Base-learners of covariates that were never selected seem to have higher degrees of freedom and thus represent the upper group. Altogether the degrees of freedom are relatively stable over the iterations and are located in the same interval as observed in the simulations, i.e., roughly in $[0.6, 0.8]$.

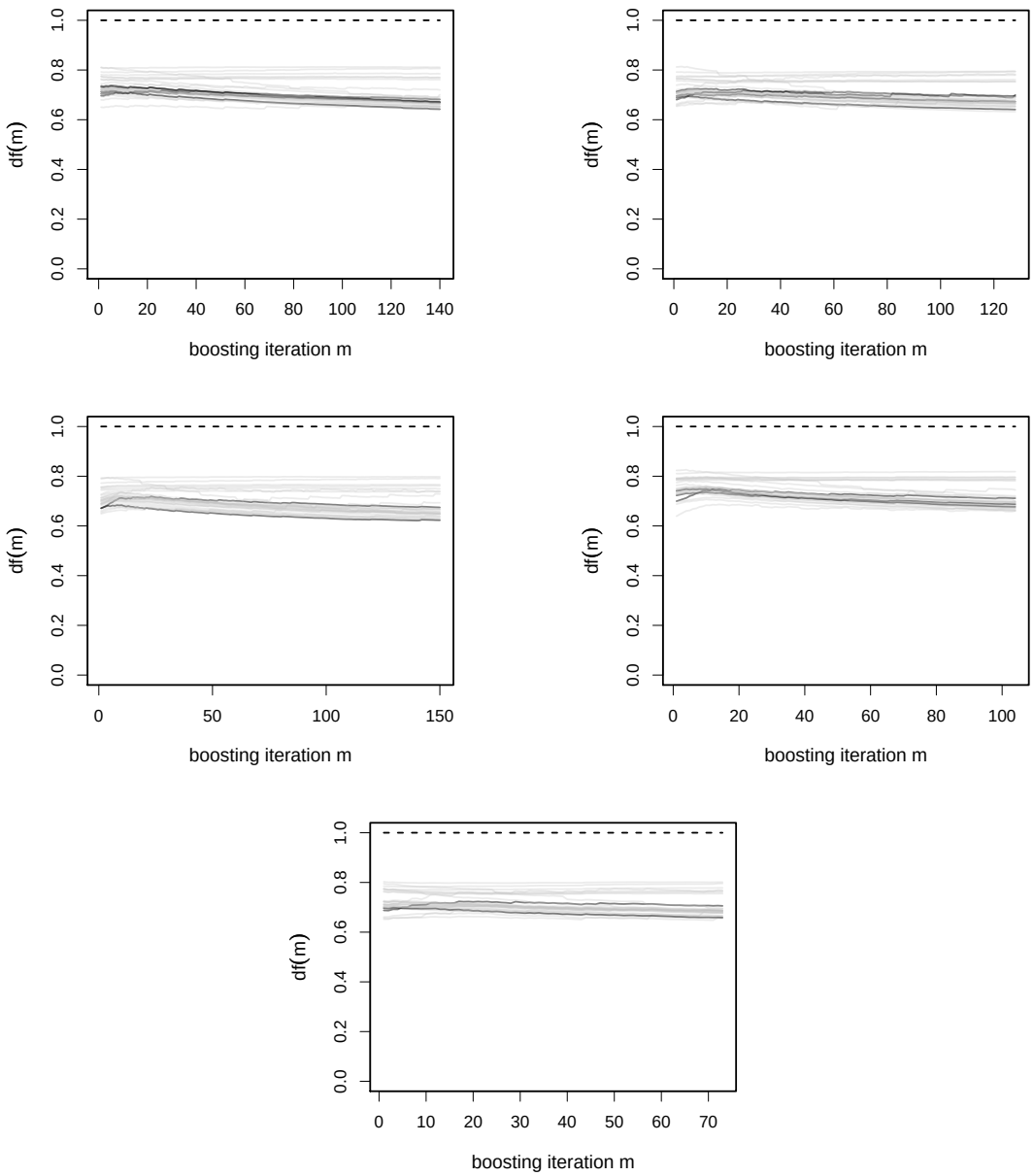


Figure 6.16.: **Cox$_{\text{flex}}$Boost:** Degrees of freedom with model choice procedure for surgical patients data in 5 sub-samples: Traces over $m_{\text{stop}}$ iterations are depicted for selected base-learners (black lines) and non-selected base-learners (gray lines).

As already stated in the simulation study, the resulting effects are hardly interpretable as many covariates are included with different modeling alternatives. They are added as smooth effects as well as time-varying effects. In Figure 6.17 the time-varying effects of four categorical covariates are depicted. We only chose those covariates that were selected in (almost) all of the five models. We

see that the log-baseline hazard without any of the 4 subgroups is only selected in 3 out of the 5 models. Furthermore, we see that all time-varying effects were added as linear base-learners. Only observations in the subgroup with "fungal infection" have a quadratic log-baseline hazard rate. This is consistent in all five models.

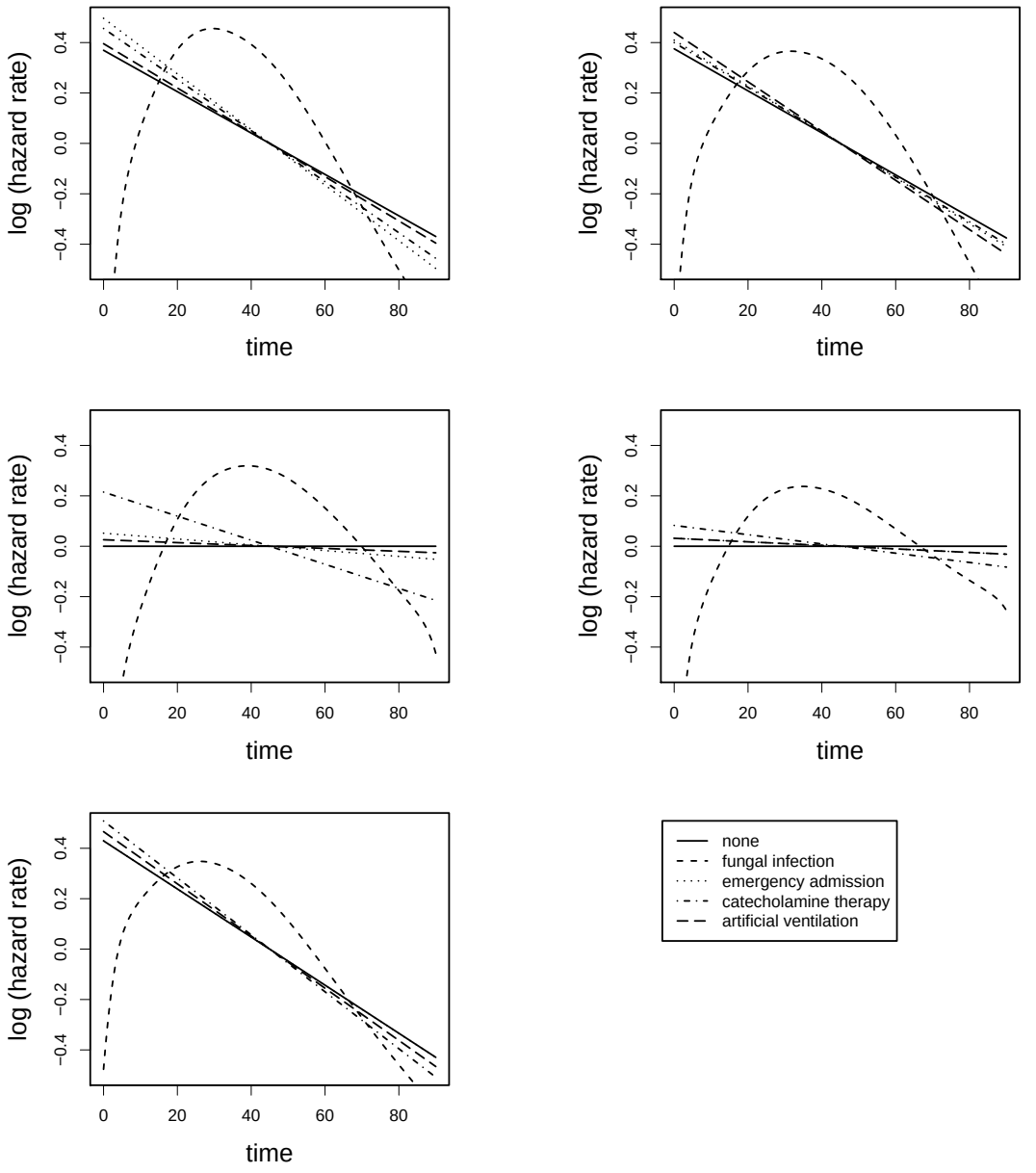


Figure 6.17.: **Cox**$_{\text{flex}}$**Boost** with model choice procedure for surgical patients data in 5 sub-samples: log(baseline hazard rate) in subgroups defined by "fungal infection" (present vs. absent), "emergency admission", "catecholamine therapy" and "artificial ventilation". All effects are centered.

As stated earlier, continuous covariates entered the model standardized. For some of the covariates, time-varying effects were also estimated. Hence, we plotted the effects of the covariates for a given time $t = \text{median}(t_i)$. Note that this does not reflect the individual combination of time $t$ and the covariate that was the basis of the estimation. Figure 6.18 shows the (time-constant) effects of 6 covariates that were frequently added to the model. "Age", "Apache II score" and "creatinine concentration" are continuous covariates. All three have a very high selection frequency for flexible time-varying effects. "Apache II score", for example, was added as a strong non-linear effect to the TSS model, whereas $\text{Cox}_{\text{flex}}$Boost estimated only a linear effect but added an additional time-varying effect. This increased flexibility (due to time-varying effects) cannot properly be depicted. All covariates given in Figure 6.18 have the same directions of the effects: Effects that were estimated positive in the TSS model are also estimated positive in $\text{Cox}_{\text{flex}}$Boost, negative effect estimates were again estimated negative. However, all depicted effects are smaller in $\text{Cox}_{\text{flex}}$Boost with respect to their norm. Note that this might not hold globally as we have additional time-varying effects that modify the given effects.

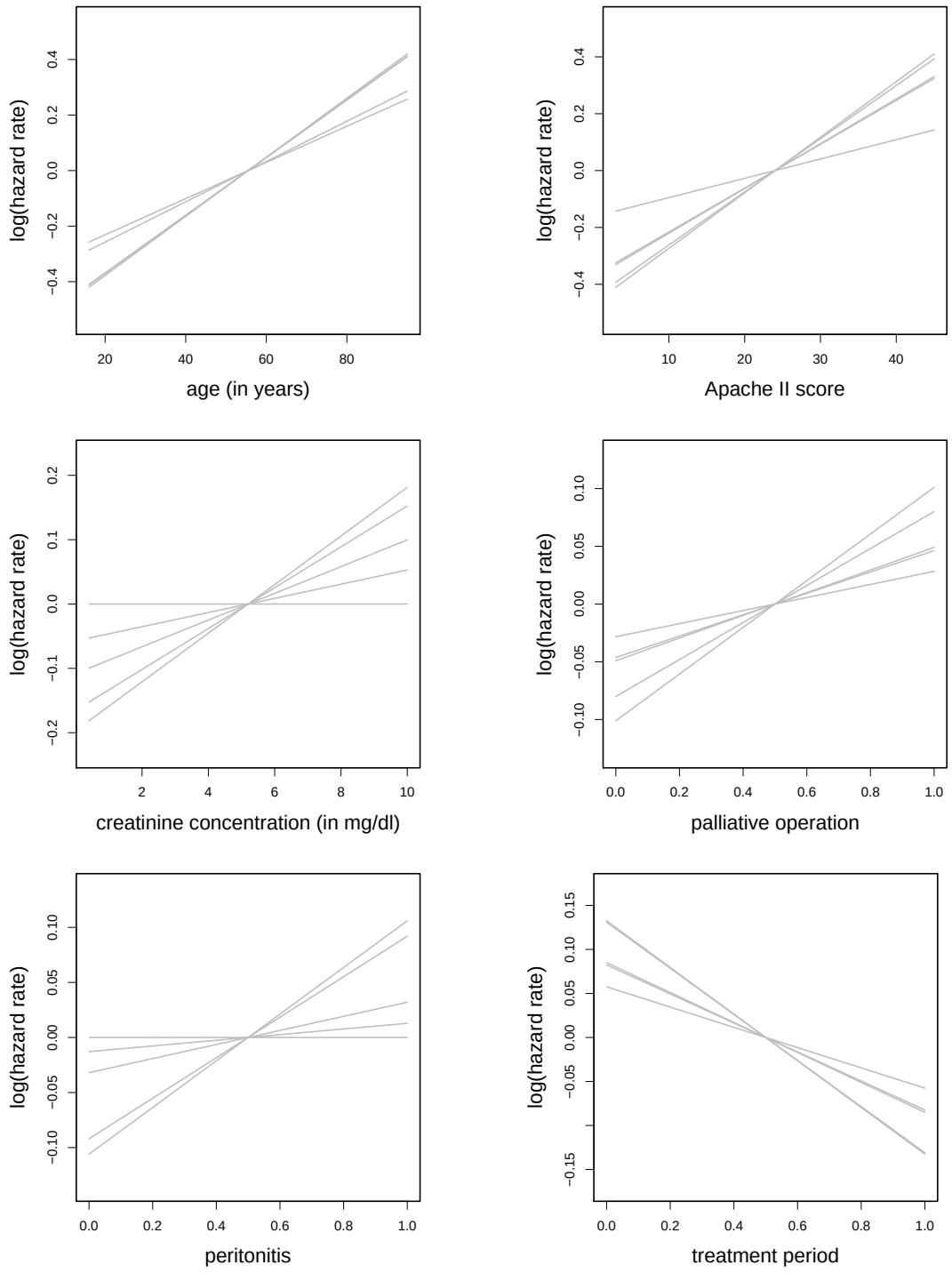


Figure 6.18.: **Cox$_{\text{flex}}$Boost:** Centered effects for the 5 sub-samples for 6 covariates that where considered to be influential by the model choice procedure for surgical patients data. "Age", "Apache II score" and "creatinine concentration" are continuous covariates, "palliative operation", "peritonitis" (present vs. absent) and "treatment period" (before vs. after 2002) are categorical covariates.

### 6.2.3. Comparison of Model Selection Strategies

From the results of the application of the two-stage stepwise procedure and $\text{Cox}_{\text{flex}}$Boost to the Großhadern data set of patients with severe sepsis we can conclude that both approaches have advantages with regard to different aspects:

- The two-stage stepwise procedure includes only *one* modeling alternative from a given set of different options, whereas $\text{Cox}_{\text{flex}}$Boost includes classically a variety of different modeling alternatives for one covariate. Thus, in the boosting context, the ability to interpret the model and the reliability of the model choice procedure suffer. A more sensible model choice scheme is needed in $\text{Cox}_{\text{flex}}$Boost without the selection bias in favor of time-varying effects.
- With respect to the variable selection procedure, we can conclude that both approaches have similar outcomes. In our application $\text{Cox}_{\text{flex}}$Boost tended to a sparser solution but this could be due to the starting model with mandatory covariates in the two-stage stepwise model.
- In settings with a large number of possible predictors, $\text{Cox}_{\text{flex}}$Boost is more convenient than the two-stage stepwise procedure as it runs fully automatized. Moreover, $\text{Cox}_{\text{flex}}$Boost is able to perform variable selection and model choice in data sets with $n \ll p$ and can even select more covariates $p$ than we have got observations $n$.
- At the moment $\text{Cox}_{\text{flex}}$Boost cannot include mandatory covariates. However, such extensions could be integrated in the algorithm. The two-stage stepwise procedure is easily extended in such a way as we showed in the application.

Altogether, we see that none of the approaches is superior to the other. Decisions have to be based on the qualities of the algorithms in the given situation. Especially in high-dimensional settings with many possible predictors, boosting with its robustness against overfitting and the built in regularization is clearly the preferred method.

# 7. Summary and Outlook

In this thesis, we derived boosting methods for survival models with time-varying effects. For that purpose we used the full likelihood (and not the partial likelihood) as basis. In a first attempt, we tried to derive an algorithm that is based on the functional gradient descent approach as introduced by Friedman (2001) (see Sec. 4.2.1). In Section 5.2 we developed the algorithm and showed that the resulting boosting approach, which utilizes the martingale residuals as negative gradient, is not working. We also tried to gain understanding of the reasons for this failure.

In a second try we implemented a likelihood-based boosting approach as proposed in Tutz & Binder (2006). The resulting $\text{Cox}_{\text{flex}}$Boost algorithm is presented in Section 5.3.1. Unfortunately, the likelihood-based approach, where one tries to maximize the (penalized) log-likelihood of the model directly, has some drawbacks with respect to the simplicity of further results: Functional gradient descent boosting fits all types of base-learners with the (penalized) least squares criterion. Hence, many results (e.g., the hat matrix and AIC) can be derived for the whole framework. On the contrary, $\text{Cox}_{\text{flex}}$Boost and other likelihood-based approaches maximize the likelihood of single base-learners with an offset consisting of the estimations of all previous iterations. Thus, for different likelihoods we need to derive many results anew, such as the hat matrix of the whole boosting procedure.

From the results of the simulation study in Section 6.1 one can see that component-wise boosting is a good variable selection procedure. To incorporate a model choice procedure in component-wise boosting, we applied the effect decomposition (4.14) or (4.15) for smooth effects or for time-varying effects, respectively. Furthermore, we assigned one degree of freedom to the resulting centered flexible base-learners to make the modeling alternatives comparable with respect to their flexibility (cf. Sec. 4.3.4). For the differentiation of linear and smooth effects, this provides good results. However, if one tries to distinguish between linear, smooth and time-varying effects at the same time, the procedure fails due to a selection bias in favor of time-varying base-learners. A possible solution could be to standardize the observed survival time that enters the model as predictor variable. This will be subject to future research.

Another approach to avoid the selection bias could try to make all base-learners comparable with respect to their degrees of freedom. Hence, the increased de-

grees of freedom that are assigned to base-learners of covariates with higher variance, including time-varying effects, would vanish (see Sec. 6.1.6). An idea to refine the initial degrees of freedom is presented in Section 5.3.2 (b).

The fact that the degrees of freedom change in every boosting iteration seems to be of minor importance because the changes are minimal and hence, they tend to be quite stable (see Sec. 6.1.6). Thus, from the current point of view no refinements are necessary here.

As discussed in Section 6.1.4, a suitable variable importance measure could be another helpful tool to asses and back up variable selection and model choice. Further research is needed here, as time-varying effects in particular tend to be very instable in the sparse, right tail (cf. Sec. 6.1.5). Hence, an importance measure cannot be based on the crude, estimated effects but must take the variability and instability of the estimations into account.

Further simulation studies should use integrated prediction error estimates as introduced in Section 6.1.2 to check the prediction performance of the proposed algorithm and to compare these results to other approaches or to investigate the effect of different estimation schemes. One possible alternative to the proposed model choice scheme in $\text{Cox}_{\text{flex}}$Boost could be to fit the model in a fashion like that proposed in the MFPT approach by Sauerbrei et al. (2007). This means, we fit a Cox-type model with *time-constant* but possibly smooth effects in a component-wise boosting framework. To estimate the model one could make use of $\text{Cox}_{\text{flex}}$Boost or apply the `mboost` package with the `CoxPH()` family. In a second step, one could try to add *time-varying* effects only for the subsample of selected variables from above, where the derived model is used as starting model (i.e., as offset). Thus, base-learners for time-varying effects, for example, could be restricted to covariates without smooth effects leading to a model that is better interpretable and perhaps overcomes the instability issues that we discussed above. Including time-varying effects for smooth effects would result in modeling an interaction of two functions: The function of the covariate and the function of time. This can be hardly ever estimated as we typically do not have enough data to fit the resulting interaction surface.

Another issue that arises frequently in medical applications is that some covariates are of clinically high importance and thus, should be included in the model by all means. These mandatory covariates can be incorporated in the boosting framework in such a way that these variables are updated in every boosting iteration (Binder & Schumacher 2008). This approach could also be included in $\text{Cox}_{\text{flex}}$Boost in future work.

However, a more urgent extension for $\text{Cox}_{\text{flex}}$Boost is to derive an appropriate information criterion such as the AIC (see Sec. 5.3.2 (d)). This would be very useful to asses the model and determine an appropriate stopping iteration in real data situations with only few observations. In this scenario, one does not

want to spend data for an independent validation sample that could be used to compute the empirical risk. To diminish the problem of data loss and still to be able to assess the model, bootstrap methods or cross-validation could be used. If we want to perform model choice when many possible predictors are present, we need numerous base-learners – a lot of them for time-varying effects. This drastically decelerates the estimation procedure and makes cross-validation or bootstrap practically infeasible. A suitable approximation of the AIC could solve this problem and allows to determine an appropriate stopping iteration, which is needed for variable selection and model choice.

# A. Derivation of Functional Derivative

In Section 5.2, a functional derivative of the loss function is needed for the functional gradient descent boosting approach for survival data with time-varying effects. With the loss function $\rho(y, f) = -\delta f(t, \boldsymbol{x}) + \int_0^t \exp(f(u, \boldsymbol{x}))du$ (5.1) the Hadamard derivative $\rho'_f(h)$ (5.4) can be derived as follows:

$$
\begin{aligned}
&\frac{\rho(y, f + sh_s) - \rho(y, f)}{s} \\
&\qquad = \frac{1}{s}\Bigg\{ - \delta \left[f(t, \boldsymbol{x}) + sh(\boldsymbol{x}, t)\right] + \int_0^t \exp\left[f(u, \boldsymbol{x}) + sh(\boldsymbol{x}, u)\right] du \\
&\qquad\qquad + \delta f(t, \boldsymbol{x}) - \int_0^t \exp\left[f(u, \boldsymbol{x})\right] du \Bigg\} \\
&\qquad = \frac{1}{s}\Bigg\{ - \delta \left[f(t, \boldsymbol{x}) + sh(\boldsymbol{x}, t) - f(t, \boldsymbol{x})\right] \Bigg\} \\
&\qquad\quad + \frac{1}{s}\Bigg\{ \int_0^t \underbrace{\exp[f(u, \boldsymbol{x}) + sh(\boldsymbol{x}, u)]}_{g_{\boldsymbol{x}}(s,u)} du - \int_0^t \exp\left[f(u, \boldsymbol{x})\right] du \Bigg\}
\end{aligned}
$$

For $s \searrow 0$ we get

$$
\begin{aligned}
&\frac{\rho(y, f + sh_s) - \rho(y, f)}{s} \\
&\qquad \stackrel{s \searrow 0}{\longrightarrow} - \delta h(\boldsymbol{x}, t) + \frac{\partial}{\partial s} \int_0^t g_{\boldsymbol{x}}(s, u) du \Bigg|_{s=0} \\
&\qquad \stackrel{a)}{=} - \delta h(\boldsymbol{x}, t) + \int_0^t \frac{\partial}{\partial s} g_{\boldsymbol{x}}(s, u) \Bigg|_{s=0} du \\
&\qquad \stackrel{b)}{=} - \delta h(\boldsymbol{x}, t) + \int_0^t h(\boldsymbol{x}, u) \exp\left[f(u, \boldsymbol{x})\right] du
\end{aligned}
$$

where

a) holds under regularity conditions

b) holds, as $g_{\boldsymbol{x}}(s,u)$ is of the form $\exp(a+s\cdot b)$ and thus it follows:

$$\frac{\partial}{\partial s} g_{\boldsymbol{x}}(s,u)\bigg|_{s=0} = b\cdot\exp(a+s\cdot b)\bigg|_{s=0} = b\cdot\exp(a)$$

with $a = f(u,\boldsymbol{x})$ and $b = h(\boldsymbol{x},u)$.

# B. Software

In the following, we describe important parts of the program and give a short introduction on how to use the software.

## B.1. Computational Considerations

$\text{Cox}_{\text{flex}}$Boost is implemented in the statistical software package `R` (R Development Core Team 2007). Some of our functions are based on functions from `mboost` (Hothorn et al. 2007). The algorithm and basic ideas were already discussed in Section 5.3.

### Estimation

We already gave a short introduction on the Fisher scoring algorithm, which is the classical algorithm to solve maximum likelihood problems as introduced earlier. Another optimization algorithm is the BFGS method, named after Broyden, Fletcher, Goldfarb and Shanno, who independently introduced this procedure in 1970. It is an important member of the Quasi-Newton methods. The Hessian is updated in a clever way instead of being computed in each iteration. More details on the BFGS method can be found, for instance, in Press et al. (1992).

We use the BFGS method, which is relatively fast and still accurate, to maximize the log-likelihood (5.14) for each base-learner in each boosting iteration. Optimizing a function with numerical methods always involves the repeated evaluation of the functions for different arguments (here for different coefficients $\boldsymbol{\beta}$). In each evaluation of (5.14) we have different values for $\boldsymbol{\beta}$ and thus, the integral in the log-likelihood needs to be computed anew. This shows that we need to integrate very often. Hence, a fast integration method can dramatically speed up the algorithm.

### B.1.1. Storage and Speed

Before we maximize the log-likelihood we compute the predictions of all base-learners that have been added until the current iteration of the boosting procedure and store the results. This prevents the maximization algorithm from computing these results anew in every iteration.

Let $m$ denote the current boosting iteration and $J$ is the number of base-learners. The subset $\mathcal{K} \subset \{1, \ldots, J\}$ represents the time-constant base-learners, whereas $\mathcal{L} \subset \{1, \ldots, J\}$ denotes the time-varying base-learners. It holds that $\mathcal{K} \cap \mathcal{L} = \emptyset$ and $\mathcal{K} \cup \mathcal{L} = \{1, \ldots, J\}$. The covariate that corresponds to the $j$-th base-learner is denoted by $\boldsymbol{x}_j = (x_{1,j}, \ldots, x_{n,j})'$ for time-constant base-learners. Time-varying base-learners depend on the covariate $\boldsymbol{x}_j(t) = (x_{1,j}(t), \ldots, x_{n,j}(t))'$. The base-leaner that is estimated in the current estimation step of the boosting iteration is denoted by $\xi \in \{1, \ldots, J\}$.

The integral for the $i$-th observation from Equation (5.14) can be decomposed in the following way, where $g_\xi(\widetilde{\boldsymbol{x}}_i(\tilde{t}); \boldsymbol{\beta})$ is the current base-learner, i.e., the base-learner that is estimated in the current loop, and $\hat{f}_j^{[m-1]}(\cdot),\ j \in \{1, \ldots, J\}$ is the function estimate for the $j$-th base-learner from the previous iteration.

$$
\begin{aligned}
&\int_0^{t_i} \exp\left\{ \hat{\eta}_i^{[m-1]}(\widetilde{\boldsymbol{x}}_i(\tilde{t})) + g_\xi(\widetilde{\boldsymbol{x}}_i(\tilde{t}); \boldsymbol{\beta}) \right\} d\tilde{t} \\
= &\int_0^{t_i} \exp\Bigg\{ \hat{\eta}^{[0]} + \sum_{k \in \mathcal{K} \setminus \xi} \hat{f}_k^{[m-1]}(x_{i,k}) && \text{(B.1a)} \\
&\qquad + \sum_{l \in \mathcal{L} \setminus \xi} \hat{f}_l^{[m-1]}(x_{i,l}(\tilde{t})) && \text{(B.1b)} \\
&\qquad + \hat{f}_\xi^{[m-1]}(x_{i,\xi}(\tilde{t})) + g_\xi(x_{i,\xi}(\tilde{t}); \boldsymbol{\beta}) \Bigg\} d\tilde{t} && \text{(B.1c)} \\
= &\quad \exp\Bigg\{ \underbrace{\hat{\eta}^{[0]} + \sum_{k \in \mathcal{K} \setminus \xi} \hat{f}_k^{[m-1]}(x_{i,k})}_{\text{TC}} \Bigg\} \cdot && \text{(B.1d)} \\
\cdot &\int_0^{t_i} \exp\Bigg\{ \underbrace{\sum_{l \in \mathcal{L} \setminus \xi} \hat{f}_l^{[m-1]}(x_{i,l}(\tilde{t}))}_{\text{TV}} + \underbrace{\hat{f}_\xi^{[m-1]}(x_{i,\xi}(\tilde{t})) + g_\xi(x_{i,\xi}(\tilde{t}); \boldsymbol{\beta})}_{\text{current}} \Bigg\} d\tilde{t} && \text{(B.1e)}
\end{aligned}
$$

The row-vector $\widetilde{\boldsymbol{x}}_i(\tilde{t})$ depicts all possible variables for the $i$-th observations. Note that in the integral we only have one single observation $i$. To speed up computations we compute all integration results vectorized for all observations at the same time. As the `integrate()` function in `R` is not able to integrate a vector of functions over a vector of different integration limits (lower limit

always zero but upper limit $t_i$ dependent on observation) at the same time we created an integration function based on the trapezoidal rule. For time-constant base-learners with the corresponding function estimates $\hat{f}_i^{[m-1]}(\cdot), i \in \mathcal{K} \setminus \xi$ we get from (B.1d) the additive predictor (vector)

$$\hat{\boldsymbol{\eta}}_{\text{TC}} = \exp\Big\{ \sum_{k \in \mathcal{K} \setminus \xi} \hat{f}_k^{[m-1]}(\boldsymbol{x}_k) \Big\} \cdot \exp\big\{ \hat{\eta}^{[0]} \big\}. \tag{B.2}$$

Time-varying base-learners are computed on equidistant time grids

$$\boldsymbol{G} = \begin{pmatrix} \boldsymbol{g}_1' \\ \vdots \\ \boldsymbol{g}_n' \end{pmatrix}, \text{ with } \boldsymbol{g}_i = (0, \frac{t_i}{sd}, \ldots, \frac{(\text{sd}-1)t_i}{sd}, t_i)', \; i = 1, \ldots, n, \tag{B.3}$$

where $sd + 1 = 101$ subdivisions of $[0, t_i]$ are used per default. This enables us to use the time-dependent prediction matrix

$$\hat{\eta}_{\text{TV}}(\boldsymbol{G}) = \exp \left\{ \begin{array}{ccc} \sum\limits_{l \in \mathcal{L} \setminus \xi} \hat{f}_l^{[m-1]}(x_{1,l}(g_{1,0})) & \cdots & \sum\limits_{l \in \mathcal{L} \setminus \xi} \hat{f}_l^{[m-1]}(x_{1,l}(g_{1,\text{sd}})) \\ \vdots & \ddots & \vdots \\ \sum\limits_{l \in \mathcal{L} \setminus \xi} \hat{f}_l^{[m-1]}(x_{n,l}(g_{1,0})) & \cdots & \sum\limits_{l \in \mathcal{L} \setminus \xi} \hat{f}_l^{[m-1]}(x_{n,l}(g_{n,\text{sd}})) \end{array} \right\} \tag{B.4}$$

for integration with the trapezoidal rule. Note that (B.4) is a matrix. Both, (B.2) and (B.4) do not depend on $\boldsymbol{\beta}$ and thus, can be used for all steps in the maximization routine.

The current base-learner $g_\xi(\cdot; \boldsymbol{\beta})$ and the corresponding function estimate $\hat{f}_\xi^{[m-1]}(\cdot)$ cannot be directly added to the vector (B.2) of time-constant base-learner predictions or the matrix (B.4) of time-varying base-learner predictions on a grid, as the parameter $\boldsymbol{\beta}$ changes in each iteration of the maximization procedure. However, for fixed values of $\boldsymbol{\beta}$ we can compute the integral from Equation (5.14): First, we predict the "current" part from Equation (B.1e). If the current base-learner is time-varying, we use the same grids $\boldsymbol{G}$ (B.3) for prediction. Then we add the predictions either to $\hat{\boldsymbol{\eta}}_{\text{TC}}$ or $\hat{\eta}_{\text{TV}}(\boldsymbol{G})$ for a time-constant or time-varying current base-learner, respectively. Now we can apply the trapezoidal rule on the resulting matrix of predictions for time-varying base-learners. This leads to a vector of estimated integrals for the time-varying base-learners for all observations. This vector now can be multiplied with the vector $\hat{\boldsymbol{\eta}}_{\text{TC}}$ of time-constant predictions which leads to the integral we wanted to obtain.

We could show in test runs that the storage of pre-computed predictions (where it is possible) and the usage of an integration method that is vectorized in the observations and in the integration variable (here: time $t$) is about 150 times quicker than the simple usage of `integrate()`, where the predictions are

evaluated in every integration anew. This is especially important as we have to integrate very frequently as this is needed to determine the parameter estimates for each base-learner in each boosting iteration with a numerical maximization method (see Sec. B.1). Note that the increased speed comes with an increased demand of memory.

### B.1.2. Checking Simulation of Survival Times

In Section 6.1.1 we introduced a simulation scheme for survival data. We implemented the method in `R` in the function `rSurvTime()`. To check the implementation, we simulated 1000 survival times given one covariate $\boldsymbol{x} = (x_1, \ldots, x_{1000})$, with $x_i \overset{i.i.d.}{\sim} U[-1, 1]$ using `rSurvTime()` with

$$\lambda_i = \lambda(t, x_i) \equiv \exp(0.01 - 10x_i),\ i = 1, \ldots, 1000, \tag{B.5}$$

i.e., with a time-constant hazard rate. This corresponds to survival times simulated according to a Cox-exponential model. Thus, we drew 1000 samples from a (conditional) exponential distribution with rates $\lambda_i$. Plotting density estimations of the 1000 replicates drawn from the conditional exponential distribution together with the sample from `rSurvTime()` shows that there seems to be no difference (Fig. B.1).

The same idea can be used to compare samples from `rSurvTime()` with a (special) hazard rate to samples from the theoretically equivalent Cox-Weibull model (see Bender et al. (2005) for details). Again, we plotted the density estimation of the 1000 replicates from the Weibull distributed survival times together with the the sample from `rSurvTime()`. Figure B.2 shows that there is no difference and thus, the two sample methods can be regarded equal.

## B.2. Using the Software

To show some features of the implementation of $\text{Cox}_{\text{flex}}$Boost we present a toy example in the following section. We presume that the necessary source files for $\text{Cox}_{\text{flex}}$Boost as supplemented on the attached CD are properly loaded before the source code of the example is used. Loading all required packages and the source files can be done by using the file "00_initialization.r" in the folder "R".

### B.2.1. Simulating Survival Times

Before we start explaining the functions that are used to estimate $\text{Cox}_{\text{flex}}$Boost models we explain how to sample data using the attached function `rSurvTime()`.

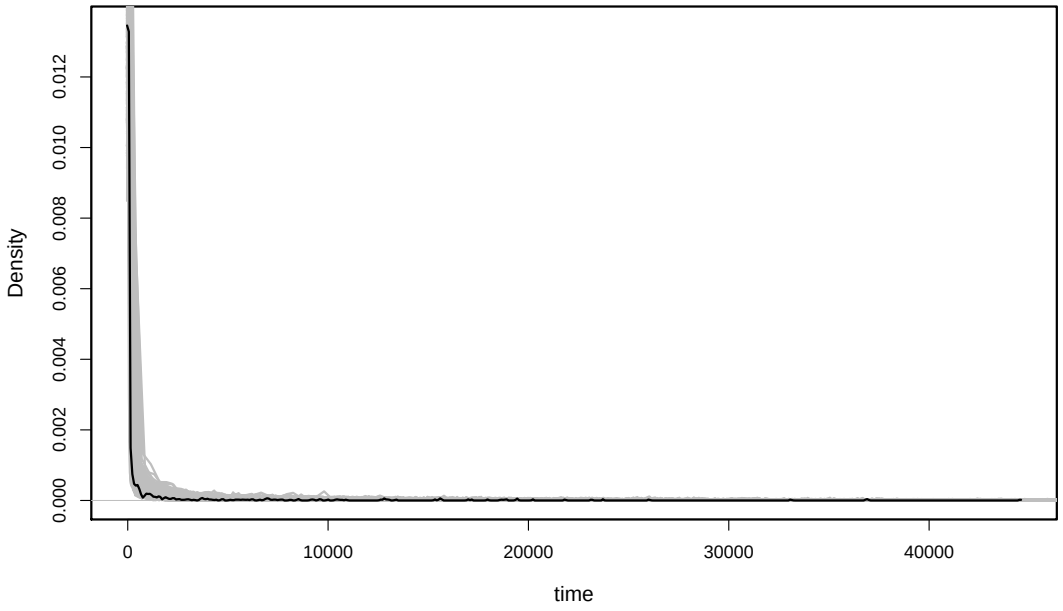


Figure B.1.: **Checking simulation algorithm:** Density estimation of 1000 simulated survival times using `rSurvTime()` (black line) and 1000 replicates of 1000 survival times simulated according to a Cox-exponential model (gray lines).

To make these results reproducible we start the analysis by setting a random but known seed:

```
R> seed <- sample(1:10000, size = 1)
R> seed

[1] 6102

R> set.seed(seed)
```

Now, we sample three independent covariates according to a uniform distribution on the interval $[-1, 1]$:

```
R> X <- matrix(NA, nrow = 400, ncol = 3)
R> X[, 1] <- runif(400, -1, 1)
R> X[, 2] <- runif(400, -1, 1)
R> X[, 3] <- runif(400, -1, 1)
```

The next step is to specify an R-function for the hazard rate $\lambda$:

```
R> lambda <- function(time, x) {
      exp(log(time) + 0.7 * x[1] + x[2]^2 + 0 * x[3])
  }
```

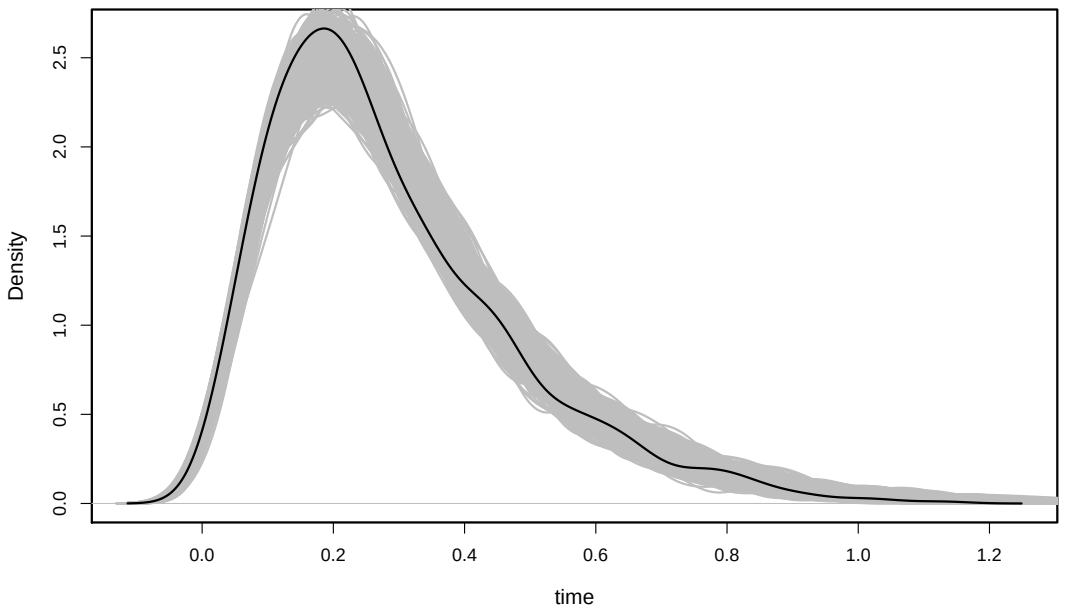


Figure B.2.: **Checking simulation algorithm:** Density estimation of 1000 simulated survival times using `rSurvTime()` (black line) and 1000 replicates of 1000 survival times simulated according to a Cox-weibull model (gray lines).

Note that the function argument `x` expects a *vector* that contains only one row of the design matrix `X`, i.e., `x = X[i,]` for changing `i`. Obviously, the term `0 * x[3]` is not needed and is just specified here to stress that this covariate has no effect. As we integrate the function `lambda` over `time` the function must be vectorized in `time`. Thus, if we have time-constant hazard rates a nice trick to achieve vectorization is to specify `0 * time` explicitly. As expected, this results in a vector with the length of the input vector `time` to be returned.

To sample censored survival data we need an additional function for the censoring mechanism. In this toy example we use $C \overset{i.i.d.}{\sim} \text{Expo}(1/\mu)$, where $\mu = \mathbb{E}(C)$ can be chosen in such a way that we get the intended amount of censoring:

```
R> cens_fct <- function(time, mean_cens) {
      censor_time <- rexp(n = length(time), rate = 1/mean_cens)
      event <- (time <= censor_time)
      t_obs <- apply(cbind(time, censor_time), 1, min)
      return(cbind(t_obs, event))
  }
```

As one can see, the function expects the vector of sampled, uncensored times `time` and returns the observed times and the non-censoring indicator.

Now, we can make a call to the function `rSurvTime()` to sample random survival times:

```
R> data <- rSurvTime(lambda, X, cens_fct, mean_cens = 5)
```

`rSurvTime()` requires the functions `lambda()` and `cens_fct()` as specified above. In addition we need to pass the covariates `X` to the sample function. If the censoring function requires further arguments we can specify them as further *named* arguments: Here we specified `mean_cens = 5` as additional argument.

In our setting we get 80 % non-censored observations. Figure B.3 depicts the sampled data.

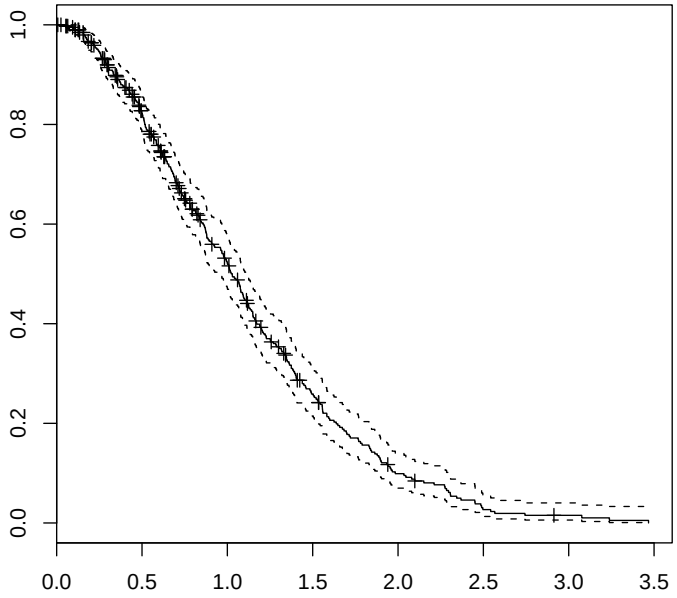


Figure B.3.: **Toy Example:** Kaplan-Meier curve for the sampled data

## B.2.2. Estimating the models

To estimate the model using `cfboost()` we have to specify the controls for the boosting algorithm

```
R> ctrl <- boost_control(mstop = 100, nu = 0.1, risk = "oobag",
      trace = FALSE, hardStop = TRUE)
```

The following options are some of the most important ones:

- `mstop` specifies the initial number of boosting iterations,
- `nu` defines the step-size and
- `risk` determines how the empirical risk should be computed. It can take the values `"inbag"`, i.e., the risk is computed for the learning sample, `"oobag"`, i.e., the risk is computed on the out-of-bag sample and `"none"`. At the moment it is only reasonable to use the option `"oobag"` as no other means of determining $m_{\text{stop}}$ is implemented so far.
- `trace` indicates if status information should be printed in the boosting procedure and
- `hardStop` indicates if the initial value of `mstop` is the maximal number of iterations (`hardStop = TRUE`) or if it should be increased if the algorithm did not convergence until `mstop` (only possible with `risk = "oobag"`).

For an explanation of the option `hardStop` see also Section 6.1.3. Note that it is generally very useful to use `trace = TRUE` to see the progress of the boosting procedure and to see that `R` is working as the algorithm can be quite time-demanding.

Furthermore, weights can be used to specify the learning sample (`weights == 1`) and the out-of-bag sample (`weights == 0`), which serves as the validation sample and is used to determine the stopping iteration $m_{\text{stop}}$. We choose the weights such that the first 300 observations are used as learning sample and further 100 observations serve as validation sample:

```
R> weights <- c(rep(1, 300), rep(0, 100))
```

Note that at the moment no weights other than zero or one are allowed in `cfboost()`.

Now we can estimate the model with flexible base-learners as in the variable selection scheme:

```
R> var_sel <- cfboost(Surv(time, event) ~ bbsTime(time) +
      bbs(x.1) + bbs(x.2) + bbs(x.3), data = data,
      control = ctrl, weights = weights)
```

Per default flexible P-spline base-learners `bbs()` consist of B-splines of degree three (`degree = 3`) with a difference penalty of order two (`differences = 2`). They have four degrees of freedom (`df = 4`) and 20 inner knots (`knots = 20`). The base-learner `bbsTime()` is just a wrapper to the P-spline base-learner `bbs()` with the option `timedep = TRUE`.

In the same manner as in the variable selection scheme we could use `cfboost()` for model choice:

```
R> mod_choice <- cfboost(Surv(time, event) ~ bolsTime(time)
       + bbsTime(time, df = 1, center = TRUE)
       + bols(x.1) + bbs(x.1, df = 1, center = TRUE)
       + bolsTime(time, z = x.1)
       + bbsTime(time, z = x.1, df = 1, center = TRUE)
       + bols(x.2) + bbs(x.2, df = 1, center = TRUE)
       + bolsTime(time, z = x.2)
       + bbsTime(time, z = x.2, df = 1, center = TRUE)
       + bols(x.3) + bbs(x.3, df = 1, center = TRUE)
       + bolsTime(time, z = x.3)
       + bbsTime(time, z = x.3, df = 1, center = TRUE),
       data = data, control = ctrl, weights = weights)
```

Here we use the option `center = TRUE` to apply the decomposition of the flexible effects (cf. Sec. 4.3.4) and define the centered effects with one degree of freedom. As stated in Section 4.3.4, for the parametric part separate linear base-learners are added with `bols()`. Here, the model choice scheme is not only applied to smooth covariate effects but it is also used to add (decomposed) time-varying effects for each covariate. For `x.1` we add for example: `bolsTime(time, z = x.1) + bbsTime(time, z = x.1, df = 1, center = TRUE)`. Here, the effect modifier `time` is specified as linear (`bolsTime()`) or flexible effect (`bbsTime()`) and `x.1` is specified as covariate `z`, which is modeled as interaction: $g(t) \cdot z$ (see Sec. 2.3.2).

## B.2.3. Processing the Output

After the model is fitted it is essential to determine the appropriate stopping iteration $m_{\text{stop}}$. This can be done with a call to

```
R> mstop(var_sel)

[1] 59
```

Another way to get information on the fitted model together with $m_{\text{stop}}$ is:

```
R> summary(var_sel)

        'summary.cfboost' is a.t.m. a wrapper to 'print.cfboost'


        CoxFlexBoost:
        Additive Survival Models with Time-Varying Effects
        Fitted via Likelihood-Based Boosting

Call:
cfboost.formula(formula = Surv(time, event) ~ bbsTime(time) +
    bbs(x.1) + bbs(x.2) + bbs(x.3), data = data, weights = weights,
```

```
    control = ctrl)

Number of boosting iterations: mstop = 100
Step size:  0.1
Offset:  -0.1840850

Size of learning sample: 300
        of test sample: 100

minimum risk: 83.08577
        in iteration  59

Number of selections in 100 iterations:
        bbs(time):        45
        bbs(x.1):        27
        bbs(x.2):        22
        bbs(x.3):         6
```

The call to `summary()` returns almost the same results as `print()`. The former only returns the additional information of the number of selections for each base-learner. It is planned to include more information in the `summary()` of a `cfboost` model in near future.

Classically, the function `mstop()` determines the optimal stopping iteration based on the empirical risk in the validation sample. To plot the empirical risk one can use

```
R> plot(risk(var_sel))
```

which produces Figure B.4. This function is also available if no out-of-bag sample is specified.

To proceed with the examination of the model a subset method is required:

```
R> var_sel_ms <- var_sel[mstop(var_sel)]
```

`var_sel_ms` now can be used for further analyses. Classical methods for models are available, such as

```
R> coef(var_sel_ms)
```

which extracts the coefficients. As we used P-spline base-learners, we do not show the results of the call to `coef()` as these are not very explanatory. However, this function can be used to extract the estimated coefficients. Further, methods to extract the fitted hazard rate (`type = "hazard"` (default)) or the fitted log-hazard rate (`type = "log-hazard"`) exist:

```
R> fitted(var_sel_ms)[1:10]
```

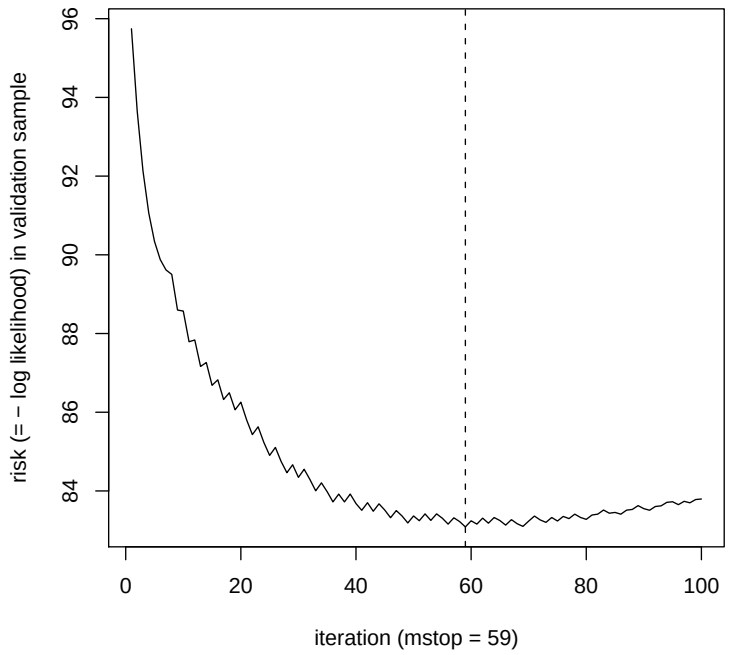


Figure B.4.: **Toy Example:** Empirical risk in validation sample with stopping iteration

```
 [1] 0.7282824 2.4524128 0.6268950 0.9164832 1.8997813 0.6338802
 [7] 1.1582462 1.4565169 1.3109424 1.6623828
```

In our case, the fitted hazard rate for the 300 in-bag samples is returned. For simplicity we only print the first 10 fitted values. The `predict()` function has the same options as `fitted()`. The call returns predictions for new observations, here for the first out-of-bag observation:

```
R> predict(var_sel_ms, newdata = data[301, ])
```

```
[1] 0.5685829
```

Using `predict(...,type = "log-hazard")` we created, for example, Figure C.1. A simple out-of-the-box plotting method for the results is also supplied:

```
R> plot(var_sel_ms)
```

In the variable selection scheme this leads to nice, interpretable plots as depicted in Figure B.5. For the model choice setting, better plots can be created using the `predict()` function as mentioned above. Supplying nice standard plots in this situation is going to be implemented in future versions of the the `plot()` function. The Figure B.5 shows quite good estimation results (which is not too

surprising in this simple setting). Looking at the frequency of the base-learner selection until $m_{\text{stop}} = 59$, e.g., by using `summary(var_sel_ms)` or

```
R> freq.sel(var_sel_ms)

Number of selections in 59 iterations:
        bbs(time):        28
        bbs(x.1):         18
        bbs(x.2):         13
        bbs(x.3):          0
```

we see that `x.3` was never selected.

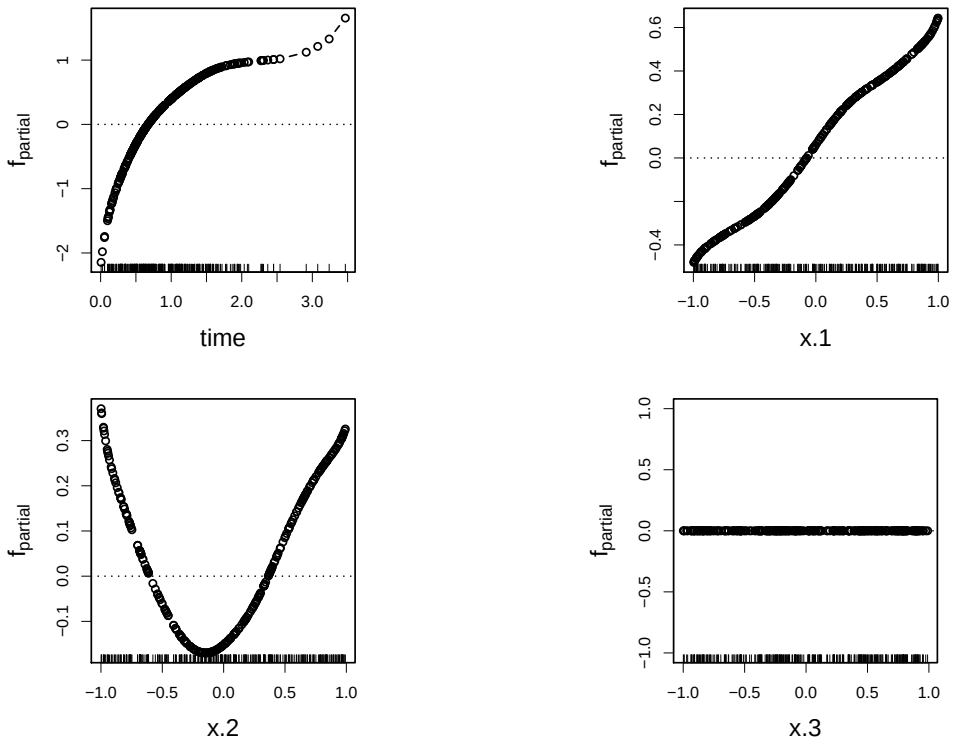


Figure B.5.: **Toy Example:** Estimated effects

# C. Figures

This chapter contains additional graphics of the simulations of Chapter 6. For a detailed explanation and interpretation see Sections 6.1.5 and 6.1.6.

## C.1. Simulation Results 2: Estimated Effects

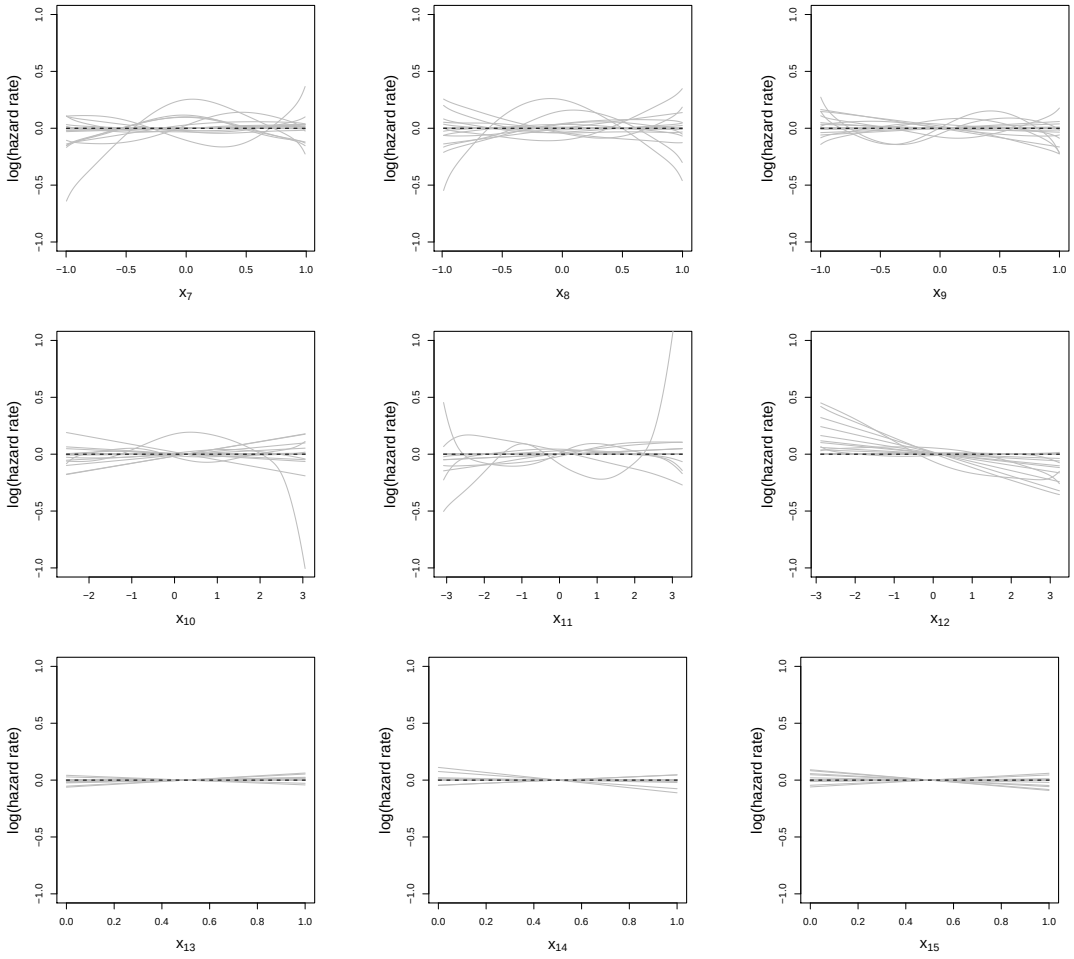


Figure C.1.: **Model Choice: Simulation Scheme 1** – Estimation of effects for non-effective covariates from 20 models (grey lines) and real effect (dashed lines). Effect estimates and true effect are centered.

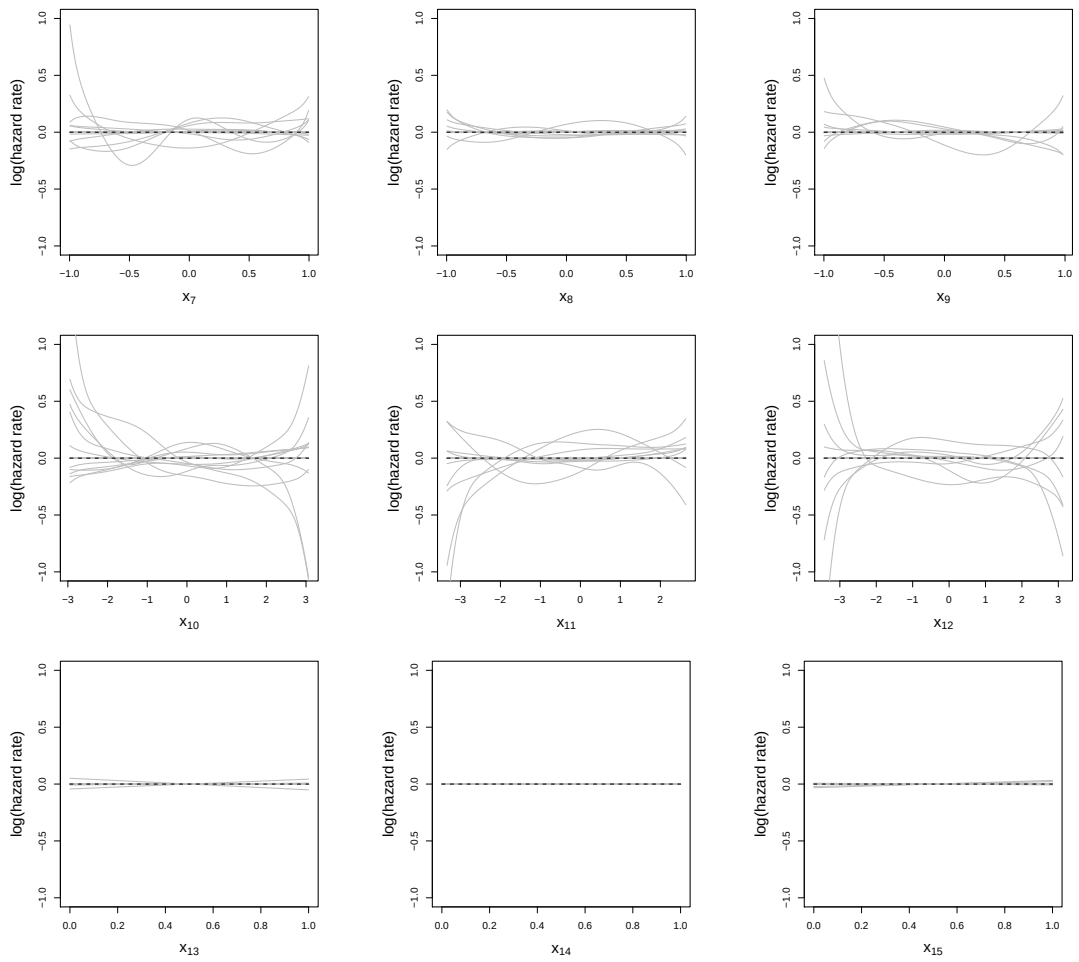


Figure C.2.: **Variable Selection: Simulation Scheme 1** – Estimation of covariate effects for non-effective covariates from 20 models (grey lines) and real effect (dashed lines). Effect estimates and true effect are centered.

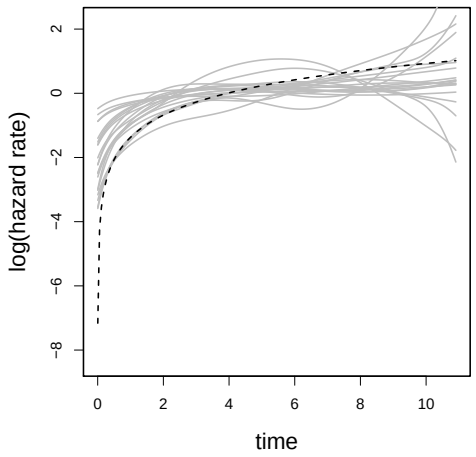


Figure C.3.: **Variable Selection: Simulation Scheme 2** – Estimation of baseline hazard from 20 models (grey lines) and real effect (dashed line). Effect estimates and true effect are centered.

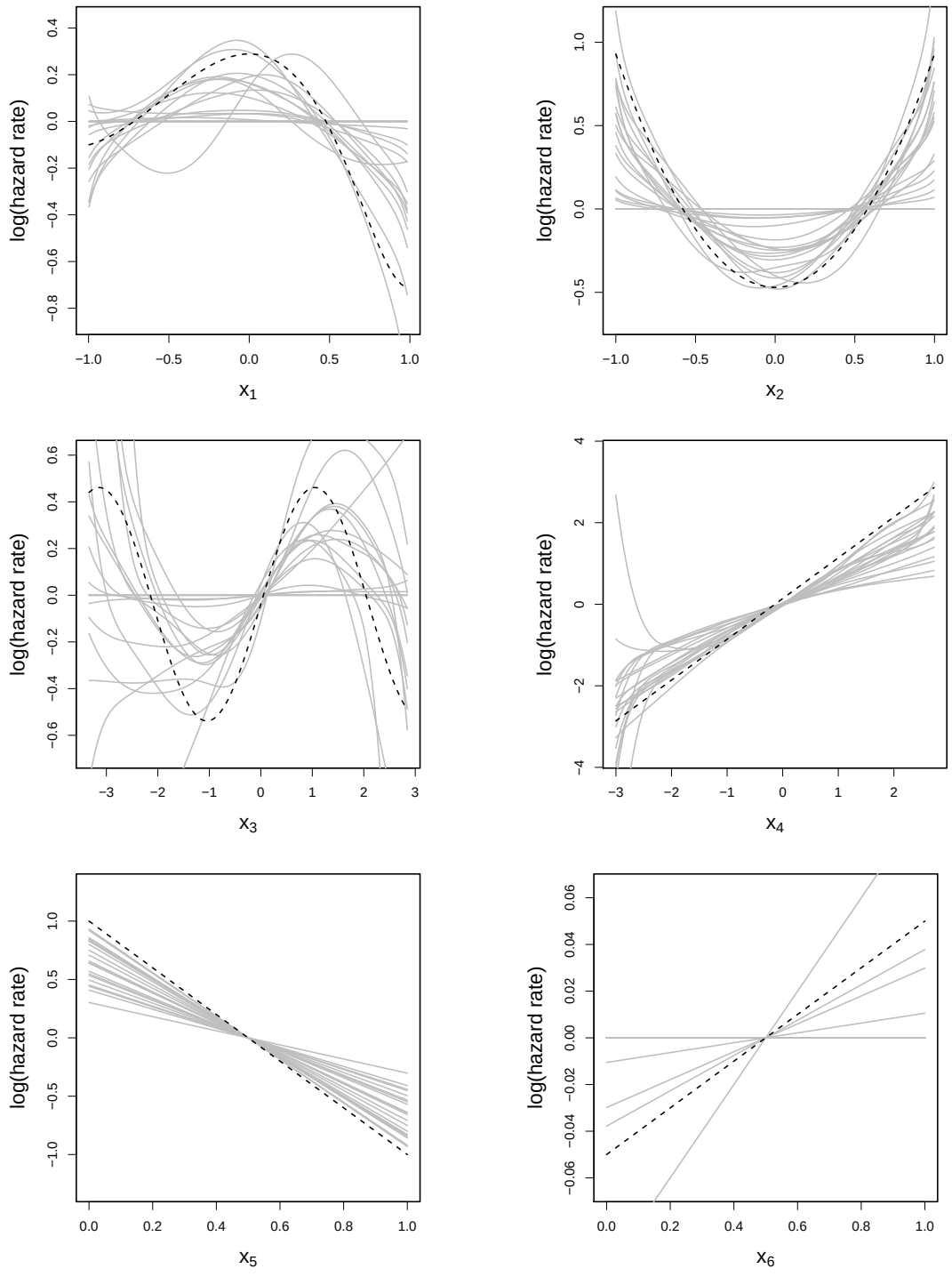


Figure C.4.: **Variable Selection: Simulation Scheme 2** – Estimation of covariate effects from 20 models (grey lines) and real effect (dashed lines). Effect estimates and true effect are centered. For $x_3$ and $x_4$ centering is only based on the "stable" areas $\in [-1, 1]$.

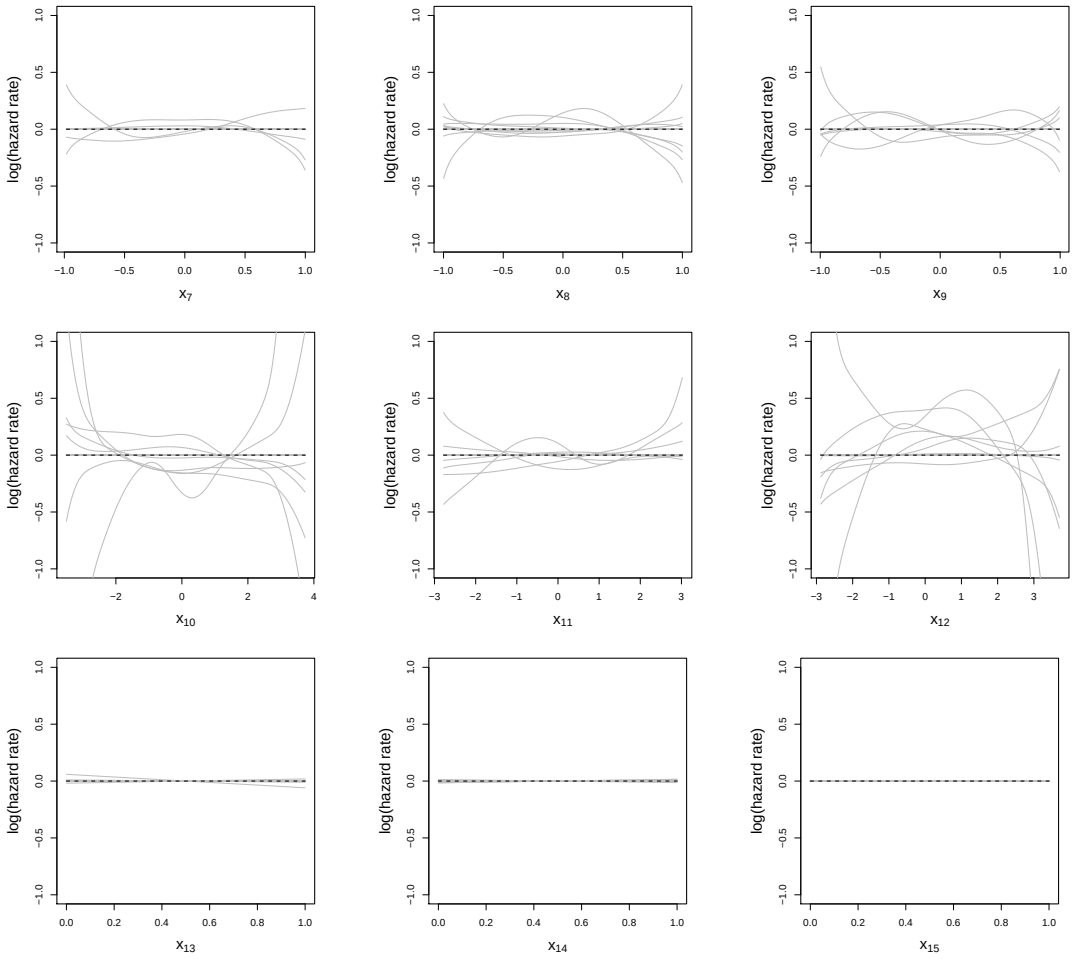


Figure C.5.: **Variable Selection: Simulation Scheme 2** – Estimation of covariate effects for non-effective covariates from 20 models (grey lines) and real effect (dashed lines). Effect estimates and true effect are centered.

## C.2. Simulation Results 3: Degrees of Freedom

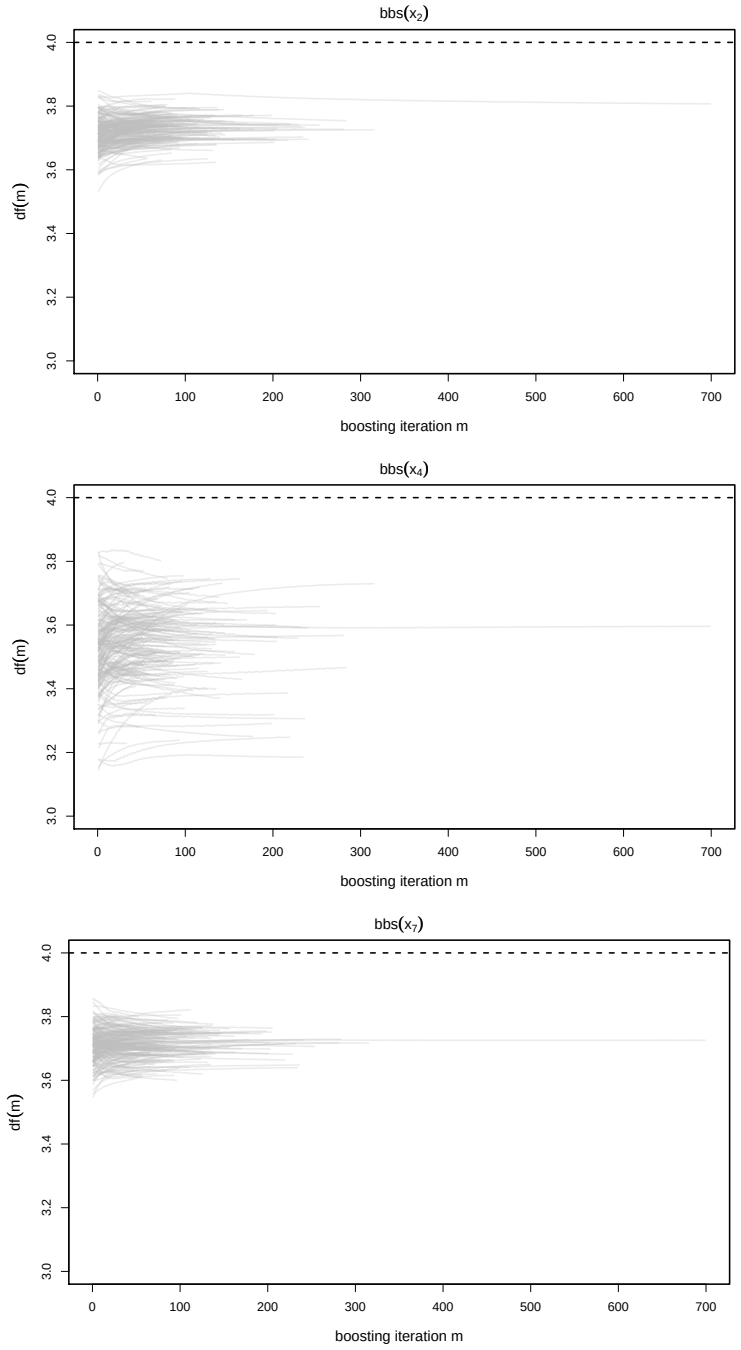


Figure C.6.: **Variable Selection: Simulation Scheme 1** ***(Part 1)*** – Estimated degrees of freedom traced over the boosting steps (in 200 replicates) for all flexible base-learners and initially specified degrees of freedom (dashed line). Plots for $x_1$ and $x_3$ can be found in Sec. 6.

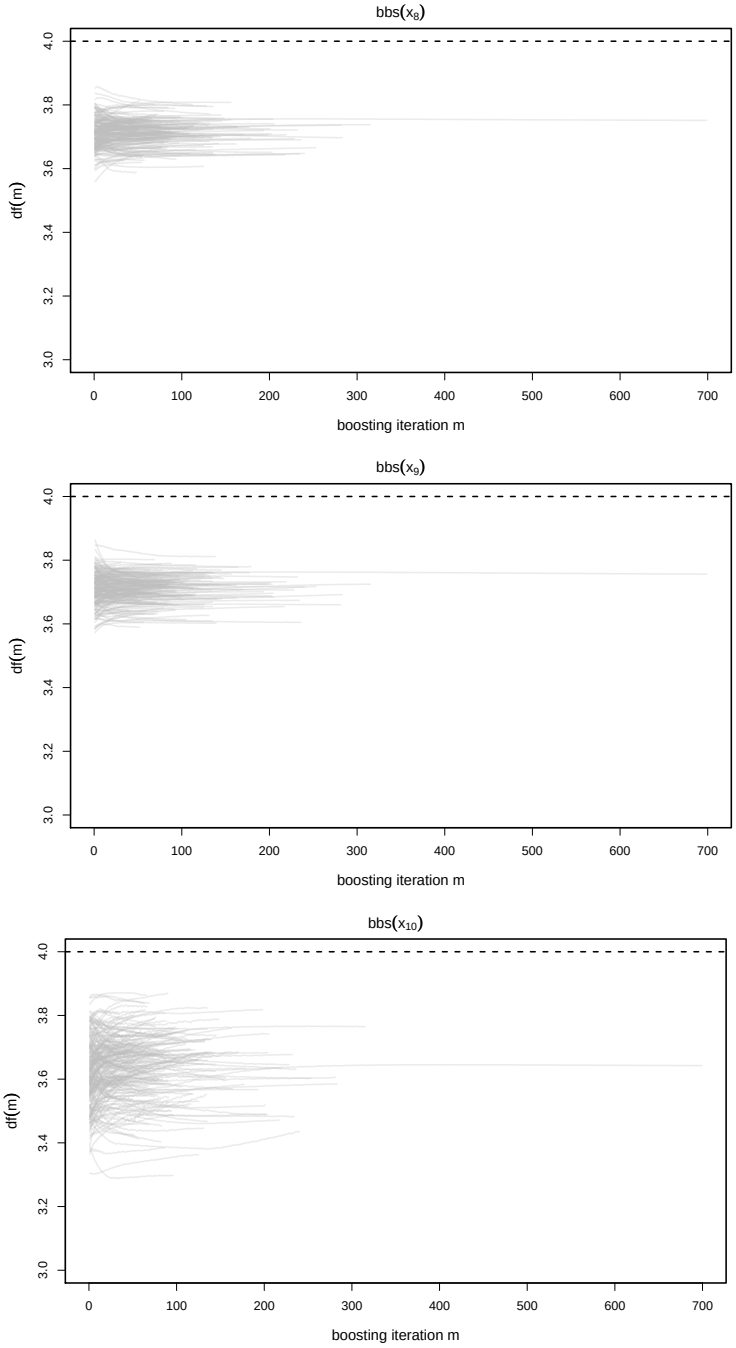


Figure C.7.: **Variable Selection: Simulation Scheme 1** ***(Part 2)*** – Estimated degrees of freedom traced over the boosting steps (in 200 replicates) for all flexible base-learners and initially specified degrees of freedom (dashed line).

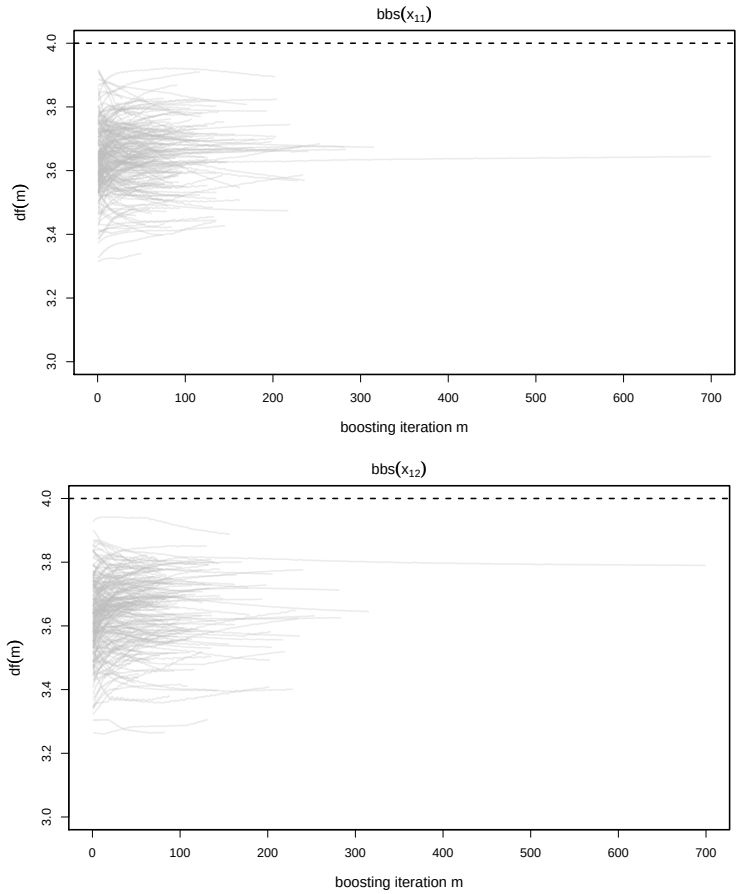


Figure C.8.: **Variable Selection: Simulation Scheme 1** ***(Part 3)*** – Estimated degrees of freedom traced over the boosting steps (in 200 replicates) for all flexible base-learners and initially specified degrees of freedom (dashed line).

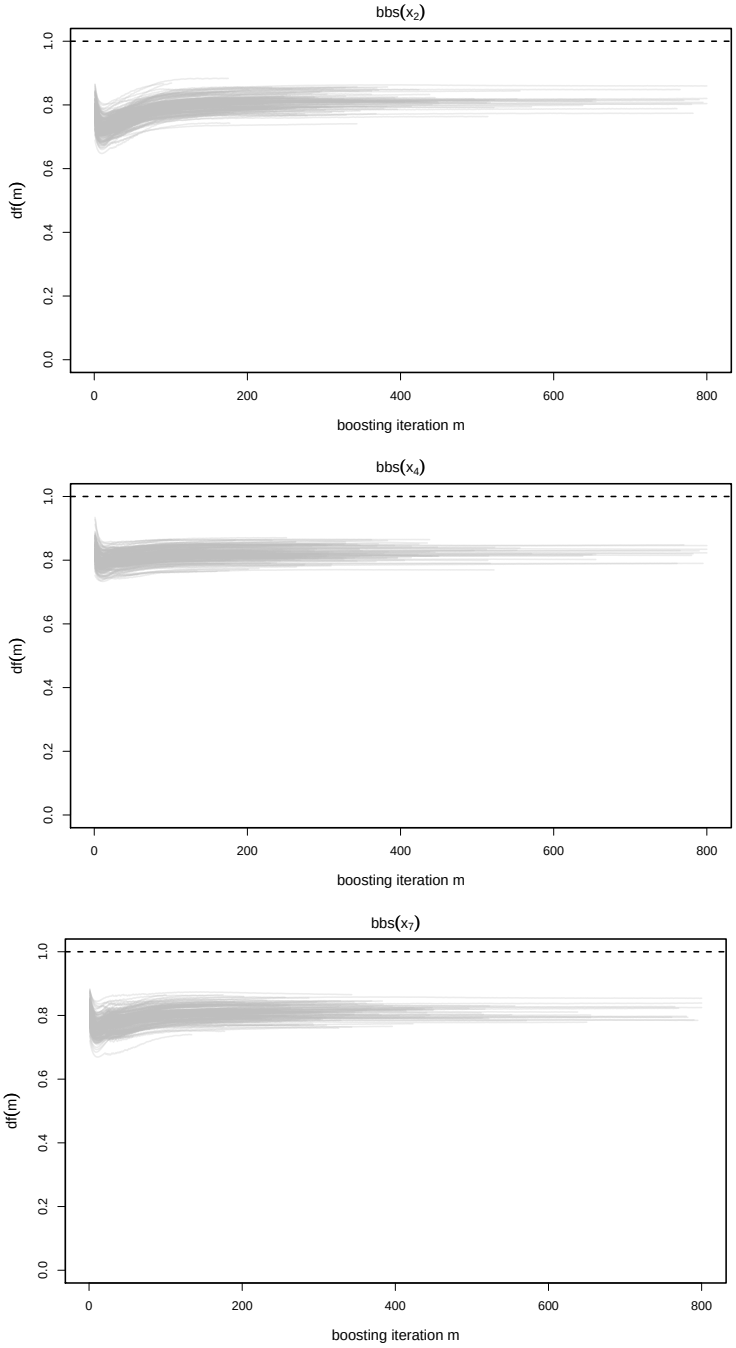


Figure C.9.: **Model Choice: Simulation Scheme 1** ***(Part 1)*** – Estimated degrees of freedom traced over the boosting steps (in 200 replicates) for all flexible base-learners and initially specified degrees of freedom (dashed line). Plots for $x_1$ and $x_3$ can be found in Sec. 6.

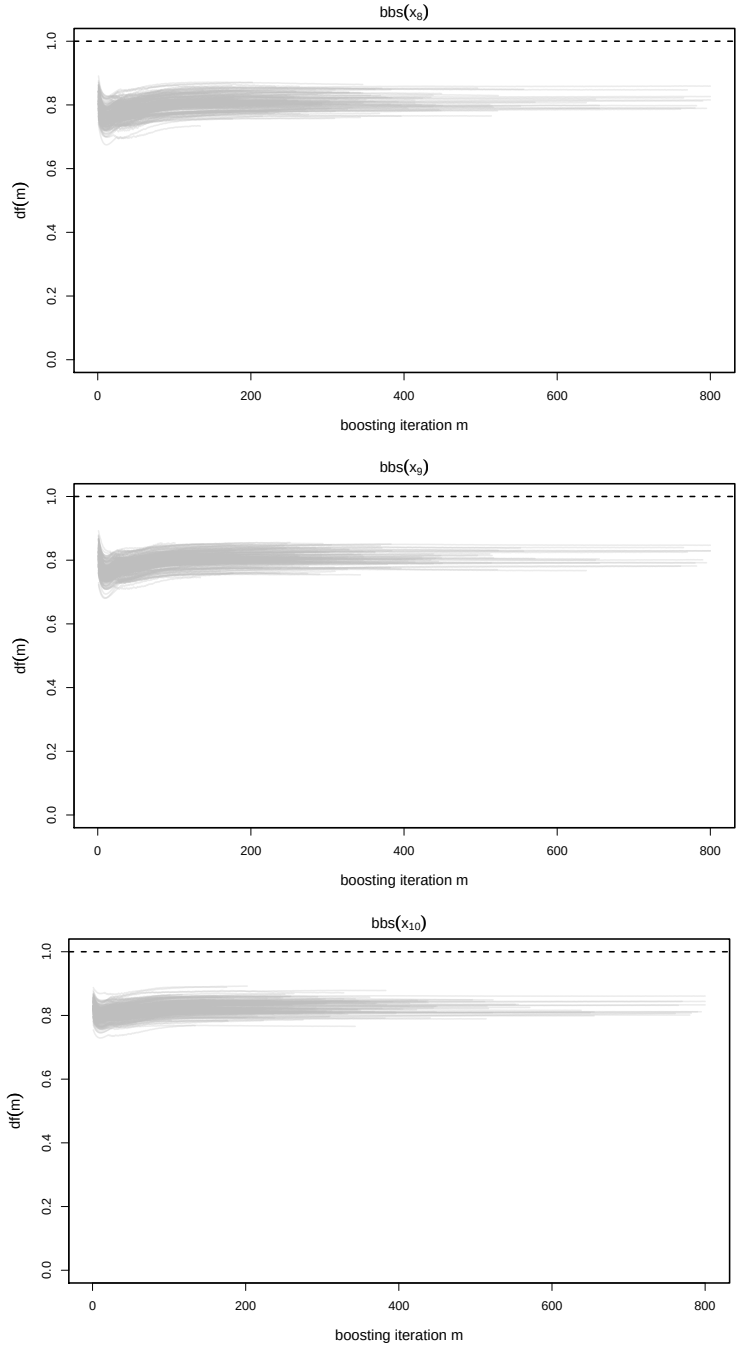


Figure C.10.: **Model Choice: Simulation Scheme 1** ***(Part 2)*** – Estimated degrees of freedom traced over the boosting steps (in 200 replicates) for all flexible base-learners and initially specified degrees of freedom (dashed line).

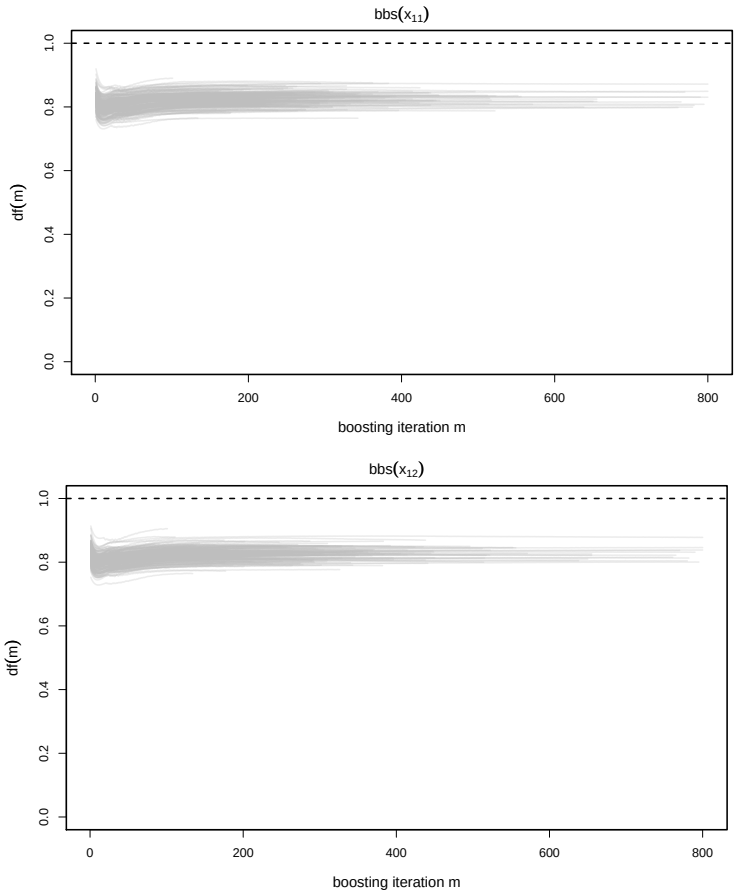


Figure C.11.: **Model Choice: Simulation Scheme 1** ***(Part 3)*** – Estimated degrees of freedom traced over the boosting steps (in 200 replicates) for all flexible base-learners and initially specified degrees of freedom (dashed line).

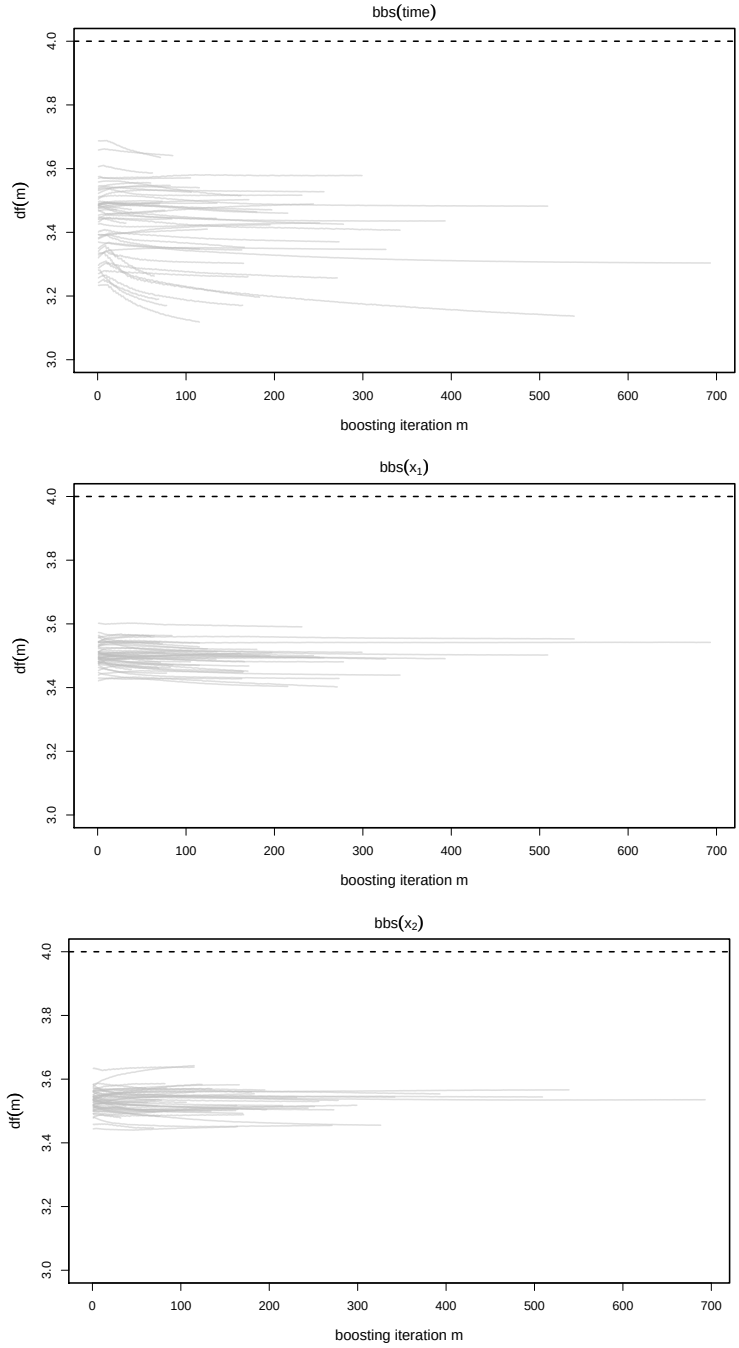


Figure C.12.: **Variable Selection: Simulation Scheme 2** ***(Part 1)*** – Estimated degrees of freedom traced over the boosting steps (in 200 replicates) for all flexible base-learners and initially specified degrees of freedom (dashed line).

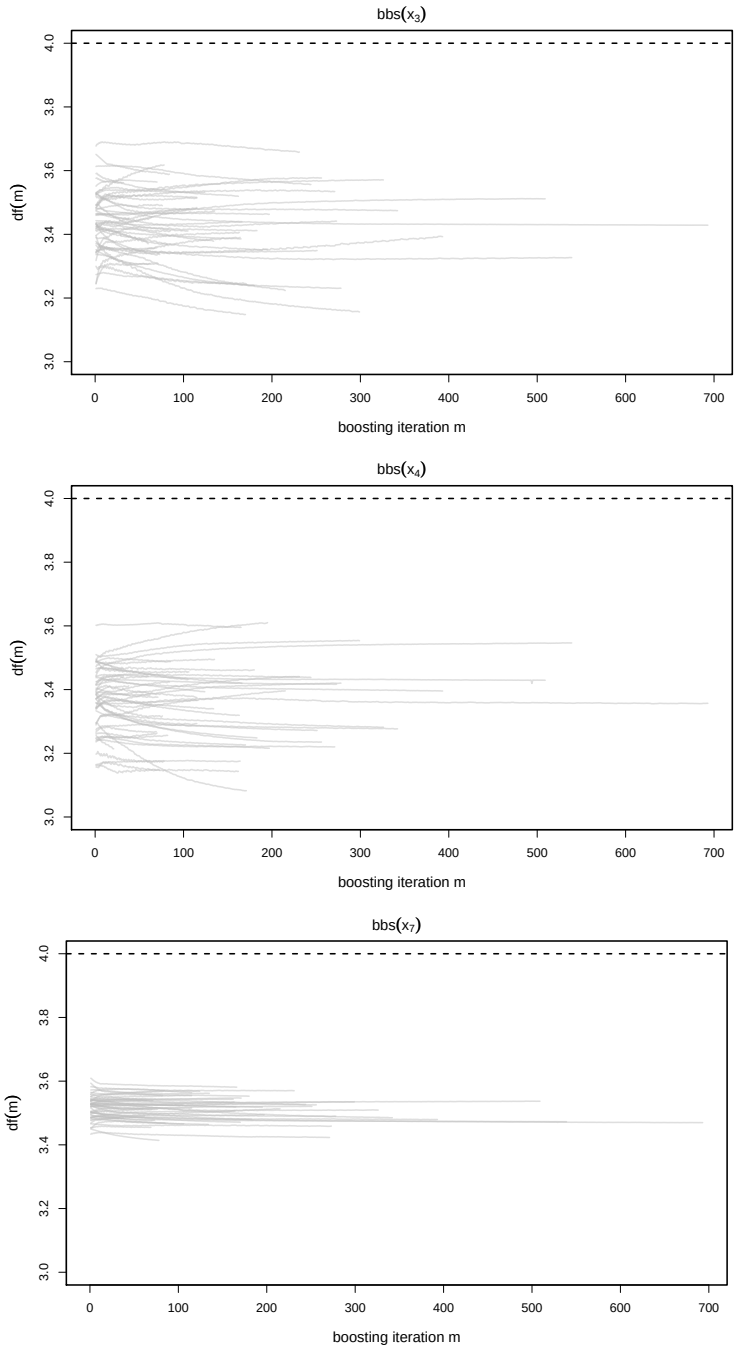


Figure C.13.: **Variable Selection: Simulation Scheme 2** ***(Part 2)*** – Estimated degrees of freedom traced over the boosting steps (in 200 replicates) for all flexible base-learners and initially specified degrees of freedom (dashed line).

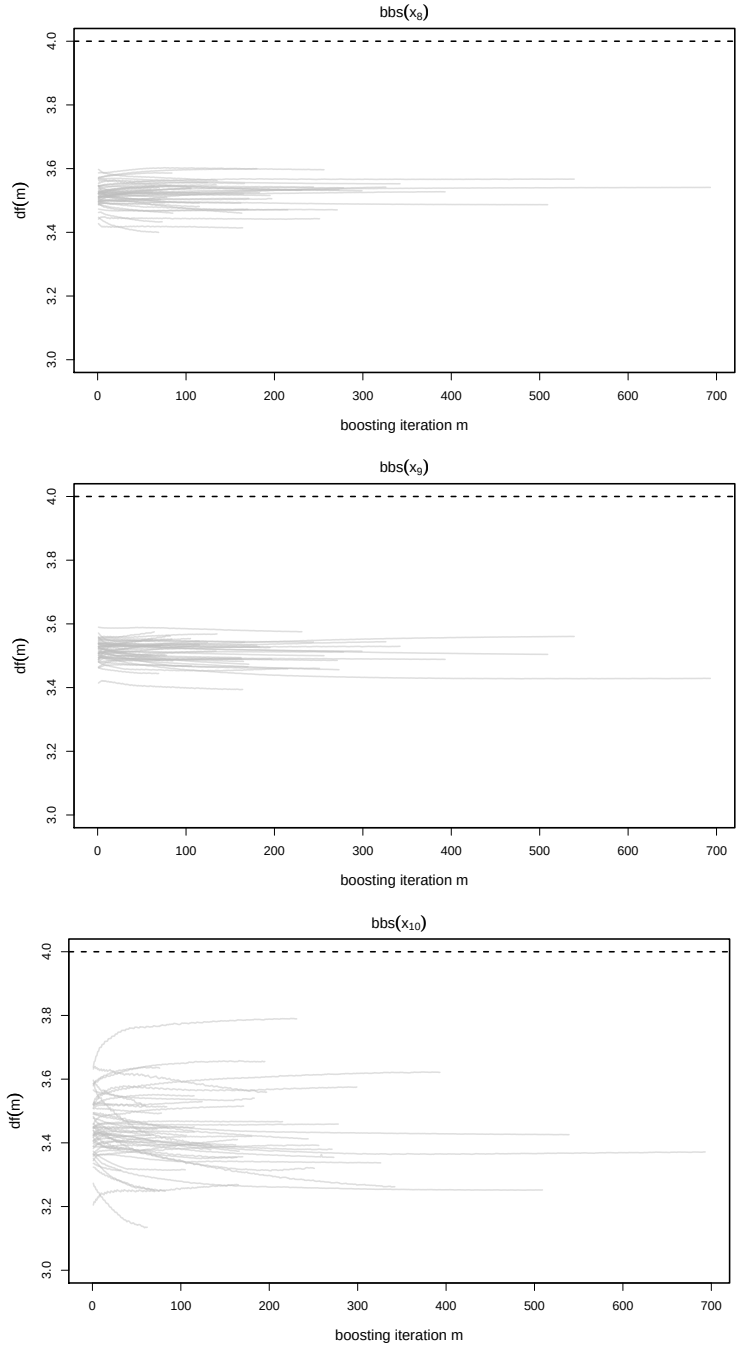


Figure C.14.: **Variable Selection: Simulation Scheme 2** ***(Part 3)*** – Estimated degrees of freedom traced over the boosting steps (in 200 replicates) for all flexible base-learners and initially specified degrees of freedom (dashed line).

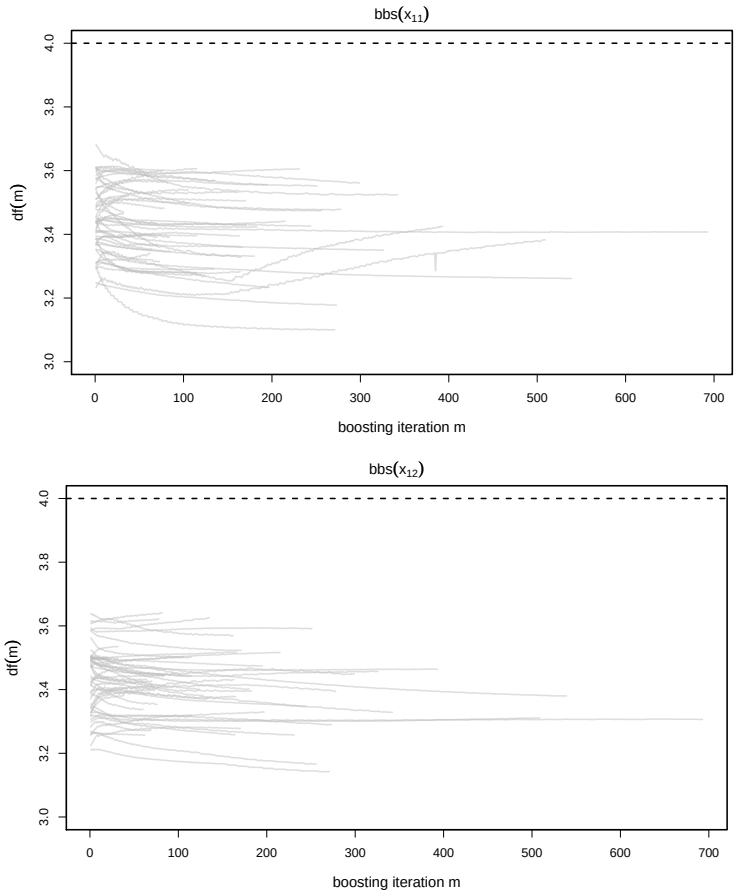


Figure C.15.: **Variable Selection: Simulation Scheme 2** ***(Part 4)*** – Estimated degrees of freedom traced over the boosting steps (in 200 replicates) for all flexible base-learners and initially specified degrees of freedom (dashed line).

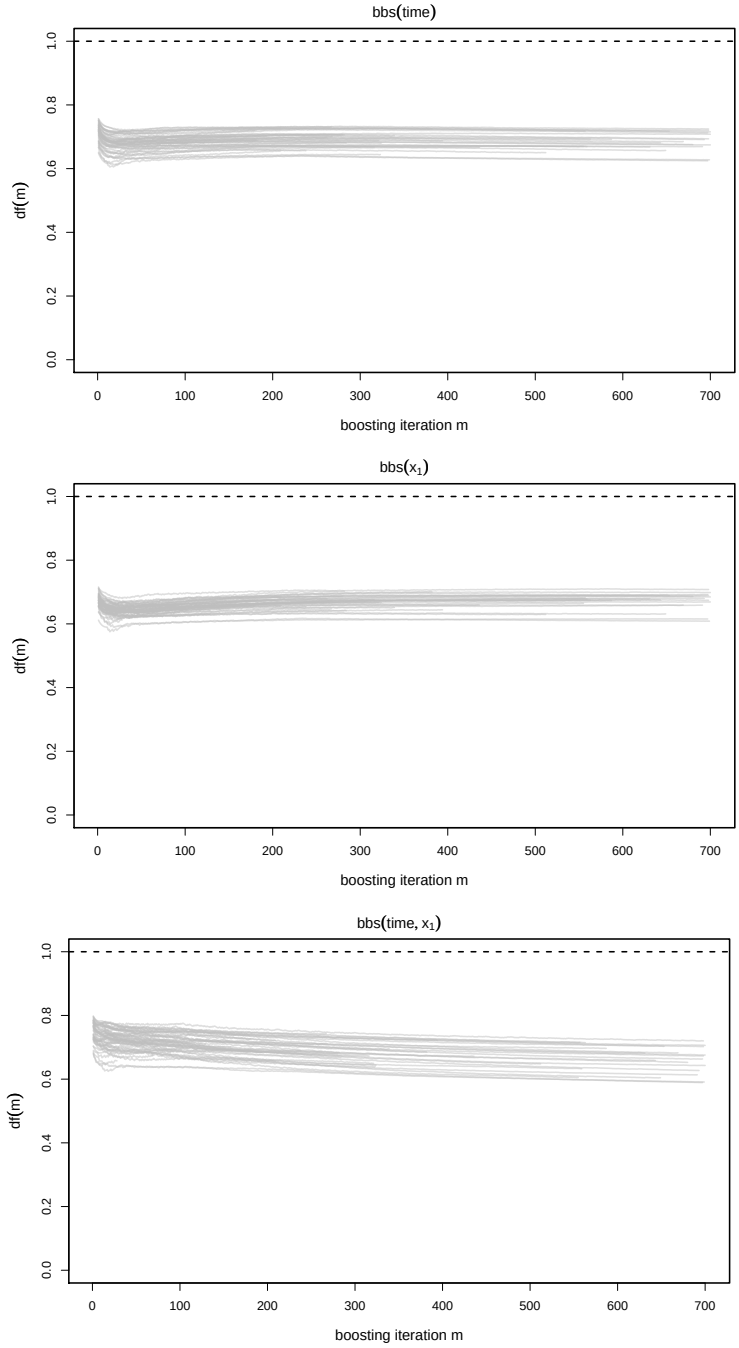


Figure C.16.: **Model Choice: Simulation Scheme 2** ***(Part 1)*** – Estimated degrees of freedom traced over the boosting steps (in 200 replicates) for all flexible base-learners and initially specified degrees of freedom (dashed line).

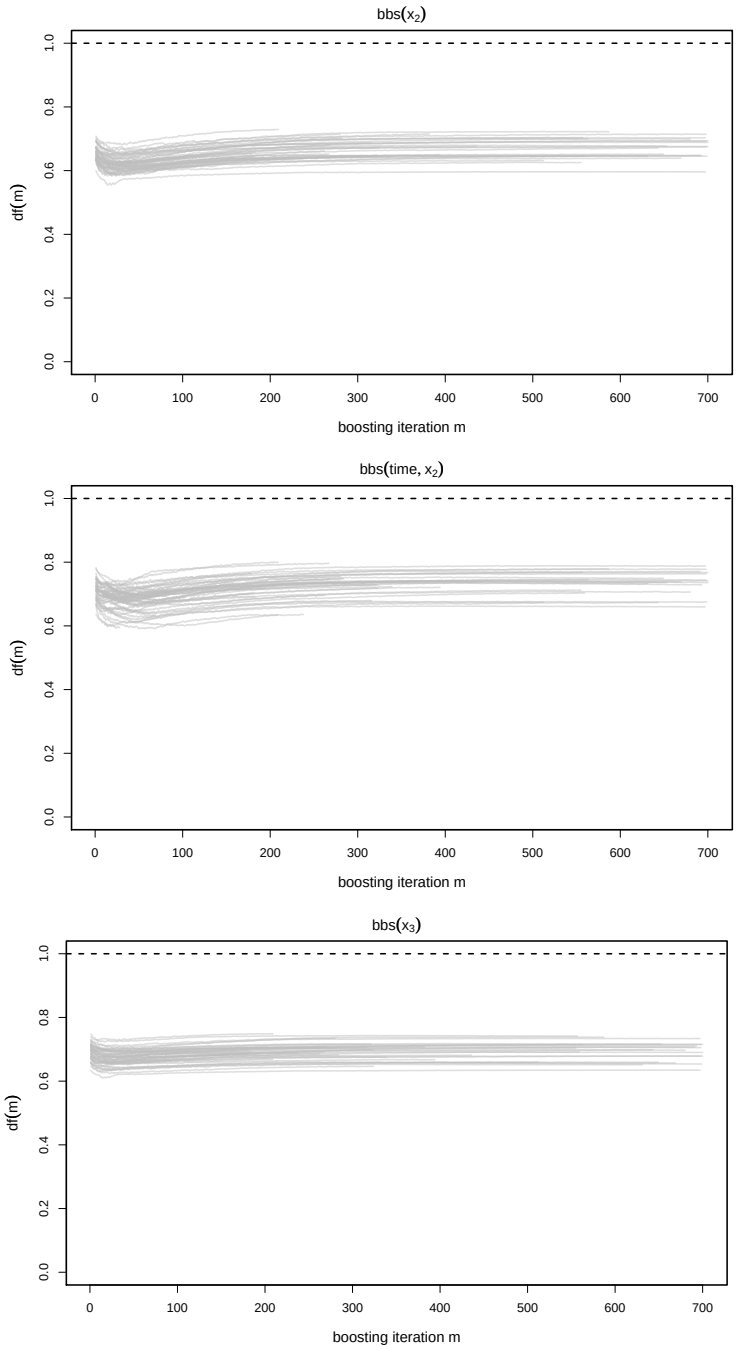


Figure C.17.: **Model Choice: Simulation Scheme 2** ***(Part 2)*** – Estimated degrees of freedom traced over the boosting steps (in 200 replicates) for all flexible base-learners and initially specified degrees of freedom (dashed line).

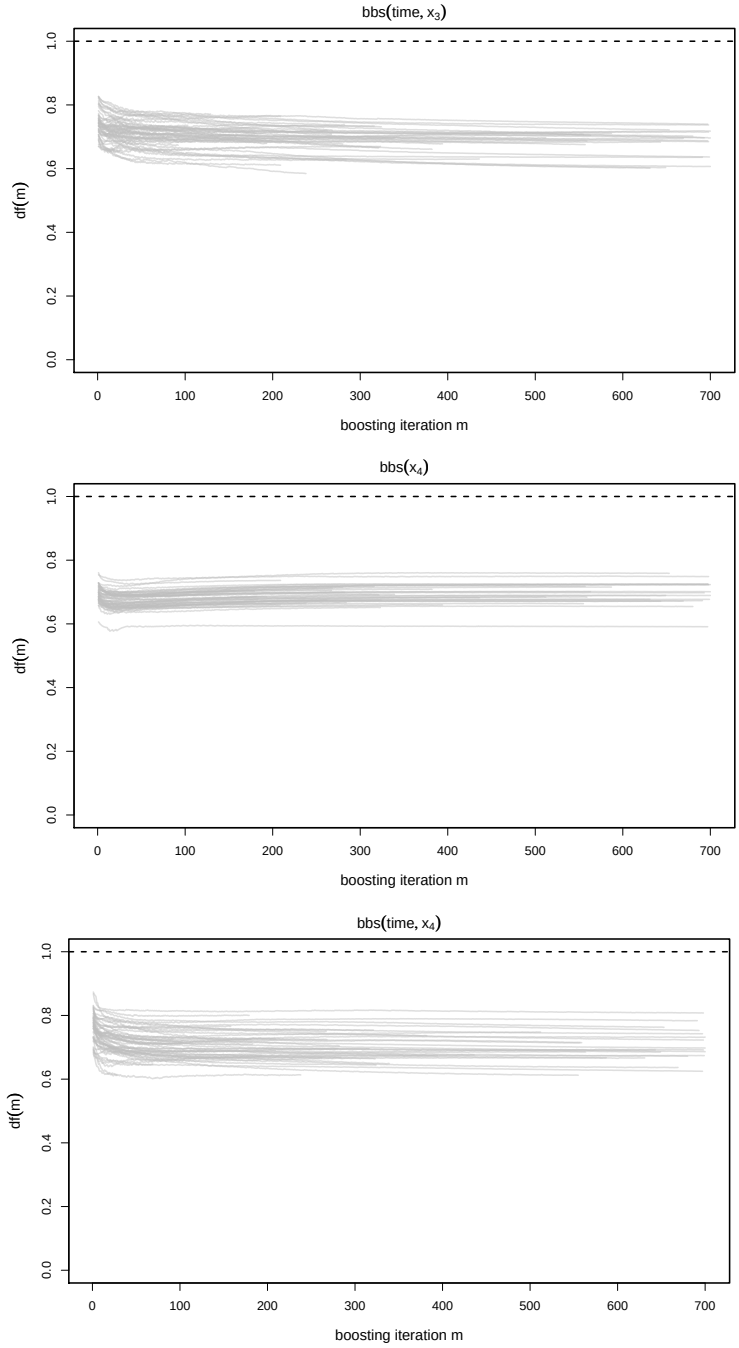


Figure C.18.: **Model Choice: Simulation Scheme 2** ***(Part 3)*** – Estimated degrees of freedom traced over the boosting steps (in 200 replicates) for all flexible base-learners and initially specified degrees of freedom (dashed line).

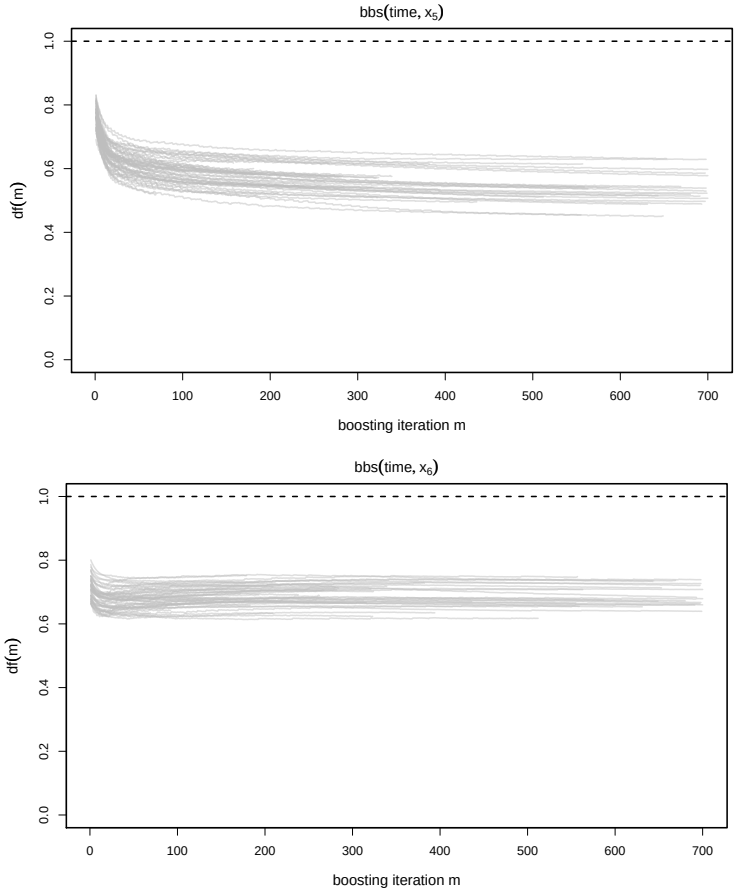


Figure C.19.: **Model Choice: Simulation Scheme 2** ***(Part 4)*** – Estimated degrees of freedom traced over the boosting steps (in 200 replicates) for all flexible base-learners and initially specified degrees of freedom (dashed line).

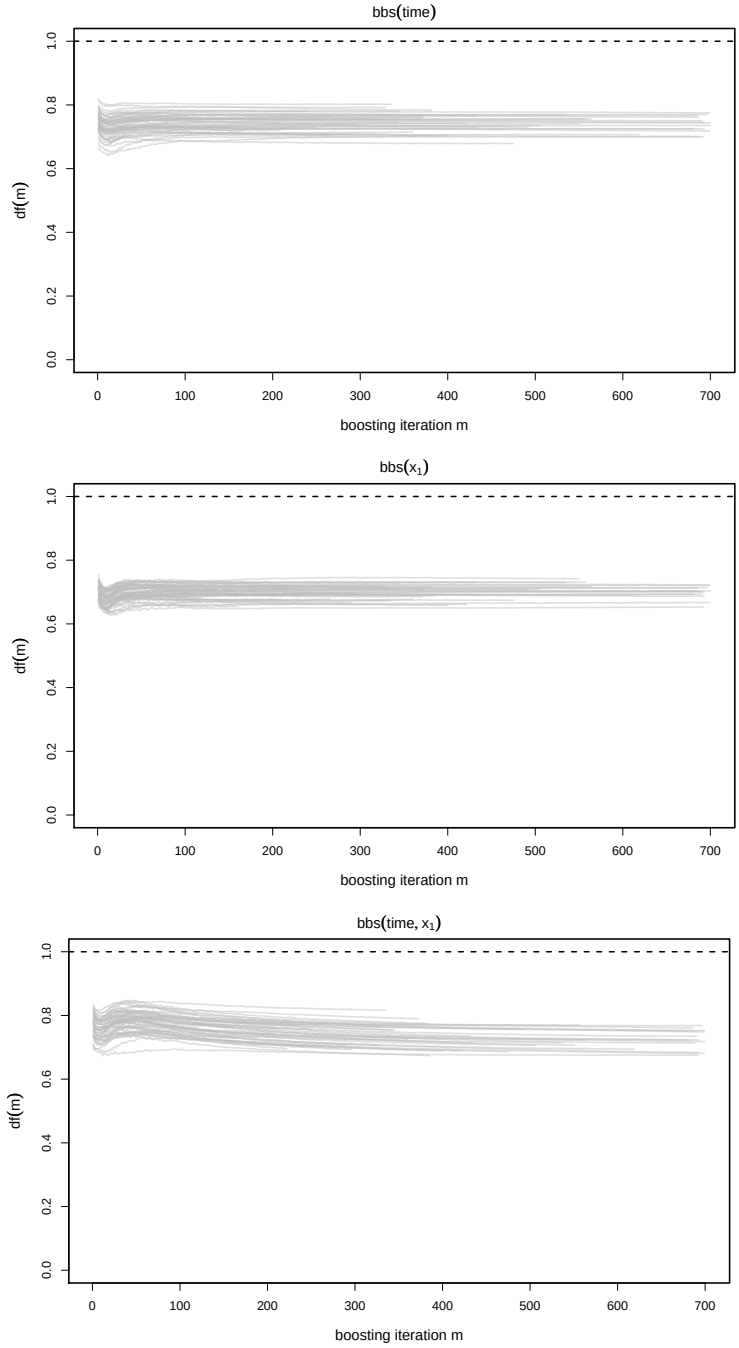


Figure C.20.: **Model Choice: Simulation Scheme 3** ***(Part 1)*** – Estimated degrees of freedom traced over the boosting steps (in 200 replicates) for all flexible base-learners and initially specified degrees of freedom (dashed line).

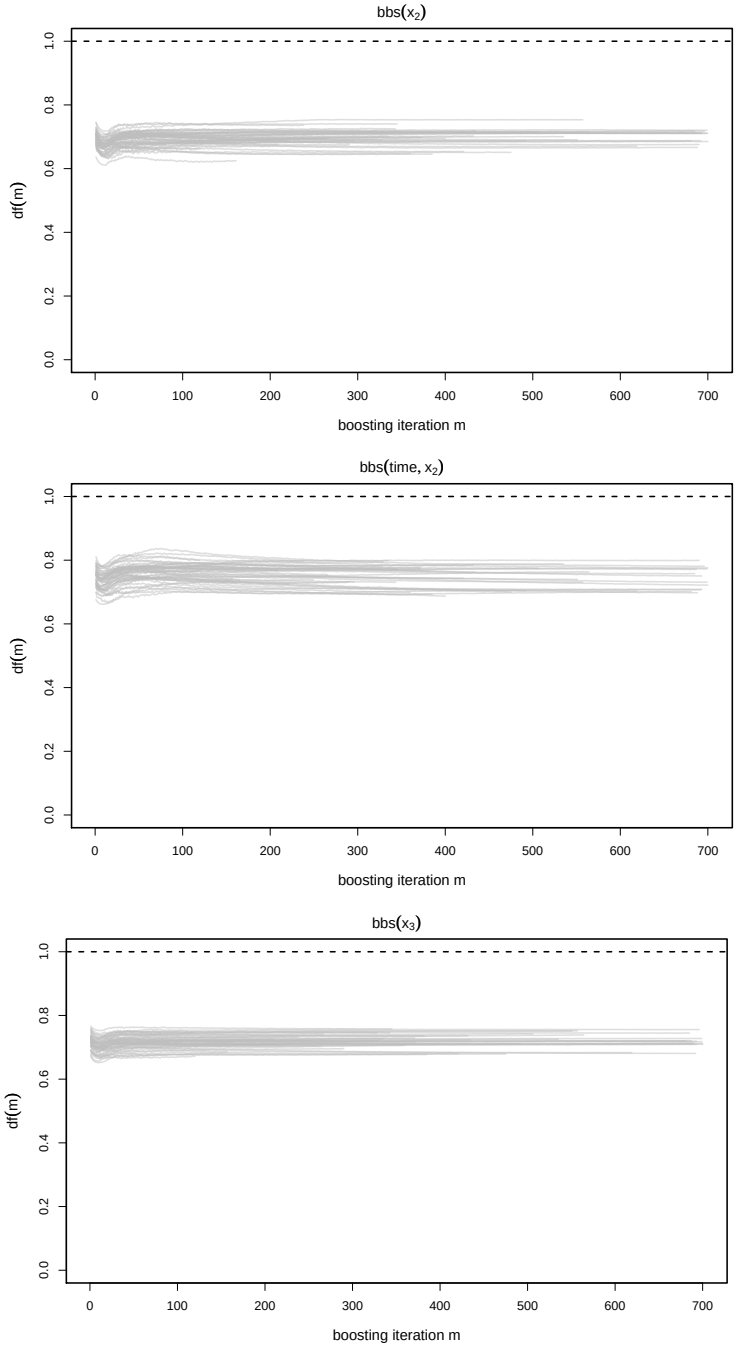


Figure C.21.: **Model Choice: Simulation Scheme 3** ***(Part 2)*** – Estimated degrees of freedom traced over the boosting steps (in 200 replicates) for all flexible base-learners and initially specified degrees of freedom (dashed line).

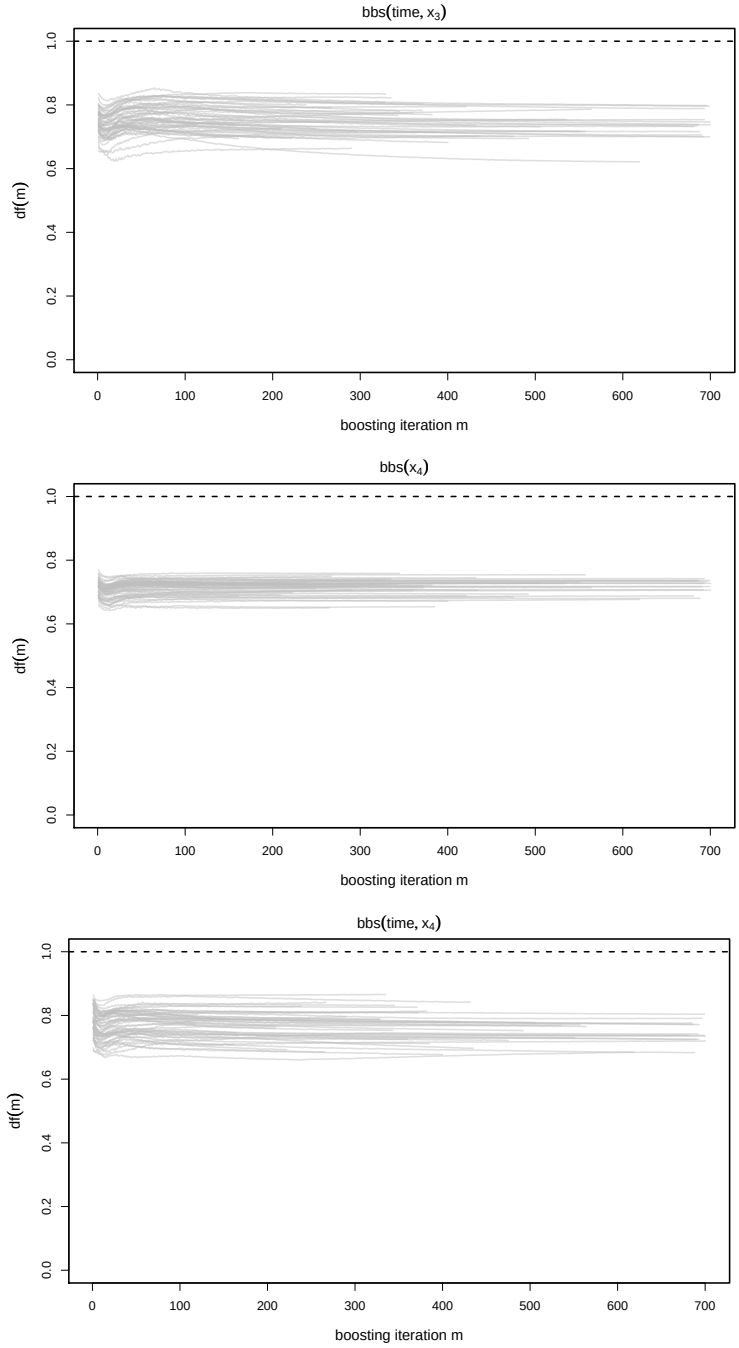


Figure C.22.: **Model Choice: Simulation Scheme 3** ***(Part 3)*** – Estimated degrees of freedom traced over the boosting steps (in 200 replicates) for all flexible base-learners and initially specified degrees of freedom (dashed line).

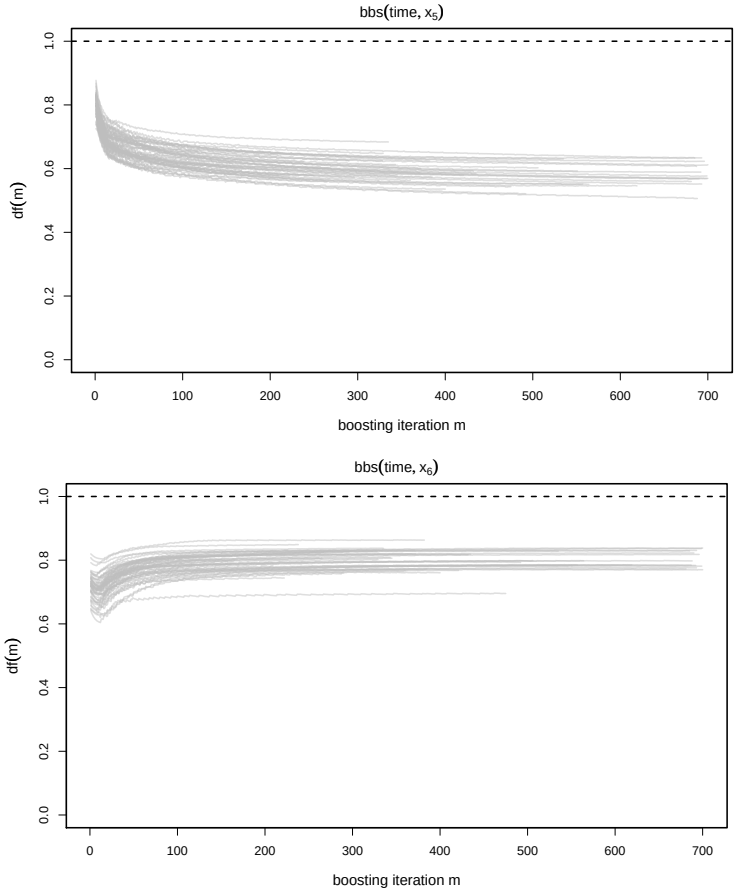


Figure C.23.: **Model Choice: Simulation Scheme 3** ***(Part 4)*** – Estimated degrees of freedom traced over the boosting steps (in 200 replicates) for all flexible base-learners and initially specified degrees of freedom (dashed line).

# Bibliography


Barlow, W. E. & Prentice, R. L. (1988). Residuals for relative risk regression, *Biometrika* **75**: 65–74.

Bender, R., Augustin, T. & Blettner, M. (2005). Generating survival times to simulate Cox proportional hazards models, *Statistics in Medicine* **24**: 1713–1723.

Bernoulli, D. & Blower, S. (2004). Review of: An attempt at a new analysis of the mortality caused by smallpox and of the advantages of inoculation to prevent it, *Reviews in Medical Virology* **14**: 275–288.

Binder, H. & Schumacher, M. (2008). Allowing for mandatory covariates in boosting estimation of sparse high-dimensional survival models, *BMC Bioinformatics* **9**: 14.

Breiman, L. (1998). Arcing classifiers (with discussion), *The Annals of Statistics* **26**: 801–849.

Breiman, L. (1999). Prediction games & arcing algorithms, *Neural Computation* **11**: 1493–1517.

Breslow, N. (1974). Covariance analysis of censored survival data, *Biometrics* **30**: 89–99.

Brier, G. W. (1950). Verification of forecasts expressed in terms of probability, *Monthly Weather Review* **78**: 1–3.

Bühlmann, P. (2006). Boosting for high-dimensional linear models, *The Annals of Statistics* **34**: 559–583.

Bühlmann, P. & Hothorn, T. (2007). Boosting algorithms: Regularization, prediction and model fitting, *Statistical Science* **22**: 477–505.

Bühlmann, P. & Yu, B. (2003). Boosting with the $L_2$ Loses: Regression and classification, *Journal of the American Statistical Association* **98**: 324–339.

Buja, A., Hastie, T. & Tibshirani, R. (1989). Linear smoothers and additive models (with discussion), *The Annals of Statistics* **17**: 453–555.

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

de Boor, C. (1978). *A Practical Guide to Splines*, Springer, New York.

Devroye, L. (1986). *Non-Uniform Random Variate Generation*, Springer, New York.

Dierckx, P. (1993). *Curve and Surface Fitting with Splines*, Clarendon Press, Oxford.

Eilers, P. H. C. & Marx, B. D. (1996). Flexible smoothing with B-splines and penalties, *Statistical Science* **11**: 89–121.

Fahrmeir, L., Kneib, T. & Lang, S. (2004). Penalized structured additive regression: A Bayesian perspective, *Statistica Sinica* **14**: 731–761.

Fahrmeir, L. & Tutz, G. (2001). *Multivariate Statistical Modelling Based on Generalized Linear Models. Second Edition*, Springer, New York.

Freund, Y. & Schapire, R. E. (1996). Experiments with a new boosting algorithm, *Machine Learning: Proc. Thirteenth International Conference*, Morgan Kaufmann Publishers Inc., San Francisco, CA, pp. 148–156.

Freund, Y. & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting, *Journal of Computer and System Sciences* **55**: 119–139.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine, *The Annals of Statistics* **29**: 1189–1232.

Friedman, J., Hastie, T. & Tibshirani, R. (2000). Additive logistic regression: A statistical view of boosting (with discussion), *The Annals of Statistics* **28**: 337–407.

Friedman, J. & Silverman, B. (1989). Flexible parsimonious smoothing and additive modeling (with discussion), *Technometrics* **31**: 3–39.

Gerds, T. & Schumacher, M. (2006). Consistent estimation of the expected Brier score in general survival models with right-censored event times, *Biometrical Journal* **48**: 1029–1040.

Graf, E., Schmoor, C., Sauerbrei, W. & Schumacher, M. (1999). Assessment and comparison of prognostic classification schemes for survival data, *Statistics in Medicine* **18**: 2529–2545.

Gray, R. J. (1992). Flexible methods for analyzing survival data using splines, with application to breast cancer prognosis, *Journal of the American Statistical Association* **87**: 942–951.

Hansen, M. & Yu, B. (2001). Model selection and minimum description length principle, *Journal of the American Statistical Association* **96**: 746–774.

Hastie, T. & Tibshirani, R. (1990a). Exploring the nature of covariate effects in the proportional hazards model, *Biometrics* **46**: 1005–1016.

Hastie, T. & Tibshirani, R. (1990b). *Generalized Additive Models*, Chapman & Hall / CRC , London.

Hastie, T. & Tibshirani, R. (1993). Varying-coefficient models, *Journal of the Royal Statistical Society. Series B* **55**: 757–796.

Hofner, B., Kneib, T., Hartl, W. & Küchenhoff, H. (2008). Model choice in Cox-type additive hazard regression models with time-varying effects. *Preprint*.

Hothorn, T., Bühlmann, P., Dudoit, S., Molinaro, A. & van der Laan, M. J. (2006). Survival ensembles, *Biostatistics* **7**: 355–373.

Hothorn, T., Bühlmann, P., Kneib, T. & Schmid, M. (2007). *mboost: Model-Based Boosting*. R package version 1.0-1.
URL: `http://R-forge.R-project.org`

Hurvich, C., Simonoff, J. & Tsai, C. (1998). Smoothing parameter selection in nonparametric regression using an improved Akaike information criterion, *Journal of the Royal Statistical Society, Series B* **60**: 271–293.

Klein, J. P. & Moeschberger, M. L. (2003). *Survival Analysis: Techniques for Censored and Truncated Data. Second Edition*, Springer, New York.

Kneib, T. & Fahrmeir, L. (2007). A mixed model approach for geoadditive hazard regression, *Scandinavian Journal of Statistics* **34**: 207–228.

Kneib, T., Hothorn, T. & Tutz, G. (2007). Variable selection and model choice in geoadditive regression. *Submitted to Biometrics*.

Kooperberg, C. & Stone, C. (1992). Logspline density estimation for censored data, *Journal of Computational and Graphical Statistics* **1**: 301–328.

Mood, A., Graybill, F. & D.C., B. (1974). *Introduction to the Theory of Statistics*, McGraw-Hill, New York.

Moubarak, P., Zilker, S., Wolf, H., Hofner, B., Kneib, T., Küchenhoff, H., Jauch, K.-W. & Hartl, W. H. (2008). Activity-guided antithrombin III therapy in severe surgical sepsis: Efficacy and safety according to a retrospective data analysis, *Shock* . Accepted.

Press, W. H., Teukolsky, S. A., Vetterling, W. T. & Flannery, B. (1992). *Numerical Recipes in C: The Art of Scientific Computing. Second Edition*, Cambridge University Press.

R Development Core Team (2007). *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0.
URL: `http://www.R-project.org`

Ridgeway, G. (1999). The state of boosting, *Computing Science and Statistics* **31**: 172–181.

Royston, P. & Altman, D. G. (1994). Regression using fractional polynomials of continuous covariates: Parsimonious parametric modelling, *Applied Statistics* **43**: 429–453.

Sauerbrei, W., Royston, P. & Look, M. (2007). A new proposal for multivariable modelling of time-varying effects in survival data based on fractional polynomial time-transformation, *Biometrical Journal* **49**: 453–473.

Schapire, R. E. (1990). The strength of weak learnability, *Machine Learning* **5**: 197–227.

Schmid, M. & Hothorn, T. (2007). Boosting additive models using component-wise P-splines. *Preprint*.

Schmid, M. & Hothorn, T. (2008). Flexible boosting of accelerated failure time models, *BMC Bioinformatics* . To appear.

Schoenfeld, D. (1982). Partial residuals for the proportional hazards regression model, *Biometrika* **69**: 239–241.

Silverman, B. W. (1985). Some aspects of the spline smoothing approach to non-parametric regression curve fitting (with discussion), *Journal of the Royal Statistical Society, Series B: Methodological* **47**: 1–52.

Therneau, T. M. & Grambsch, P. M. (2000). *Modeling survival data: Extending the Cox model*, Springer, New York.

Therneau, T. M., Grambsch, P. M. & Fleming, T. R. (1990). Martingale based residuals for survival models, *Biometrika* **77**: 147–160.

Tutz, G. & Binder, H. (2006). Generalized additive modelling with implicit variable selection by likelihood-based boosting, *Biometrics* **62**: 961–971.

van der Vaart, A. (1998). *Asymptotic Statistics*, Cambridge University Press.

# List of Figures

# List of Tables

**Erklärung zur Urheberschaft**
Hiermit versichere ich, dass ich die vorliegende Diplomarbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe.

München, den 07.07.2008

(Benjamin Hofner)