

# Bayesian analyses of time-to-event data using the `rstanarm` R package

*Eren M. Elçi and Sam Brilleman*

## Abstract

Time-to-event data refers to the observed time from a defined origin (e.g. diagnosis of a disease) until a terminating event of interest (e.g. death). Such data emerges in a range of industries and scientific disciplines, although it is particularly common in medical and pharmaceutical research. In this talk we introduce the `stan_surv` survival modelling function that is being incorporated into the `rstanarm` R package. We use a clinical case study to demonstrate some of the features of `stan_surv` and its related post-estimation functions. We show how the software can be used as part a robust Bayesian workflow that includes prior and posterior predictive checks. In our prior and posterior predictive checks we consider standard quantities used widely in survival analysis (e.g. hazard functions, survival curves), as well as less widely discussed but equally useful quantities such as the restricted mean survival time (RMST). Date this notebook was compiled: 26 August 2019.

## 1 Introduction

Time-to-event analysis is concerned with an outcome variable that corresponds to the time from some defined baseline until an event of interest occurs. The methodology is used in a range of industries and scientific disciplines, although it is particularly common in medical and pharmaceutical research. For instance, a common scenario in medical research is analysis of the time from disease diagnosis until death. In different disciplines it is known by different names including survival analysis (medicine), duration analysis (economics), reliability analysis (engineering), and event history analysis (sociology). Throughout this notebook we refer to the methodology as survival analysis – partly because we introduce the concepts through a clinical case study and partly because the related terminology (e.g. survival models, survival probabilities, survival curves) then follows naturally.

In standard survival analyses one event time is measured for each individual. In practice however the end of the study may occur before the individual experiences the event (known as administrative censoring). Alternatively, there may be some other reason the event time cannot be observed exactly. If that is the case then the event time is *censored* (i.e. left, right or interval censored) and is only known to have occurred within the relevant censoring interval. These combined aspects of time and censoring make survival analysis methods relatively distinct from many other regression modelling approaches.

To date, much of the software developed for survival analysis has been based on maximum likelihood or partial likelihood estimation methods. This is in part due to the popularity of the Cox model which is based on a partial likelihood approach that does not require any significant computing resources. To our knowledge there are few general purpose Bayesian survival analysis packages for the R software. Perhaps one notable exception is the `spBayesSurv` R package (Zhou, Hanson, and Zhang 2018), which focuses on Bayesian spatial survival modelling but can also be used for non-spatial survival data.

The `rstanarm` R package (Goodrich et al. 2018) is a high-level interface to the Stan software (Carpenter et al. 2017). It has become one of the most widely used R packages for Bayesian applied regression modelling. In this notebook we introduce new functionality for modelling survival data using `rstanarm`. This functionality is built around the new `stan_surv` modelling function. We present a clinical case study that demonstrates several key features of `stan_surv` and its post-estimation functions. We show how the software can be used as part a robust Bayesian workflow that includes prior and posterior predictive checks. In our prior and posterior predictive checks we not only consider standard quantities used widely in survival analysis (e.g. hazard functions, survival curves), but also less widely discussed (but equally useful) quantities such as the restricted mean survival time (RMST).

## 2 Motivating example

In this case study we work with survival data from a *randomized clinical trial* run by NCOG (the Northern California Oncology Group) comparing two treatments for head and neck cancer: radiotherapy (Arm A) versus radiotherapy *and* chemotherapy (Arm B). The outcome variable used in these analyses is time to death.

The abstract from the corresponding publication (Fu et al. 1987) reads as follows:

Between 1978 and 1984, the Northern California Oncology Group (NCOG) conducted a randomized trial to study the efficacy of combined radiotherapy (RT) and chemotherapy (CT) for stage III or IV inoperable head and neck cancer. One hundred four patients were randomized to receive: (1) RT alone, or (2) RT plus CT. RT consisted of 7,000 cGy to the involved areas and 5,000 cGy to uninvolved neck at 180 cGy/fraction, five fractions/wk. CT consisted of bleomycin, 5 U intravenously (IV), twice weekly during RT, followed by bleomycin, 15 U IV, and methotrexate, 25 mg/m<sup>2</sup> IV weekly for 16 weeks after completion of RT. Fifty-one patients in the RT alone group and 45 in the combined treatment group were evaluable.

[...]

The relapse-free survival curves were significantly different ( $P = .041$ ), favoring the combined treatment. However, the survival curves were not significantly different ( $P = .16$ ). Patient compliance to maintenance CT was poor. Bleomycin significantly increased the acute radiation mucositis, although the difference in late normal tissue toxicity was not statistically significant. Thus, bleomycin and concurrent RT produced a more favorable CR rate, local-regional control rate, and relapse-free survival, but the difference in survival was not statistically significant.

Although patients were enrolled into the study at different calendar times, we consider time zero (i.e. baseline) for each patient to be the beginning of his or her treatment. We also conduct our analyses using *months* as the time scale, where a month is defined as 365/12 days.

Our dataset contains 96 patients. The first few rows of the dataset look like:

	arm	event	time	year
1	A	1	8.153425	78
2	A	1	5.260274	78
3	A	0	10.487671	78
4	A	1	9.106849	78
5	A	1	14.465753	78
6	A	1	2.991781	79

Before we conduct any Bayesian analyses we will estimate a simple Cox proportional hazards model for the effect of treatment arm on the hazard of death. We use the `coxph` modelling function from the widely-used **survival** R package (Therneau 2019). Estimation of the Cox model is based on a partial likelihood, but has a relationship with Bayesian survival analysis (Sinha, Ibrahim, and Chen 2003) and therefore serves a useful model for comparison in later sections of the notebook.

```
fit.coxph <- coxph(Surv(time, event) ~ arm, data = df, x = TRUE)
```

```
summary(fit.coxph)
```

```
## Call:
```

```
## coxph(formula = Surv(time, event) ~ arm, data = df, x = TRUE)
```

```
##
```

```
##   n= 96, number of events= 73
```

```
##
```

```
##           coef exp(coef) se(coef)      z Pr(>|z|)
```

```
## armB -0.5526    0.5754    0.2444 -2.261    0.0237 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##      exp(coef) exp(-coef) lower .95 upper .95
## armB    0.5754    1.738    0.3564    0.929
##
## Concordance= 0.558 (se = 0.032 )
## Likelihood ratio test= 5.23 on 1 df,  p=0.02
## Wald test               = 5.11 on 1 df,  p=0.02
## Score (logrank) test = 5.24 on 1 df,  p=0.02
```

Based on the Cox model there appears to be some evidence of a treatment effect with patients in Arm B (radiotherapy + chemotherapy) having approximately a 42% lower (HR = 0.58, 95% CI: 0.36 to 0.93) hazard of death when compared with Arm A (radiotherapy only).

### 3 Methodological framework

In this section we describe the methodological framework we will use throughout this notebook. This includes the modelling framework as well as several predicted quantities that can be derived from the fitted model.

We introduce these quantities in some detail since many of them are specific to survival analysis and may be new to some readers. Note however that this notebook does not describe the entirety of the methods available in **rstanarm** but rather the subset of methods relevant to the case study presented here.

#### 3.1 Notation

We assume that a true event time for individual  $i$  ( $i = 1, \dots, N$ ) exists and can be denoted  $T_i^*$ . However, in practice  $T_i^*$  may not be observed due to right censoring. We therefore observe  $T_i = \min(T_i^*, C_i)$  where  $C_i$  is a right censoring time. We define an event indicator  $d_i = I(T_i^* \leq C_i)$  where  $I(x)$  is the indicator function taking value 1 if  $x$  is true or value 0 otherwise.

Note that for simplicity we are only considering *right* censoring here, although the **stan\_surv** modelling function in **rstanarm** can in fact accommodate left, right, and interval censoring as well as delayed entry.

#### 3.2 Hazard, cumulative hazard and survival probability

There are three key quantities of interest in standard survival analysis: the hazard rate, the cumulative hazard, and the survival probability. It is these quantities that are used to form the likelihood function for the survival models in **rstanarm**.

The hazard of the event at time  $t$  is the instantaneous rate of occurrence for the event at time  $t$ . Mathematically, it is defined as:

$$h_i(t) = \lim_{\Delta t \rightarrow 0} \frac{P(t \leq T_i^* < t + \Delta t | T_i^* > t)}{\Delta t}$$

where  $\Delta t$  is the width of some small time interval.

The numerator is the conditional probability of the individual experiencing the event during the time interval  $[t, t + \Delta t)$  given that they were still at risk of the event at time  $t$ . The denominator converts the conditional probability to a rate per unit of time. As  $\Delta t$  approaches the limit, the width of the interval approaches zero and the instantaneous event rate is obtained.

The cumulative hazard is defined as:

$$H_i(t) = \int_{u=0}^t h_i(u) du$$

The survival probability is the probability that the individual does not experience the event before time  $t$ . It is defined as:

$$S_i(t) = \exp(-H_i(t)) = \exp\left(-\int_{u=0}^t h_i(u) du\right)$$

It can be seen here that in the standard survival analysis setting – where there is one event type of interest (i.e. no competing events) – there is a one-to-one relationship between each of the hazard, the cumulative hazard, and the survival probability.

### 3.3 Proportional hazards regression model

Rather than specifying a regression model for the survival time, the most common regression modelling approach in survival analysis is to model the effect that covariates have on the hazard.

In this case study we specify a proportional hazards model for the hazard of death. We consider the effect of treatment on the hazard of death. That is, treatment is the only covariate in our model. Therefore the hazard of death for individual  $i$  at time  $t$  is modelled as:

$$h_i(t) = h_0(t) \exp(\eta_i)$$

with linear predictor:

$$\eta_i = \beta_0 + \beta_1 x_i$$

and where  $h_0(t)$  is the baseline hazard (i.e. the hazard of death for an individual in the reference group, Arm B) at time  $t$ ,  $\beta_0$  denotes an intercept parameter,  $\beta_1$  denotes a coefficient for treatment, and  $x_i$  is a treatment indicator taking value 1 if the individual is in Arm A and value 0 if they are in Arm B.

The quantity  $\exp(\beta_1)$  is referred to as a *hazard ratio* (HR). The HR quantifies the relative increase in the hazard that is associated with a unit-increase in the relevant covariate, whilst hold any other covariates in the model constant. Therefore, in this case the HR quantifies the relative increase (or decrease) in the hazard of death that is associated with being in Arm A compared to Arm B.

Note that in this example our HR is a time-fixed quantity (i.e. it does not depend on  $t$ ), however in more complex settings we may wish to allow the HR to change as a function of time.

### 3.4 Baseline hazards

Different distributional assumptions can be made for the baseline hazard  $h_0(t)$  and affect how the baseline hazard changes as a function of time. The **rstanarm** package currently accommodates several standard parametric distributions for the baseline hazard (exponential, Weibull, Gompertz) as well as more flexible approaches that directly model the baseline hazard as a piecewise constant or smooth function of time using splines. In this case study we consider several of the available baseline hazards and discuss how they affect the fit of our model to the NCOG data.

### 3.4.1 Exponential model:

Under an exponential proportional hazards model the hazard rate is assumed to be constant over time. The baseline hazard takes the form:

$$h_0(t) = 1$$

and the quantity  $\lambda_i = \exp(\eta_i)$  is often referred to as the scale parameter.

### 3.4.2 Weibull model:

Under a Weibull proportional hazards model the hazard rate is assumed to be monotonically increasing or decreasing over time. The baseline hazard takes the form:

$$h_0(t) = \gamma t^{\gamma-1}$$

for some shape parameter  $\gamma > 0$  and the quantity  $\lambda_i = \exp(\eta_i)$  is often referred to as the scale parameter. In the special case where  $\gamma = 1$  the Weibull model reduces to the exponential model.

### 3.4.3 M-splines (or piecewise constant) model:

The exponential and Weibull models are obtained by making a distributional assumption for the event times and then deriving the corresponding parametric form for the hazard function. However, a potential issue with the exponential and Weibull models is that they do not allow much flexibility in how the hazard changes as a function of time. For instance they do not allow turning or inflection points in the hazard function. A more flexible alternative is therefore to model the hazard function directly using splines. For instance, M-splines (Ramsay 1988) can be used to directly model the baseline hazard function since they are strictly non-negative.

Under an M-spline model our baseline hazard takes the form:

$$h_0(t) = \sum_{l=1}^L \gamma_l M_l(t; \mathbf{k}, \delta)$$

where  $M_l(t; \mathbf{k}, \delta)$  denotes the  $l^{\text{th}}$  ( $l = 1, \dots, L$ ) basis term for a degree  $\delta$  M-spline function evaluated at a vector of knot locations  $\mathbf{k} = \{k_1, \dots, k_J\}$ , and  $\gamma_l$  denotes the  $l^{\text{th}}$  M-spline coefficient. The M-spline basis is evaluated using the method described in Ramsay (1988) and implemented in the **splines2** R package (Wang and Yan 2018).

To ensure that the hazard function  $h_i(t)$  is not constrained to zero at the origin (i.e. when  $t$  approaches 0) the M-spline basis incorporates an intercept. To ensure identifiability of the spline coefficients and the intercept parameter in the linear predictor we constrain the M-spline coefficients to a simplex, that is,  $\sum_{l=1}^L \gamma_l = 1$ .

The default degree in **rstanarm** is  $\delta = 3$  (i.e. cubic M-splines) such that the baseline hazard can be modelled as a flexible and smooth function of time, however this can be changed by the user. It is worthwhile noting that setting  $\delta = 0$  is treated as a special case that corresponds to a piecewise constant baseline hazard. In this notebook we consider cubic M-splines ( $\delta = 3$ ) and piecewise constant ( $\delta = 0$ ).

The vector of knot locations  $\mathbf{k} = \{k_1, \dots, k_J\}$  includes a lower boundary knot  $k_1$  at the earliest entry time (equal to zero if there isn't delayed entry) and an upper boundary knot  $k_J$  at the latest event or censoring time. The location of these boundary knots is enforced and cannot be changed by the user. The location of the remaining "internal" knots (that is  $k_2, \dots, k_{J-1}$  when  $J \geq 3$ ) can be explicitly specified by the user or are determined by default.

If the knot locations are not explicitly specified by the user (e.g. the user only specifies the degrees of freedom for the M-splines), then the default is to place the internal knots at equally spaced percentiles of the

distribution of uncensored event times. For instance, if there are three internal knots they would be placed at the 25<sup>th</sup>, 50<sup>th</sup>, and 75<sup>th</sup> percentiles of the distribution of the uncensored event times.

The number of internal knots and/or their locations can be controlled via the `basehaz_ops` argument to the `stan_surv()` modelling function. When specifying `basehaz_ops` the degrees of freedom for the M-spline function are  $L = J + \delta - 1$  where  $J$  is the total number of knots (including boundary knots).

### 3.5 Predictions and other quantities of interest

#### 3.5.1 Restricted mean survival time

The restricted mean survival time (RMST) can be interpreted as the average event-free survival time up to a pre-specified clinically important time point  $\tau$ . Whereas the survival probability is evaluated at a single time  $t$  and corresponds to a single point along the survival curve, the RMST is a summary of the entire *evolution* of the survival curve up to time  $\tau$ . It is essentially the area under the survival curve between  $t = 0$  and  $t = \tau$  however it is formally defined as follows.

For a (continuous) non-negative random variable  $T$ , limited to some horizon  $\tau$ , the RMST is denoted by  $\rho(\tau)$  and defined as:

$$\begin{aligned}\rho(\tau) &= \mathbb{E}[\min(T, \tau)] \\ &= \int_{x=0}^{\infty} \left( \int_{t=0}^{\infty} I(t \leq x) I(t \leq \tau) dt \right) p(x) dx \\ &= \int_{t=0}^{\infty} \left( \int_{x=0}^{\infty} I(t \leq x) p(x) dx \right) I(t \leq \tau) dt \\ &= \int_{t=0}^{\infty} S(t) I(t \leq \tau) dt \\ &= \int_{t=0}^{\tau} S(t) dt\end{aligned}$$

Where  $p(\cdot)$  above denotes the probability density function of R.V.  $T$ . Given posterior samples of the survival function  $S$  we can approximate  $\rho(\tau)$  using a 15-point Gauss-Kronrod quadrature, i.e.

$$\rho(\tau) \approx \frac{\tau}{2} \sum_{i=1}^{15} w_i S\left(\frac{\tau}{2} + \frac{\tau}{2} \chi_i\right)$$

where:

$i$	$\chi_i$	$w_i$
1	-0.99146	0.02294
2	-0.94911	0.06309
3	-0.86486	0.10479
4	-0.74153	0.14065
5	-0.58609	0.16900
6	-0.40585	0.19035
7	-0.20778	0.20443
8	0.00000	0.20948
9	0.20778	0.20443
10	0.40585	0.19035
11	0.58609	0.16900
12	0.74153	0.14065
13	0.86486	0.10479

$i$	$\chi_i$	$w_i$
14	0.94911	0.06309
15	0.99146	0.02294

In later sections of the notebook we will use the RMST as both a diagnostic tool (when conducting prior predictive checks) and for drawing inferences from our fitted model.

### 3.5.2 Standardised survival probabilities

Using our posterior draws from our fitted model, it is possible to generate posterior predictions for the survival probability, say  $\hat{S}_i(t)$ . Of course, the survival probability  $\hat{S}_i(t)$  is an individual-specific prediction in the sense that it is evaluated conditional on the  $i^{th}$  individual's covariates, say  $x_i$ .

However sometimes we wish to predict an “average” survival probability. One possible approach is to predict at the mean value of all covariates (Cupples et al. 1995). However this doesn't always make sense, especially not in the presence of categorical covariates. For instance, suppose our covariates are gender and a treatment indicator. Then predicting for an individual at the mean of all covariates might correspond to a 50% male who was 50% treated.

A better alternative is to average over the individual survival probabilities. This essentially provides an approximation to marginalising over the joint distribution of the covariates. At any time  $t$  it is possible to obtain a so-called standardised survival probability, denoted  $\hat{S}(t)$ , by averaging the individual-specific survival probabilities:

$$\hat{S}(t) = \frac{1}{N^P} \sum_{i=1}^{N^P} \hat{S}_i(t)$$

where  $\hat{S}_i(t)$  is the survival probability for individual  $i$  ( $i = 1, \dots, N^P$ ) at time  $t$ , and  $N^P$  is the number of individuals included in the predictions.

We can use this standardised survival probability for both inference and model checking. For instance, in later sections of this notebook we compare the standardised survival curve (evaluated using all individuals from our estimation sample) to the observed Kaplan-Meier survival curve. This provides a quick diagnostic tool for comparing the fit of the model to the observed data and is implemented via the `ps_check()` function in **rstanarm**.

### 3.5.3 Brier score

The Brier score is essentially a measure of average prediction error (“calibration”). It is based on the mean squared difference between expected and observed outcomes. Of course in survival analysis our outcome is an event *time* and so the Brier score becomes a time-dependent quantity. At some time  $t$  we can evaluate expected survival based on the survival probability (i.e.  $S_i(t)$ ) and evaluate observed survival as whether or not the individual was still event free (i.e.  $I(T_i > t)$ ). However we must also make some adjustments for censoring.

To provide a more reliable estimate of prediction error we use leave-one-out cross validation when calculating the Brier score. This helps to avoid over-optimism when calculating the prediction error. A formal definition of the Brier score used in this notebook follows.

Denote by  $\mathbf{T}_{-i}$  the  $N - 1$  dimensional vector of observed times for all individuals except individual  $i$ . Furthermore denote by  $\mathbf{T}$  the  $N$  dimensional vector of observed times for *all* individuals.

Now, write  $\bar{S}_i(t|\mathbf{T}_{-i})$  for the expected survival probability of individual  $i$  at time  $t$ , where the expectation is with respect to the posterior over latent parameters  $\boldsymbol{\theta}$ , given  $\mathbf{T}_{-i}$ .

Suppose  $\{\boldsymbol{\theta}_s\}_{s=1\dots S}$  is a (correlated) sequence of latent model parameters drawn from the posterior, given  $\mathbf{T}$ . We approximate  $S_i(t|\mathbf{T}_{-i})$  as follows:

$$S_i(t|\mathbf{T}_{-i}) \approx \frac{\sum_{s=1}^S r_{s,i} S_i(t|\boldsymbol{\theta}_s)}{\sum_{s=1}^S r_{s,i}}$$

where  $S_i(t|\boldsymbol{\theta}_s) = \hat{S}_i(t)$  is the probability that individual  $i$  survives beyond time  $t$ , given latent parameters  $\boldsymbol{\theta}_s$ , and:

$$r_{s,i} \equiv \frac{1}{p(T_i|\boldsymbol{\theta}_s)}$$

Coming to the Brier score, we now define

$$\eta_i(t) \equiv I(T_i > t) - S_i(t|\mathbf{T}_{-i})$$

Based on the residuals  $\eta_i(t)$  we define, following Gerds and Schumacher (2006), the *inverse probability of censoring weighted* estimator  $\Omega(t)$ :

$$\Omega(t) \equiv \frac{1}{N} \sum_{i=1}^N w_i(t) \eta_i(t)^2$$

with:

$$w_i(t) \equiv \frac{I(T_i \leq t) d_i}{G_i(T_i)} + \frac{I(T_i > t)}{G_i(t)}$$

Here  $G_i(t) = \mathbb{P}[C_i > t]$ . Note that for both the event and censoring survival functions we indicate a potential dependence on covariates by adding the index  $i$ .  $G_i(t)$  can be obtained by fitting, for example, a Cox model to the data with *reversed* event indicators, i.e.  $d_i \rightarrow 1 - d_i$ . By introducing a weighting based on  $G_i$  one can guarantee that the bias of the estimator for  $\Omega(t)$  does not depend on the survival model (otherwise model comparison would be a hopeless task). Finally, it can be shown that under weak conditions,  $\Omega(t)$  is a uniformly consistent estimator for the expected mean squared error in time (aka Brier score). We refer the reader for more details and proofs to Gerds and Schumacher (2006).

Practically, this means we can estimate  $S_i(t|\mathbf{T}_{-i})$  for all  $i$  and required times  $t$  using posterior samples  $S_i(t|\boldsymbol{\theta}_s)$  obtained from a model for all individuals. This matrix  $\{S_i(t|\mathbf{T}_{-i})\}_{i,t}$  can then be passed to the `pec()` function in the **pec** R package (Gerds 2018), which calculates  $\Omega(t)$  and provides ways to choose  $G_i$ .

## 4 Prior Predictive Checks

An essential part of any Bayesian workflow is carrying out prior predictive checks. Prior predictive checks are carried out by simulating directly from our joint prior distribution. They are carried out *before* conditioning on the data and can be used to assess whether the prior distributions for our model are sensible and only encompass values for the parameters that are considered plausible.

We start the prior predictive checks with an exponential proportional hazards model that has a constant baseline hazard. In **rstanarm** this model can be estimated using the `stan_surv()` modelling function and setting `basehaz = "exp"`. Setting the `prior_PD` argument equal to `TRUE` ensures that we do not condition on the outcome data and instead draw samples only from the prior distributions. In this first model we leave all prior specifications at their respective default values:



```

# declare desired parameters for stan
CHAINS <- 4
CORES <- 2
ITER <- 2000
SEED <- 42

# draw from the prior predictive distribution of the stan_surv survival model
prior.stan.const <- stan_surv(
  formula = Surv(time, event) ~ arm,
  data = df,
  basehaz = "exp",
  prior_PD = TRUE,
  chains = CHAINS,
  cores = CORES,
  iter = ITER,
  seed = SEED)

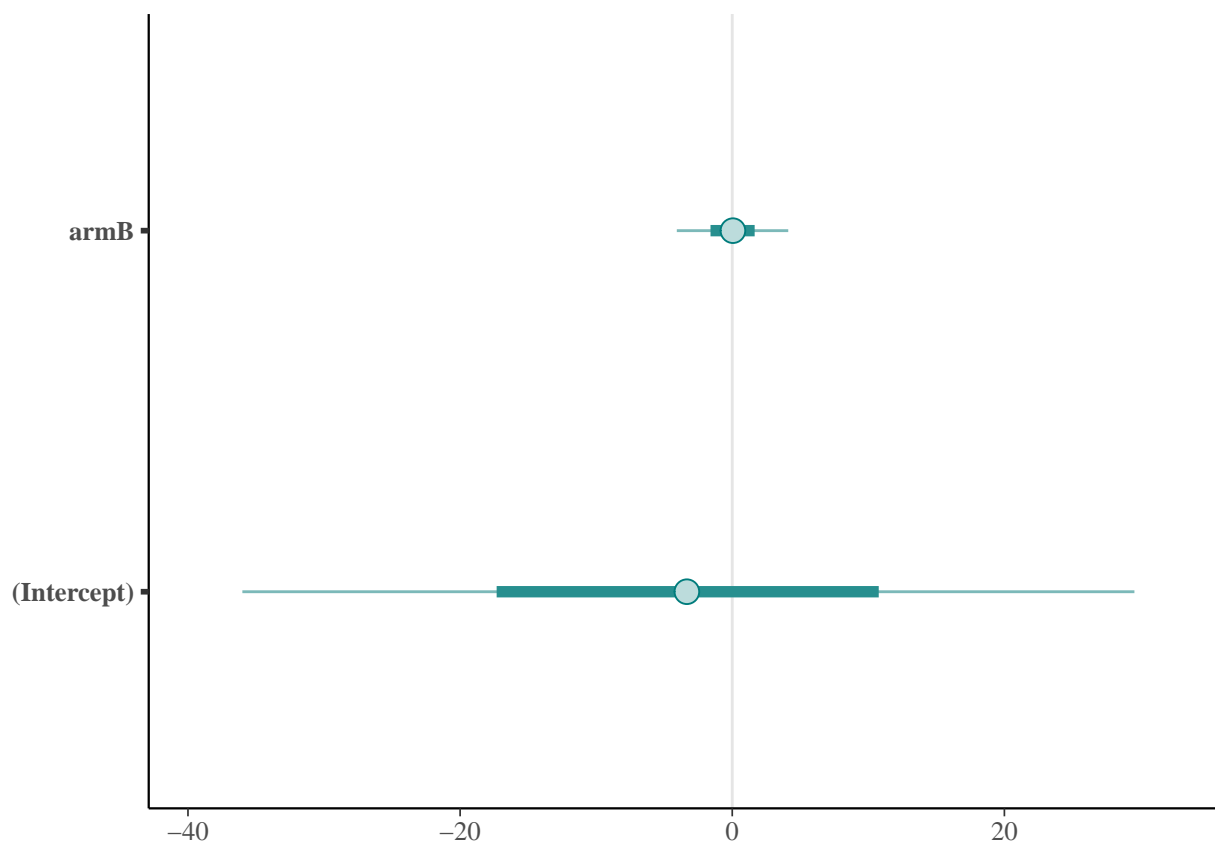
```

The first thing we can do is to inspect the marginal prior distributions for log HR for treatment (`armB`) and the intercept parameter (`(Intercept)`):

```

mcmc_intervals(prior.stan.const, pars = c("armB", "(Intercept)"))

```

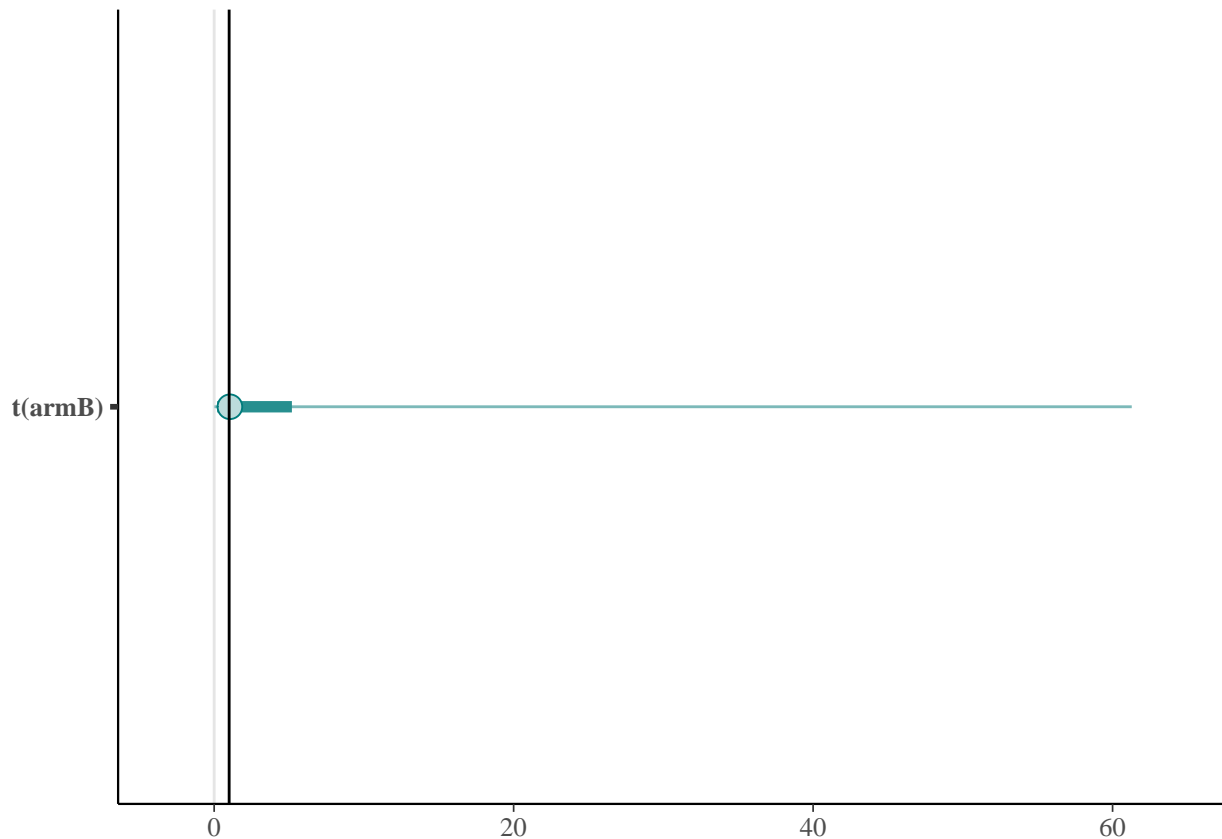


It is also natural to consider the marginal prior distribution for the HR for treatment, defined as  $\exp(\text{armB})$ :

```

mcmc_intervals(prior.stan.const, pars = c("armB"),
  transformations = exp) + vline_at(1)

```



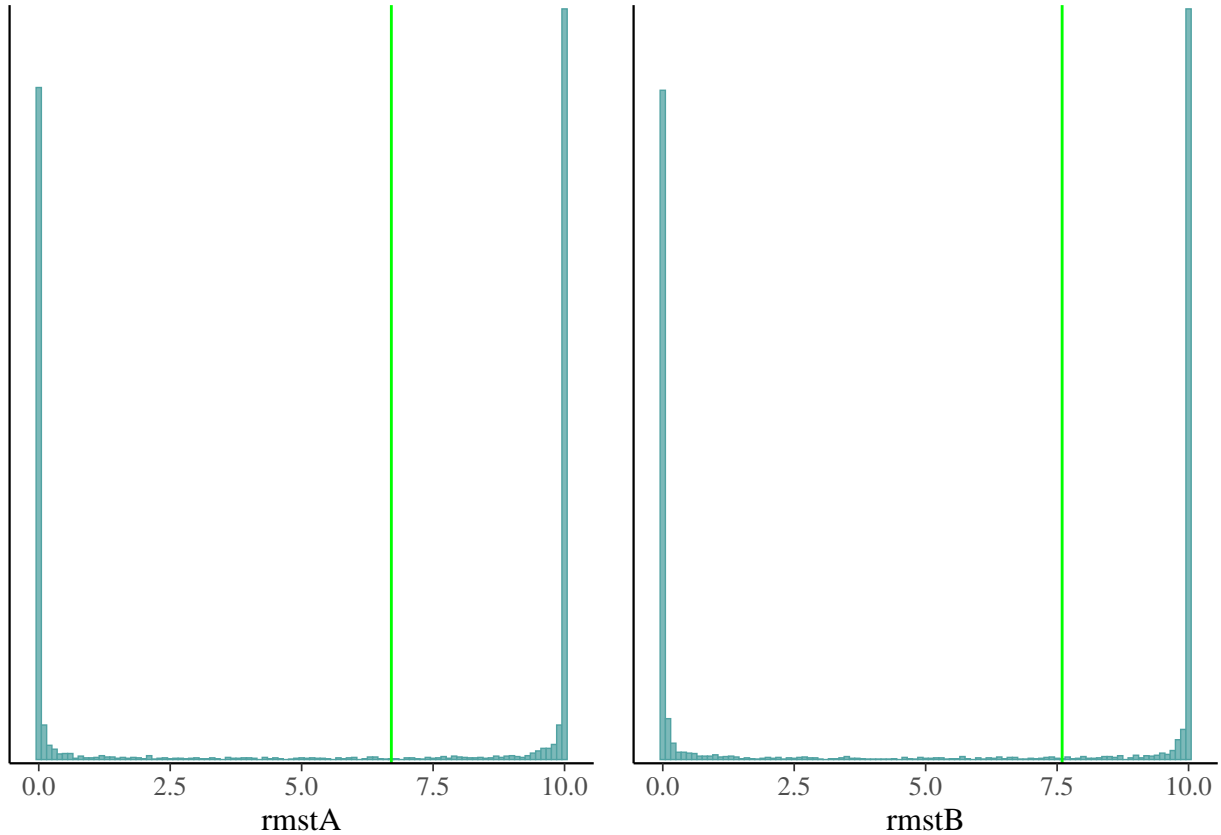
Subject matter experts would likely suggest that a HR around 1 would be plausible for a clinical trial such as NCOG. However, the plot of the marginal prior distribution for our HR shows some very extreme values. For instance, we can examine the 10 largest prior values for the HR and see that they are completely unrealistic for a clinical setting:

```
prior.stan.const %>%
  as.data.frame() %>%
  mutate(HR = exp(armB)) %>%
  select(HR, armB) %>%
  arrange(desc(HR)) %>%
  head(10)
```

```
##      HR      armB
## 1 7226.135 8.885460
## 2 4118.488 8.323241
## 3 3680.786 8.210882
## 4 3191.684 8.068304
## 5 2495.596 7.822283
## 6 2250.293 7.718816
## 7 1950.331 7.575754
## 8 1399.292 7.243721
## 9 1372.624 7.224480
## 10 1326.364 7.190196
```

Let's now inspect the prior predictive distributions for the RMST at horizon  $\tau = 10$ , that is  $\rho(10)$ . To do this we will use the `rmst_check_plot()` function that we defined (internally) at the start of this notebook. See the R Markdown source file for this notebook to see the definition of the `rmst_check_plot()` function:

```
rmst_check_plot(prior.stan.const, df, tau = 10)
```

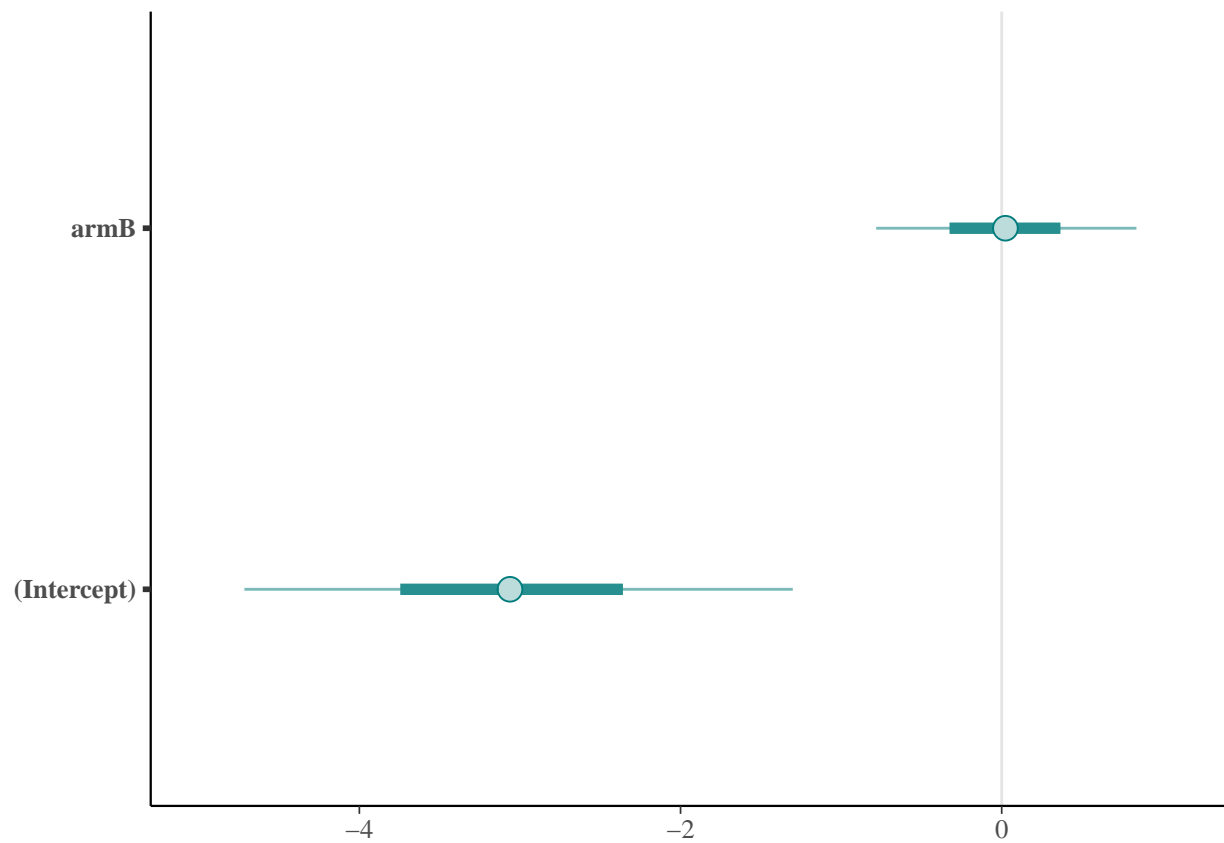


This pile-up of mass in both tails is too extreme and does not reflect prior knowledge, which would most often exclude at least one of the two extreme peaks. This and the previous HR observation are reason enough for us to adjust the default prior distributions for the log HR for treatment (`armB`) and the intercept (`(Intercept)`). To fit the model with the new prior distributions we will use the `update()` method for `stansurv` models:

```
prior.stan.const <- update(prior.stan.const,
                           prior_intercept = normal(0, 1),
                           prior           = normal(0, 0.5))
```

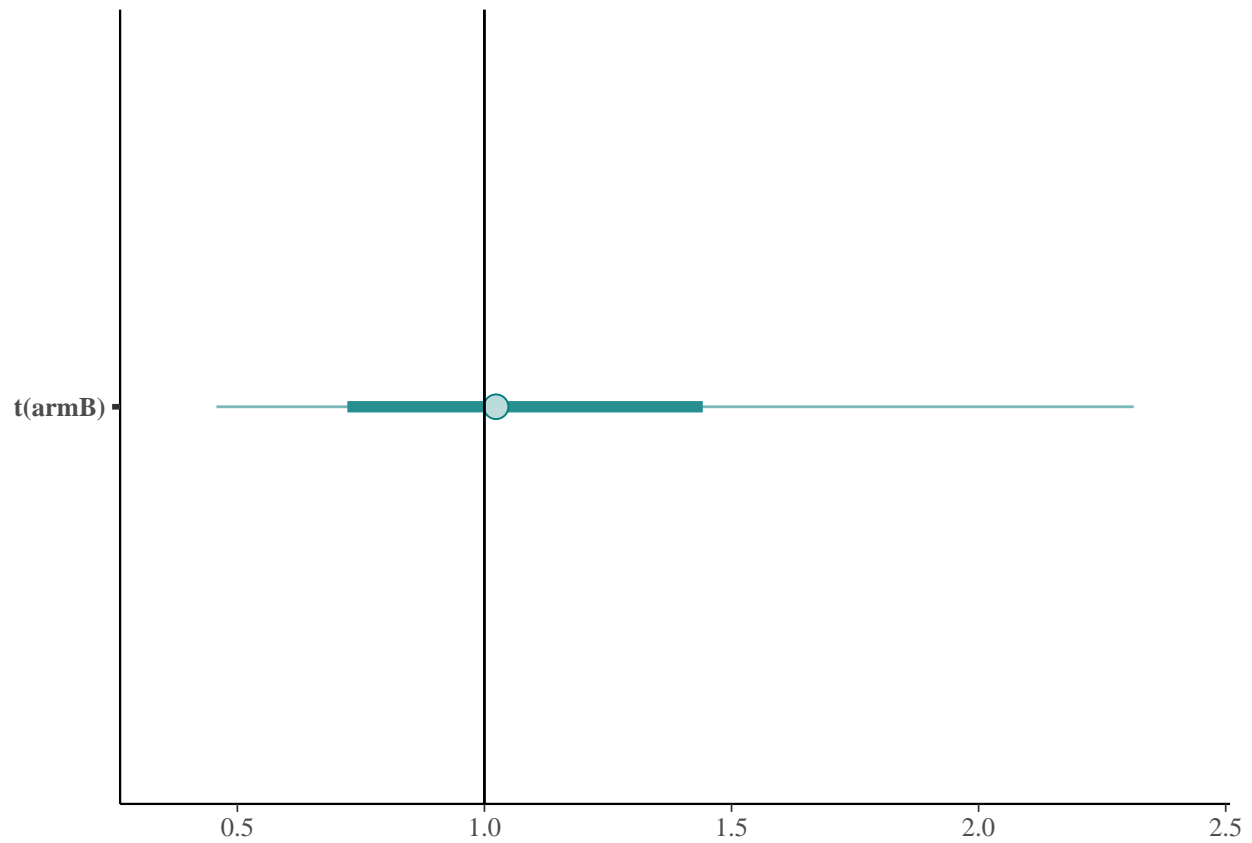
Again, we can start by looking at the marginal prior distributions:

```
mcmc_intervals(prior.stan.const, pars = c("armB", "(Intercept)"))
```



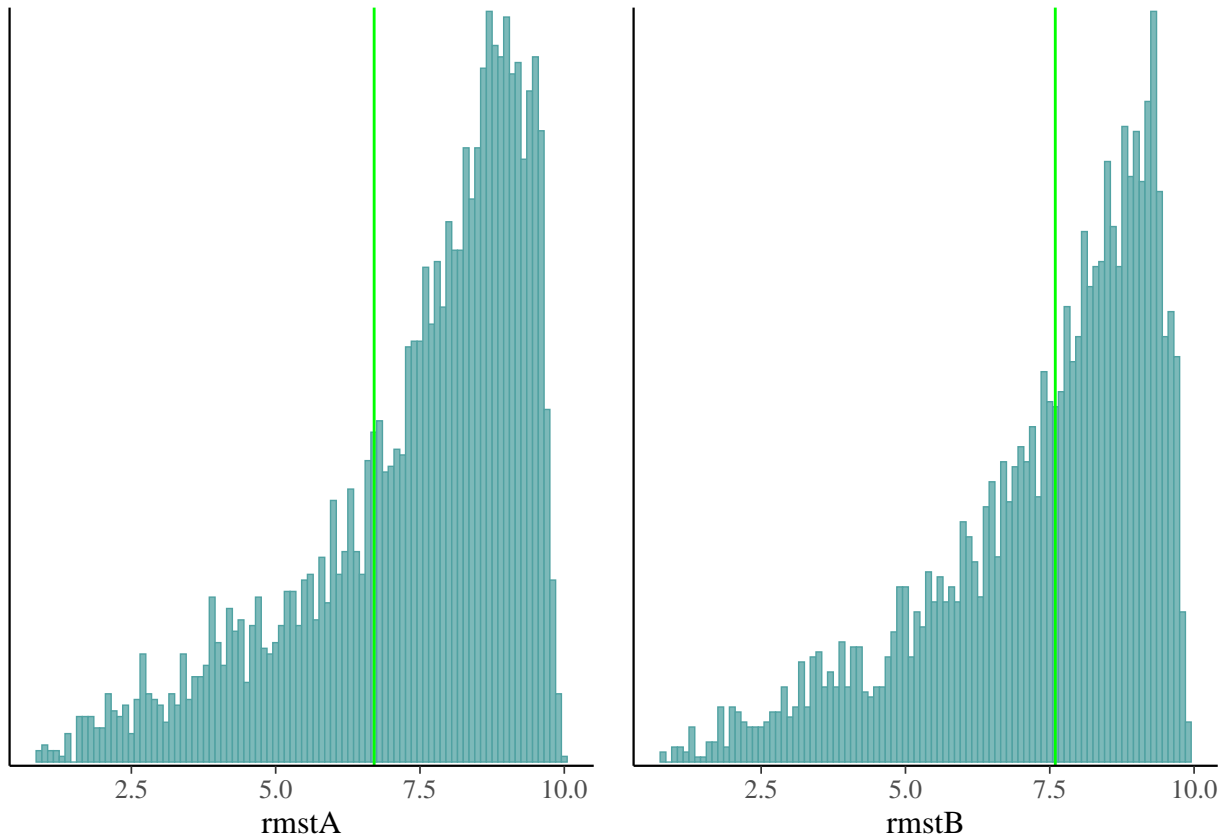
The induced marginal prior distribution for the HR also looks more reasonable now:

```
mcmc_intervals(prior.stan.const, pars = c("armB"),
               transformations = exp) + vline_at(1)
```



The prior predictive checks for the RMST  $\rho(10)$  also look better with the new prior distributions (at least we avoid the extreme tails):

```
rmst_check_plot(prior.stan.const, df, tau = 10)
```



Of course we could go on and optimize this further. For instance, we can imagine a scenario where subject matter experts suggest a particular value for `rmstA` and `rmstB` and some uncertainty around it, or perhaps they would suggest a uniform prior distribution over `rmstA` and `rmstB`. However at this stage, for the purposes of this notebook, we will decide to accept these prior distributions as adequate and start our inference (that is we will condition on the observed data!).

## 5 Preliminary model fitting

### 5.1 Model estimation

Given the `prior.stan.const` model we can use the `update()` function again but this time set `prior_PD = FALSE` so that we condition on the outcome data. This will generate posterior draws for our exponential proportional hazards model so that we can conduct our inference:

```
fit.stan.const <- update(prior.stan.const, prior_PD = FALSE)
print(fit.stan.const, digits = 3)
```

```
## stan_surv
## baseline hazard: exponential
## formula:         Surv(time, event) ~ arm
## observations:    96
## events:          73 (76%)
## right censored:  23 (24%)
## delayed entry:   no
## -----
##               Median MAD_SD exp(Median)
## (Intercept) -2.730  0.153      NA
```

```
## armB          -0.625  0.213  0.535
##
## -----
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

The sampler did not issue any warnings related to divergences or exceedance of max-treedepth, which is reassuring. We observe that the inferred median treatment effect is  $-0.625$  with an estimate for `MAD_SD` of  $0.213$ , which is a measure of the standard deviation of the marginal posterior distribution of the treatment effect. More precisely, it is based on a scaling of the Median Absolute Deviation (MAD) from the posterior median of the treatment effect.

Overall, the estimated treatment effect can be considered compatible with the one we obtained from the simple Cox proportional hazard model above, albeit with large uncertainty margin (in both estimates).

## 5.2 Prior vs posterior checks

### 5.2.1 Intercept and log hazard ratio

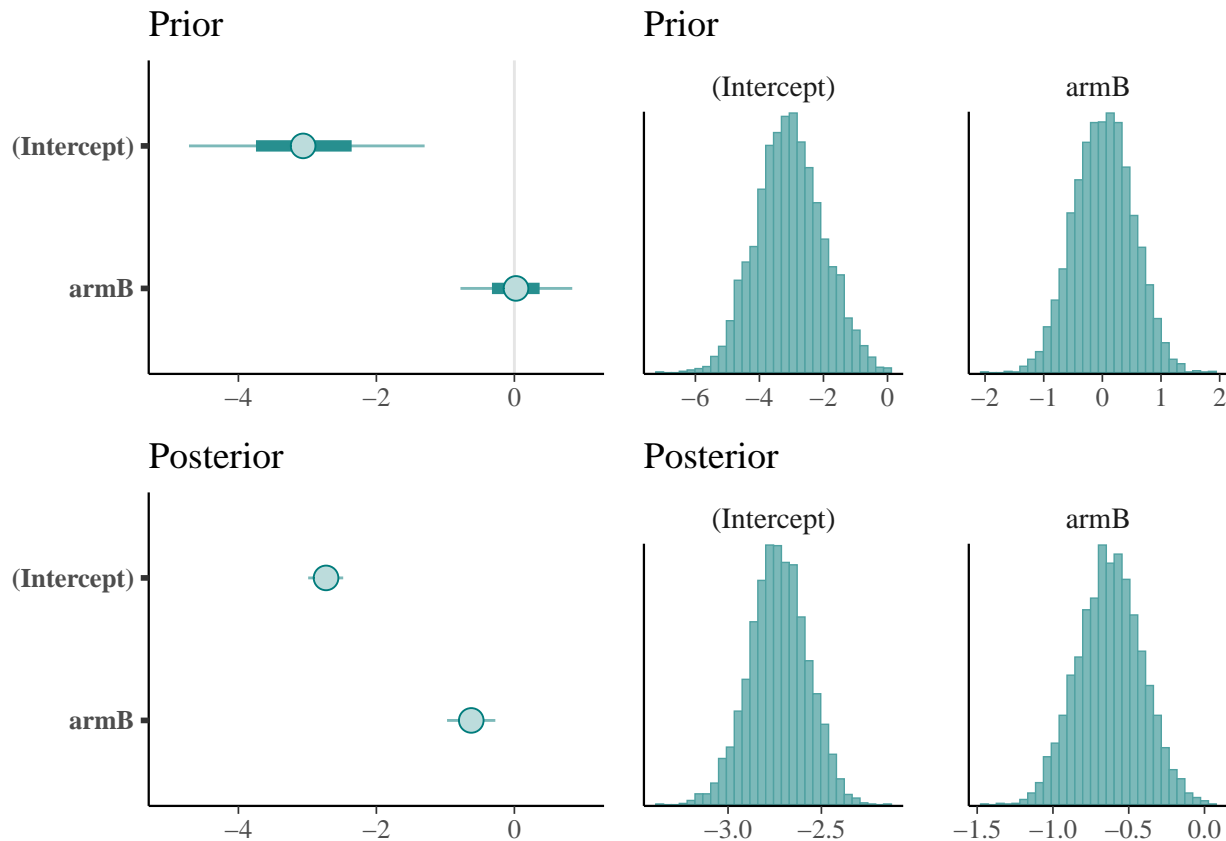
It is particularly useful to visualize the posterior results and compare them to our priors. First we examine the estimated parameters in the model (the intercept and the log HR for treatment):

```
plot_grid(

  bayesplot_grid(mcmc_intervals(prior.stan.const),
                 mcmc_intervals(fit.stan.const),
                 titles      = c("Prior", "Posterior"),
                 xlim        = c(-5, 1),
                 grid_args = list(nrow = 2)),

  bayesplot_grid(mcmc_hist(prior.stan.const),
                 mcmc_hist(fit.stan.const),
                 titles      = c("Prior", "Posterior"),
                 grid_args = list(nrow = 2)),

  ncol = 2
)
```



Here we see that our estimated posterior distributions are not at all incompatible with our prior beliefs. This is reassuring.

### 5.2.2 Hazard ratio

We can also visually compare the prior and posterior for the estimated HR. For comparison we also show the HR calculated using the Cox proportional hazards model (vertical green line):

```
add_cox_hr <- vline_at(exp(coef(fit.coxph)), color = "green")

bayesplot_grid(

  mcmc_hist(prior.stan.const,
    pars      = c("armB"),
    transformations = exp,
    binwidth  = 0.05) + add_cox_hr,

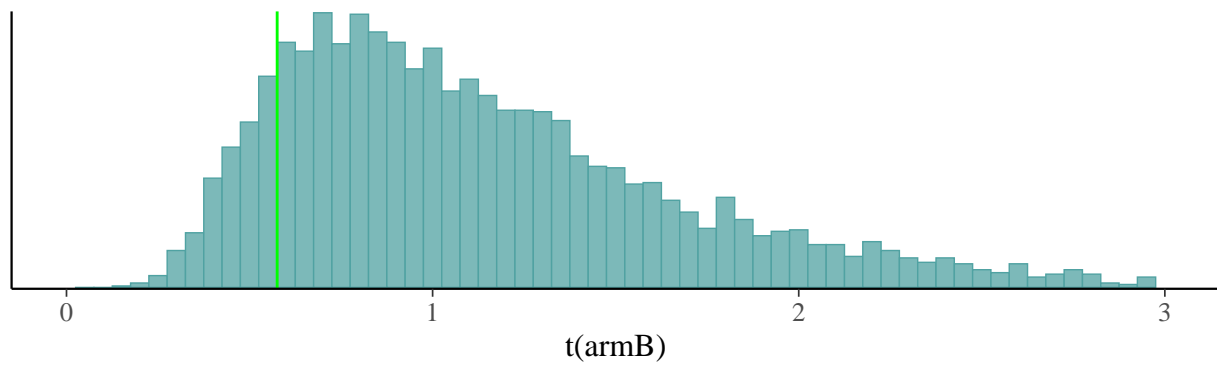
  mcmc_hist(fit.stan.const,
    pars      = c("armB"),
    transformations = exp,
    binwidth  = 0.05) + add_cox_hr,

  titles     = c("Prior", "Posterior"),
  xlim       = c(0, 3),
  grid_args  = list(nrow = 2)

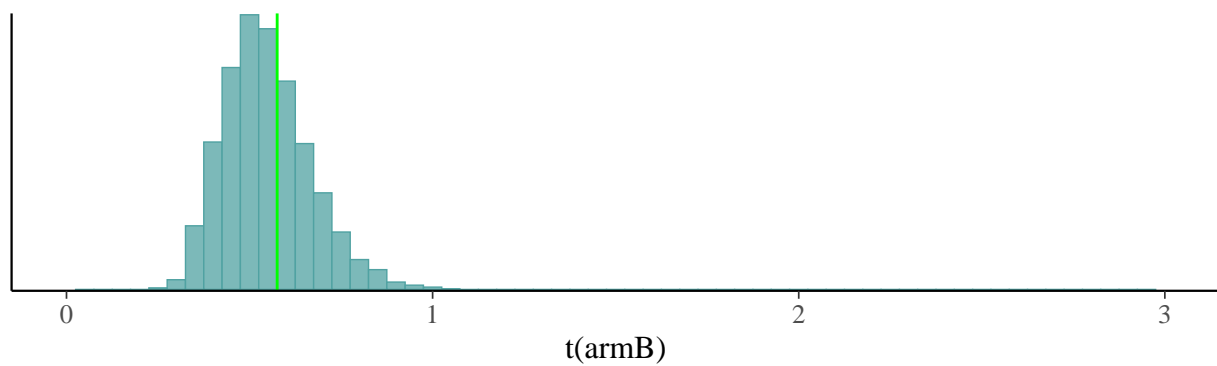
)
```



Prior



Posterior



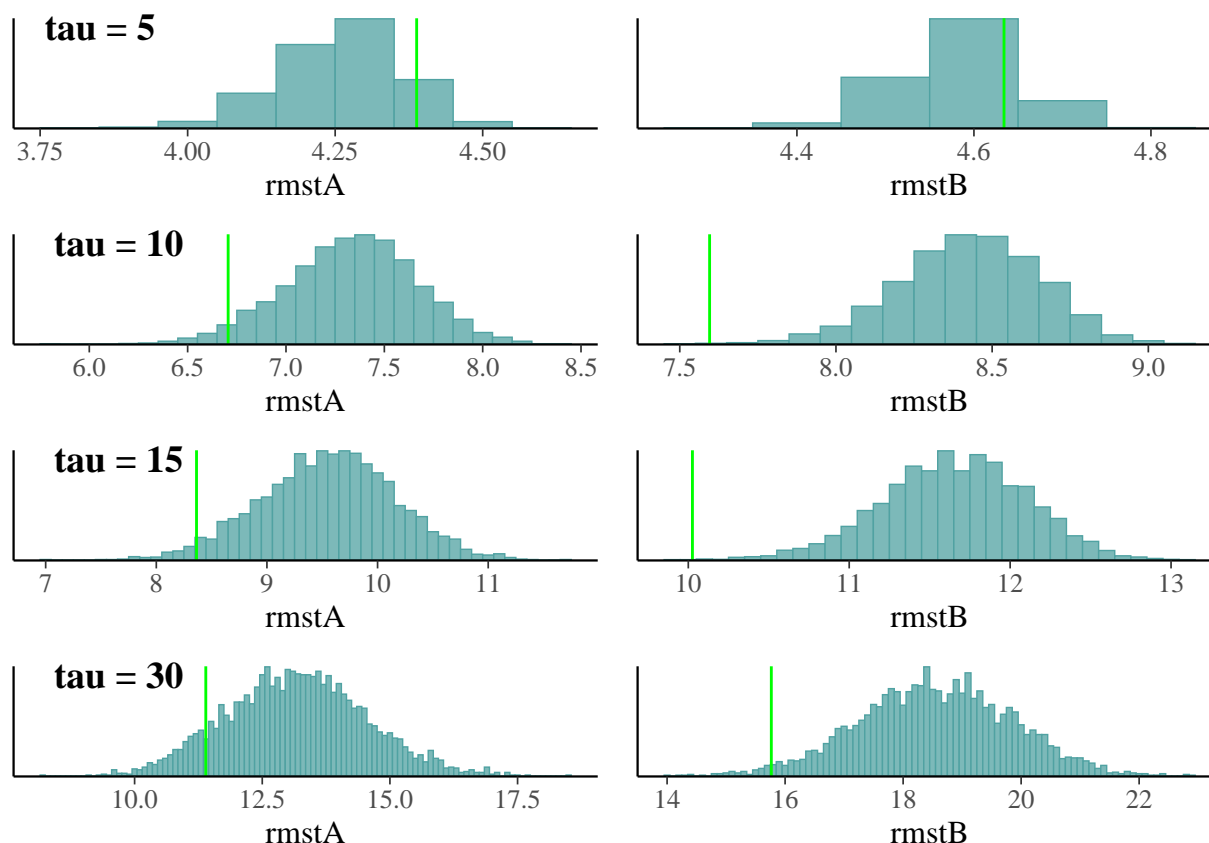
### 5.2.3 Restricted mean survival time

We now continue with posterior checks using RMST for  $\tau = 5, 10, 15, 30$ . The green vertical lines below show the respective non-parametric estimate for  $\rho(\tau)$  as calculated by the `survRM2::rmst2` function (Uno et al. 2017).

```
tau_values <- c(5, 10, 15, 30)

plots <- map(tau_values, ~ rmst_check_plot(fit.stan.const, df, tau = .))

plot_grid(plotlist = plots,
  labels = sprintf("tau = %d", tau_values),
  nrow = 4)
```



This doesn't look good! The RMST evaluated from our fitted model is quite different from the non-parametric estimate of the RMST based on the data (except potentially for  $\tau = 5$ ). It appears that the model has the worst performance when  $\tau = 10, 15$ , or  $30$  where it tends to predict better survival (larger RMST) than observed in the data.

Since this pattern is observed for both treatment groups it might suggest that this is an issue with the estimated baseline hazard (i.e. (`Intercept`)) rather than the estimated effect of treatment (`armB`). Of course, the exponential proportional hazards model assumes a constant baseline hazard, which may be unrealistic for this data. Thus let's consider a more flexible form for the baseline hazard, while still assuming proportional hazards for the effect of treatment.

## 6 Model extensions and comparison

To provide greater flexibility for the baseline hazard we will consider a Weibull baseline hazard, a piecewise-constant baseline hazard (with two different numbers of knots), and a cubic M-spline baseline hazard (also with two different numbers of knots).

### 6.1 Model estimation

To fit these models we again use the `update()` function and pass in our previous fitted `stansurv` model as well as arguments to change the baseline hazard (`basehaz`) and the number of knots (`basehaz_ops`):

```
# weibull model
fit.stan.weib <- update(fit.stan.const,
                        basehaz      = "weibull")

# cubic m-spline model (with df = 5)
fit.stan.ms5  <- update(fit.stan.const,
```

```

        basehaz      = "ms",
        basehaz_ops = list(df = 5))

# cubic m-spline model (with df = 10)
fit.stan.ms10 <- update(fit.stan.const,
                        basehaz      = "ms",
                        basehaz_ops = list(df = 10))

# piecewise constant model (with df = 5)
fit.stan.pw5  <- update(fit.stan.const,
                        basehaz      = "ms",
                        basehaz_ops = list(degree = 0, df = 5))

# piecewise constant model (with df = 10)
fit.stan.pw10 <- update(fit.stan.const,
                        basehaz      = "ms",
                        basehaz_ops = list(degree = 0, df = 10))

fits_stan <- list("Constant"      = fit.stan.const,
                  "Weibull"       = fit.stan.weib,
                  "MS (df = 5)"   = fit.stan.ms5,
                  "MS (df = 10)"  = fit.stan.ms10,
                  "PW (df = 5)"   = fit.stan.pw5,
                  "PW (df = 10)"  = fit.stan.pw10
)

```

## 6.2 Comparison of estimated baseline hazards

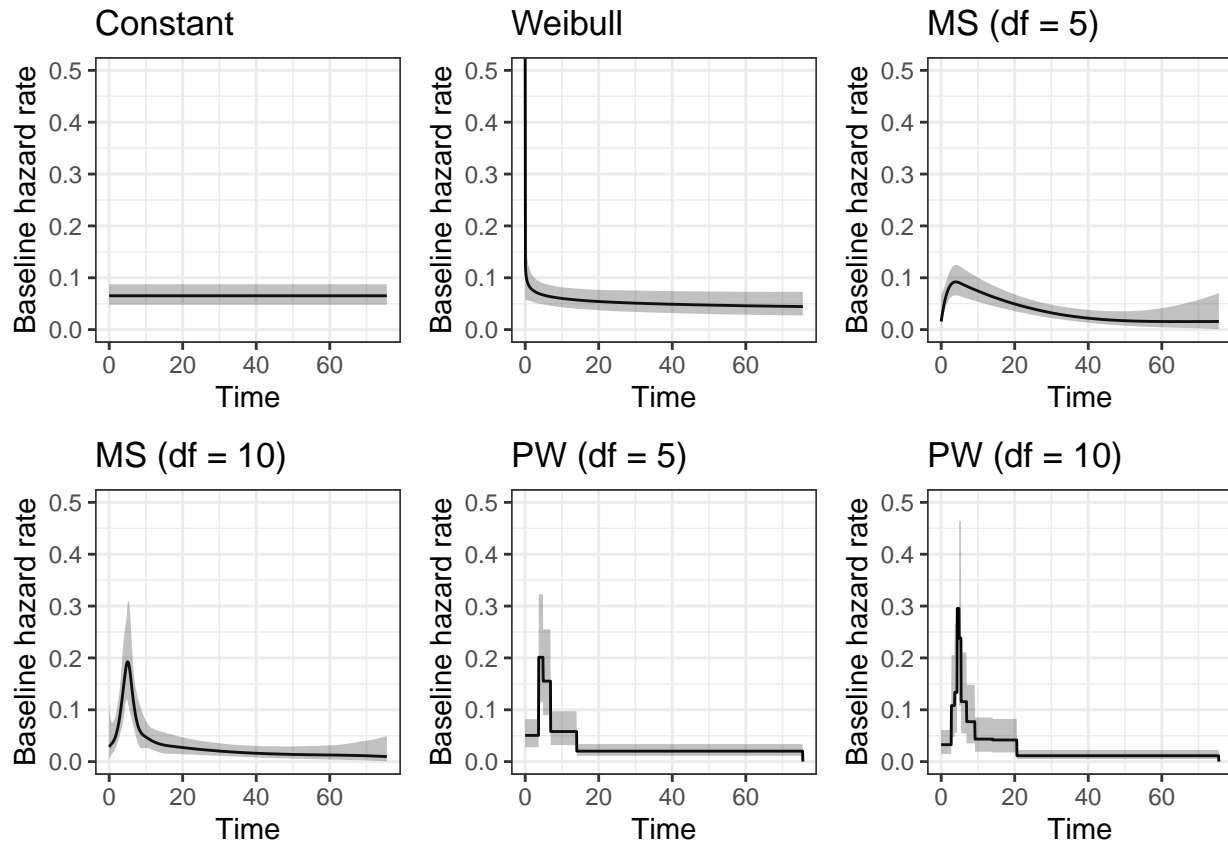
We can visually compare the estimated baseline hazards for each of the fitted models. This can be achieved using the default `plot()` function for `stansurv` models. In the plot below the abbreviation “PW” is used for piecewise constant and “MS” for M-splines:

```

plots <- map(fits_stan, plot)

bayesplot_grid(
  plots      = plots,
  ylim       = c(0, 0.5),
  titles     = names(fits_stan),
  grid_args  = list(ncol = 3))

```



It is particularly apparent from the plot that the constant baseline hazard (i.e. exponential model) is inappropriate – it fails to capture any changes in the baseline hazard. Conversely, the other baseline hazards quite clearly capture changes in the baseline hazard over time.

### 6.3 Assessing fit of the survival function

For each model we can compare the posterior estimate of the standardised survival curve to the Kaplan-Meier survival curve. This helps to assess how well each model fits the observed data. This approach is implemented via the `ps_check()` function in `rstanarm`. We will also write a short helper function to visualize the knot placement for the piecewise constant and M-spline models:

```
# define helper function to add knot locations
add_knots <- function(x) {

  knots <- x$basehaz$knots

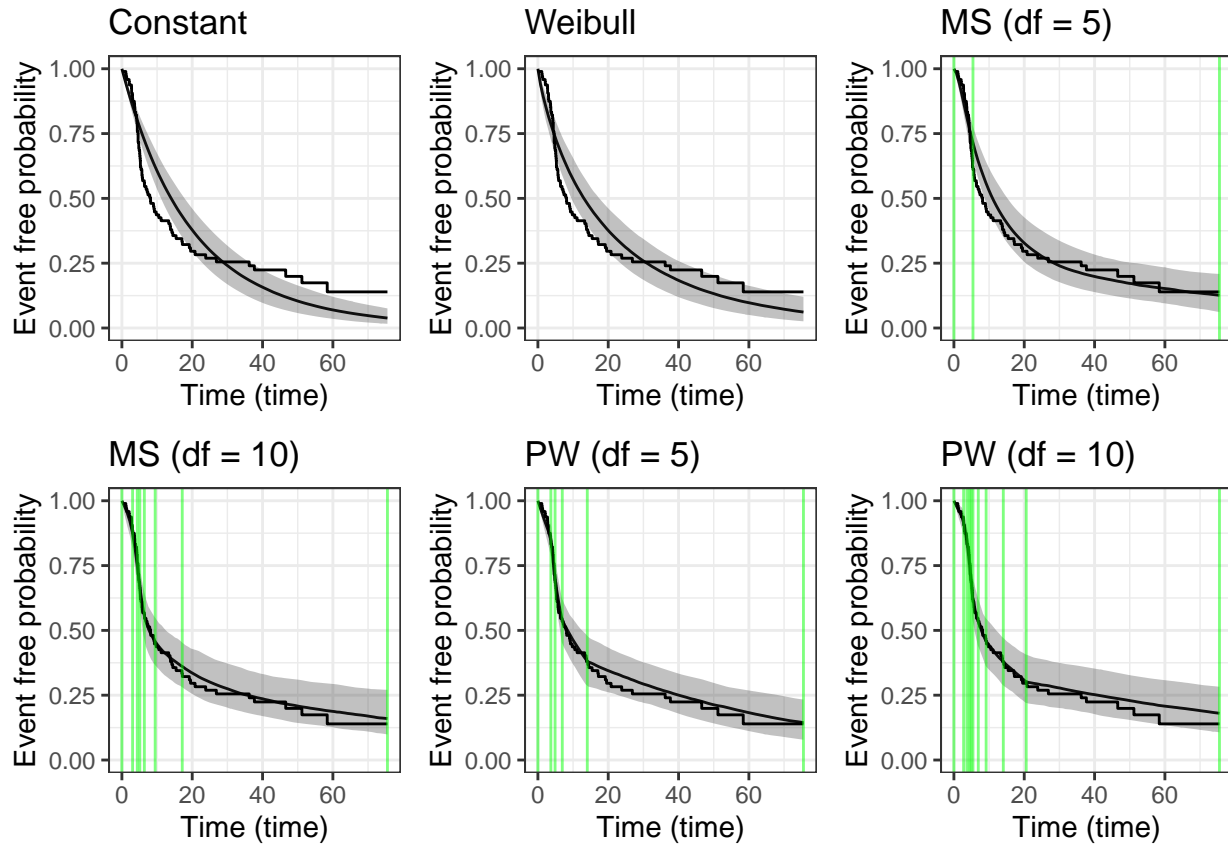
  if (is.null(knots))
    return(NULL)

  geom_vline(xintercept = knots, color = "green", alpha = 0.5)
}

# generate the 'ps_check' plots
plots <- map(fits_stan, ~ (ps_check(.) + add_knots(.)))

# combine the plots
bayesplot_grid(
  plots = plots,
```

```
titles = names(fits_stan),
grid_args = list(ncol = 3))
```



Again we see evidence that the constant (i.e. exponential) model fits very poorly. The Weibull model also appears to be inadequate. The remaining models appear to do reasonably well, although there is some evidence that the M-spline model with only 5 degrees of freedom may not provide sufficient flexibility.

## 6.4 Comparison using leave-one-out cross validation

The `stansurv` objects returned by `stan_surv()` are fully compatible with the `loo` R package (Vehtari, Gelman, and Gabry 2019). The `loo` package allows efficient leave-one-out cross validation and model comparison based on the expected log predictive density (`elpd`):

```
loos <- map(fits_stan, loo)
```

```
loo_compare(loos)
```

```
##           elpd_diff se_diff
## MS (df = 10)    0.0      0.0
## PW (df = 10)  -1.0      3.0
## PW (df = 5)   -2.8      3.0
## MS (df = 5)   -4.5      4.5
## Weibull       -16.5     6.5
## Constant      -17.7     7.1
```

Based on the differences in `elpd` we can conclude that the models with the most restrictive baseline hazards – namely the constant (exponential) and Weibull models – are inferior to the others. However there is no clear winner among the remaining models. The `loo` estimates suggest that the piecewise constant and cubic

M-spline models all fit the data similarly well, although there is some indication that increasing the number of knots (i.e. higher degrees of freedom) slightly improves the fit of the model.

## 6.5 Restricted mean survival time

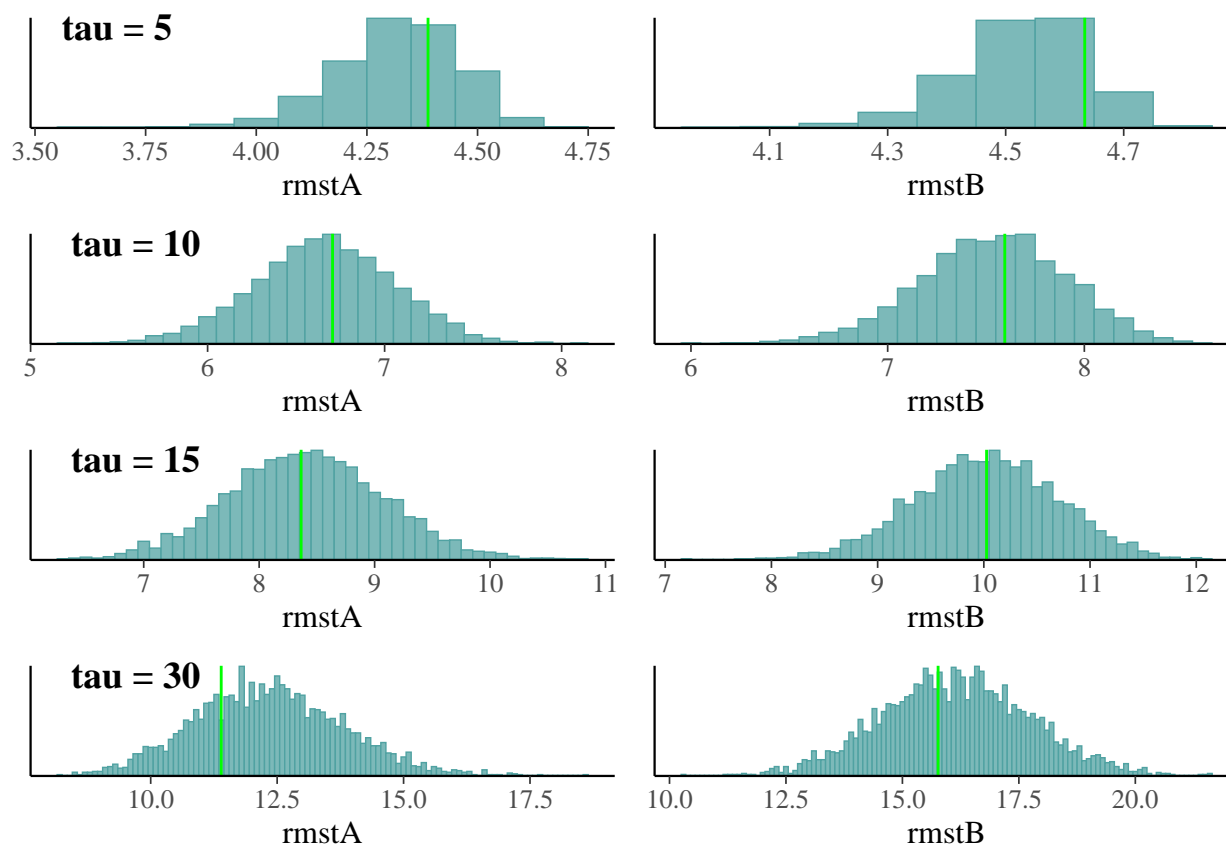
Let's now examine the posterior estimate of the RMST for the best fitting model. We will evaluate the RMST for each treatment arm at each of four time horizons, namely  $\tau = 5, 10, 15$ , and  $30$ . This will allow us to compare the estimates with the RMST that we obtained for the constant baseline hazard (exponential) model in earlier sections.

Here we plot the RMST for the model with lowest estimated `elpd` (i.e. the "best" fitting model), which is the M-spline model with `df=10`. Similarly to before we use the vertical green line to show the non-parametric estimate of the RMST based on the observed data:

```
tau_values <- c(5, 10, 15, 30)

plots <- map(tau_values, ~ rmst_check_plot(fit.stan.ms10, df, .))

plot_grid(plotlist = plots,
  labels = sprintf("tau = %d", tau_values),
  nrow = 4)
```



The plots demonstrate that the posterior estimate of the RMST in each treatment arm is far better than it was under the constant baseline hazard model. We can be confident that our flexible M-spline model provides a much better fit to the observed data when concerned with the RMST.

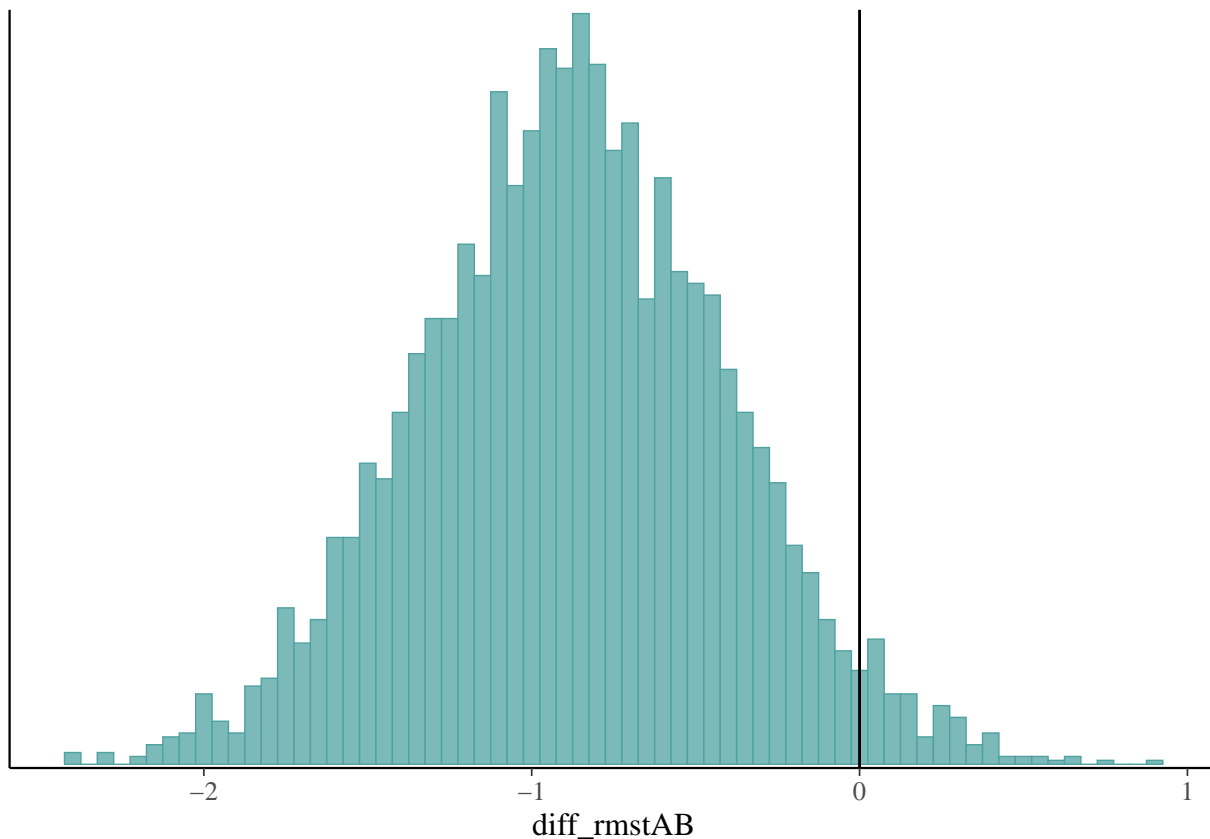
## 7 Treatment effects based on the RMST

When analysing clinical survival data the HR is by far the most commonly used measure of a treatment effect. However, an alternative measure of treatment effect is the absolute difference in RMST between the two arms.

A key advantage of the RMST is that it does not require a proportional hazard structure in the data, and hence is applicable for cases with e.g. time dependent treatment effects, which are a particular instance of situations that lead to non-proportional hazards. For more details see e.g. Royston and Parmar (2013). Furthermore it is worth emphasising that the RMST has a very intuitive interpretation. For illustration we plot the difference in RMST at a horizon time of  $\tau = 10$  based on our best fitting model:

```
df_rmst <- rmst_check(fit.stan.ms10, tau = 10)

df_rmst %>%
  mutate(diff_rmstAB = rmstA - rmstB) %>%
  mcmc_hist(pars = "diff_rmstAB", binwidth = 0.05) + vline_0()
```



The posterior distribution for the absolute difference in RMST (between the two treatment arms) has the majority of its mass located below zero. Because the difference is negative this provides evidence that the RMST is higher in Arm B – a finding that aligns with our earlier conclusions – namely that survival is better for treatment Arm B (radiotherapy + chemotherapy).

We can also estimate the posterior probability that the RMST at  $\tau = 10$  is larger for Arm B than Arm A:

```
mean(df_rmst$rmstB > df_rmst$rmstA) %>% round(3)
```

```
## [1] 0.966
```

Below we compare the RMST treatment effect at  $\tau = 10$  among all fitted models:

```

d <- fits_stan %>%
  map_dfr(~ rmst_check(.,10) %>%
    mutate(diffAB = rmstA - rmstB) %>%
    select(diffAB))

model_nms <- rep(names(fits_stan), each = nrow(d) / length(fits_stan))

d <- d %>% mutate(model = model_nms)

# Credit for this code goes to Eric Novik
# c.f. https://gist.github.com/ericnovik/20e8f292c5ddb125c79bfff4bc8e8f77a

n <- nrow(d) / length(unique(d$model))

d %<>%
  group_by(model) %>%
  mutate(x = density(diffAB, n = n)$x,
    y = density(diffAB, n = n)$y)

cutoff <- 0

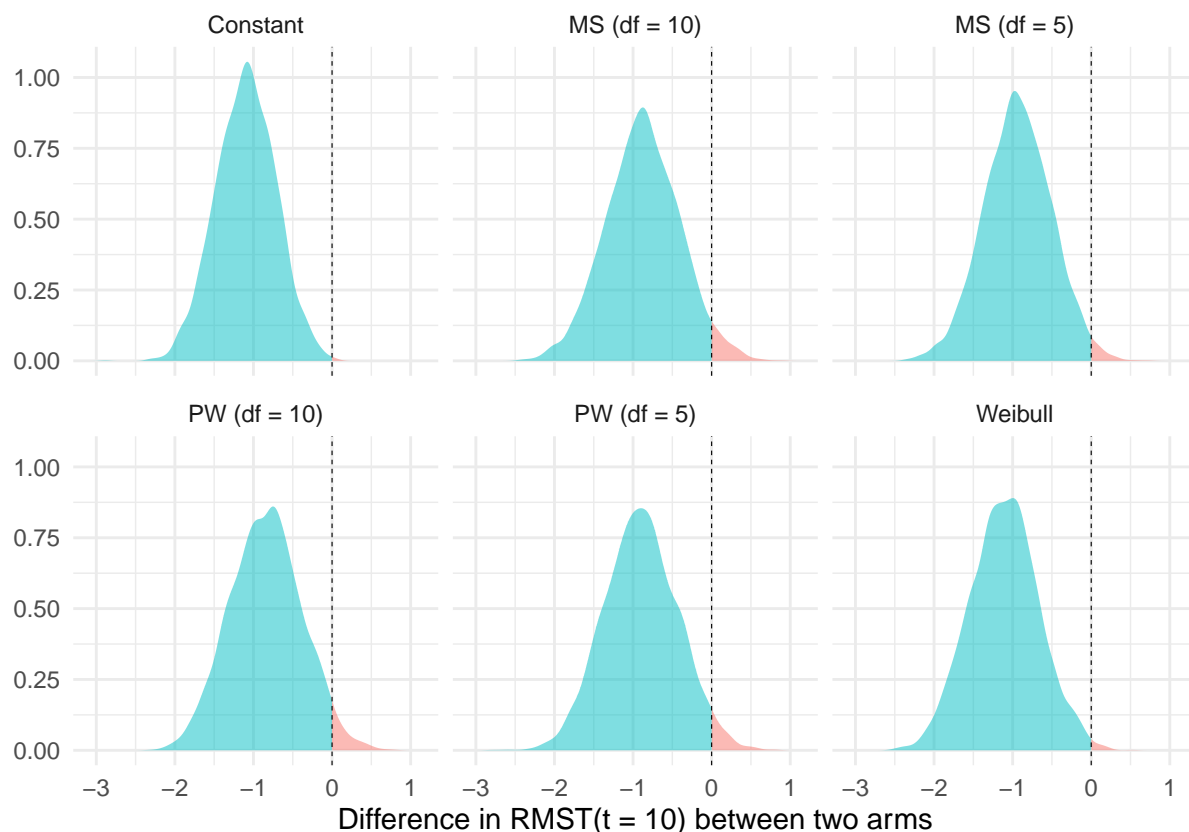
d <- mutate(d, area = x < cutoff)

p <- ggplot(d, aes(x, y)) +
  geom_ribbon(aes(ymin = 0, ymax = y, fill = area), alpha = 1/2) +
  geom_vline(xintercept = cutoff, size = 0.2, linetype = 2) +
  facet_wrap(~ model) +
  ylab("") +
  xlab("Difference in RMST(t = 10) between two arms") +
  theme_minimal() +
  theme(legend.position = "none")

```

p





All models share the qualitative observation that the majority of posterior mass is on events for which Arm B has a larger RMST than Arm A, albeit qualitatively there are differences between the models. In particular, we observe that the Weibull and Constant models both are somewhat “over optimistic” regarding the advantage of Arm B over Arm A. For completeness, we show below the corresponding posterior probabilities for all models:

```
pprob_RMSTBgtRMSTA <- function(fit, tau = 10) {
  rmst_check(fit, tau = tau) %>%
    mutate(IND = rmstB > rmstA) %>%
    pull("IND") %>%
    mean() %>%
    round(3)
}

tibble(
  Model = names(fits_stan),
  Probability = map_dbl(fits_stan, pprob_RMSTBgtRMSTA)
)
```

```
## # A tibble: 6 x 2
##   Model      Probability
##   <chr>         <dbl>
## 1 Constant      0.999
## 2 Weibull       0.995
## 3 MS (df = 5)   0.985
## 4 MS (df = 10)  0.966
## 5 PW (df = 5)   0.972
## 6 PW (df = 10)  0.97
```

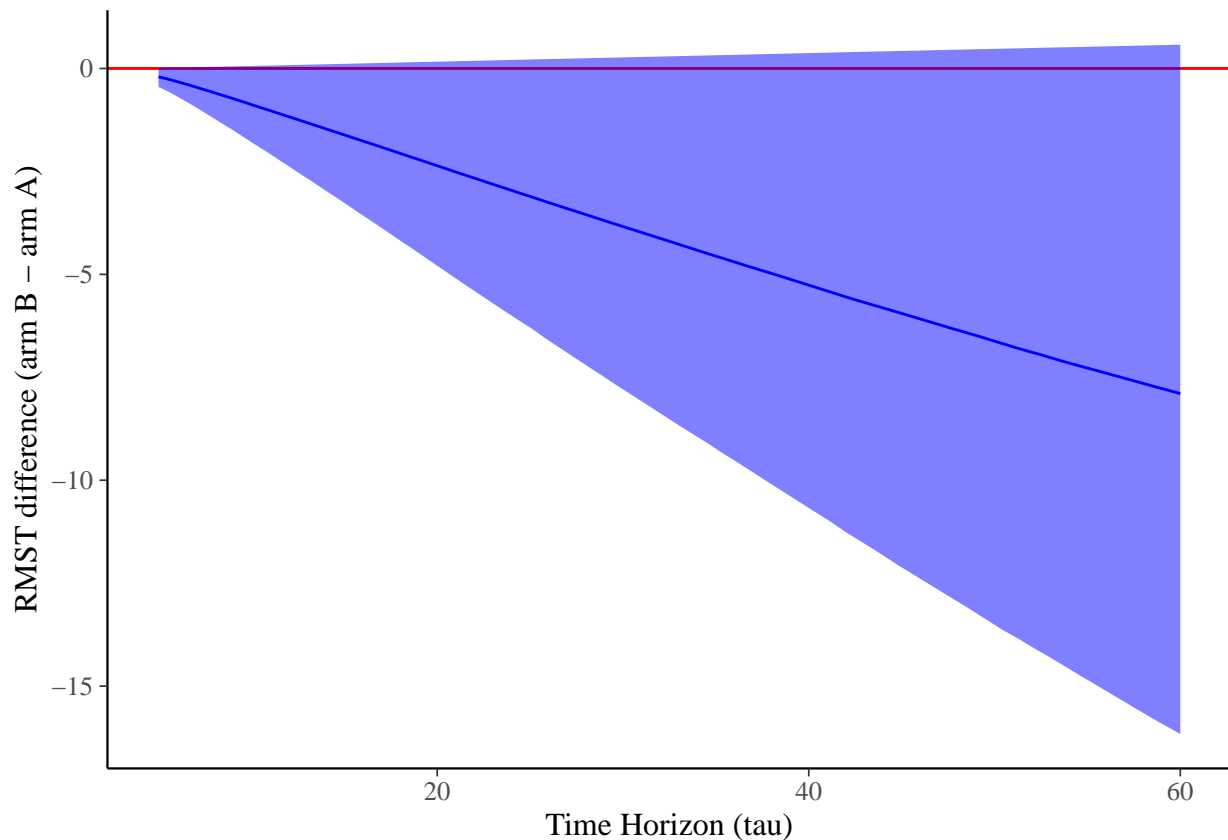
It is also interesting to consider how the (posterior) expected RMST treatment effect and its uncertainty (quantified here by a 95% credible interval) are affected by the time horizon  $\tau$ . For our best fitting model we evaluate the posterior estimates of the RMST treatment effect across a wide range of  $\tau$  values:

```
tau_values <- seq(5, 60, length.out = 200)

rmst_diffs <-
  tau_values %>%
  map_dfr(
    ~ rmst_check(fit.stan.ms10, .) %>%
      mutate(diff = rmstA - rmstB) %>%
      summarise(diff_median = quantile(diff, probs = 0.500),
                diff_low    = quantile(diff, probs = 0.025),
                diff_high    = quantile(diff, probs = 0.975)))

rmst_diffs_plot <-
  rmst_diffs %>%
  mutate(tau = tau_values) %>%
  ggplot(aes(x = tau, y = diff_median)) +
  geom_hline(yintercept = 0, color = "red") +
  geom_line(color = "blue") +
  geom_ribbon(aes(ymin = diff_low, ymax = diff_high), fill = "blue", alpha = 0.5) +
  xlab("Time Horizon (tau)") +
  ylab("RMST difference (arm B - arm A)")

rmst_diffs_plot
```



## 8 Brier score

The (time-dependent) Brier Score (BS) is a particularly useful for assessing the (time-dependent) calibration of survival models. For instance, when working with spline-based (or piecewise-constant) baseline hazard parametrisations, one can search for (temporal) periods with particularly large BS estimates and try to improve this by adding more knots at representative time points in the corresponding “suspicious” periods.

In addition to the beforementioned specific diagnostics used to construct posterior predictive checks, as well as the leave-one-out cross validation evaluation of the expected log posterior density, the BS provides a ubiquitous “global” diagnostic and hence extends our toolbox for scrutinizing and comparing survival models.

In order to avoid over optimistic estimates for the BS due to overfitting, we recommend a cross-validation scheme for calculating the BS. The Bayesian / sampling-based approach allows for a particular efficient method to estimate the leave-one-out BS. We show below how to calculate both the in-sample and loo BS and compare them graphically.

```
surv_mat <- function(fit) {  
  
  ps <- posterior_survfit(fit,  
                        times      = 0,  
                        extrapolate = TRUE,  
                        control     = list(edist = 75),  
                        draws      = 500)  
  
  ps_mat <- do.call(rbind,  
                  map(ps$id %>% unique(),  
                    ~ pull(filter(ps, id == .), "median")))  
  
  attr(ps_mat, "times") <- ps$time %>% unique()  
  
  return(ps_mat)  
}  
  
# Restrict the comparison to a subset of all models considered  
sel <- c("Constant", "MS (df = 10)", "MS (df = 5)")  
  
fits_stan_for_brier <- fits_stan[sel]  
  
surv_mats <- map(fits_stan_for_brier, surv_mat)  
  
times_pec <- attr(surv_mats[["Constant"]], "times")
```

To reduce the computational effort required to calculate the BS, we specify `draws=500` which implies that we only work with a subset of 500 posterior samples instead of the entire posterior sample.

We calculate the BS estimate using the `pec()` function from the `pec` R package (Gerds 2018):

```
pec_rslt <- pec(surv_mats,  
              formula = Surv(time, event) ~ arm,  
              data    = df,  
              times   = times_pec,  
              reference = FALSE,  
              exact    = FALSE,  
              cens.model = "cox",  
              splitMethod = "none")
```

The `formula` argument is used to construct the conditional censoring model required for the improved BS

estimator and to determine the censoring indicator variable (in our case `event`). By specifying `cens.model = "cox"` we tell `pec()` to use a (non-Bayesian) Cox regression model with predictor `arm` as the conditional censoring model, which will be used to calculate the weights in the BS estimator. By specifying `splitMethod = "none"` we tell `pec()` to estimate the BS via the same data as was used for obtaining the posterior (in-sample). Setting `reference = FALSE` suppresses the calculation of a non-parametric (Kaplan-Meier) reference prediction model. Lastly, by setting `exact = FALSE` we evaluate the BS at times that are specified via the `times` argument.

Next, we use Pareto Smoothed Importance Sampling (PSIS) to obtain an estimator for the leave-one-out version of the BS. This is done two steps.

First, we estimate the matrix  $\{S_i(t|\mathbf{T}_{-i})\}_{i,t}$  by using the `get_loo_mean_surv()` function. This uses methods implemented in the `loo` package, in particular the `E_loo` function, to do the PSIS importance sampling. For details see the function definition of `get_loo_mean_surv()` that we defined (internally) at the start of this notebook. See the R Markdown source file for this notebook to see the definition of the `get_loo_mean_surv()` function. However, in contrast to the earlier estimation that used only 500 posterior samples, here we use all posterior samples to obtain an estimate for the posterior *mean* of  $\{S_i(t|\mathbf{T}_{-i})\}_{i,t}$ :

```
surv_mats_loo <- map(fits_stan_for_brier, get_loo_mean_surv, times_pec)
```

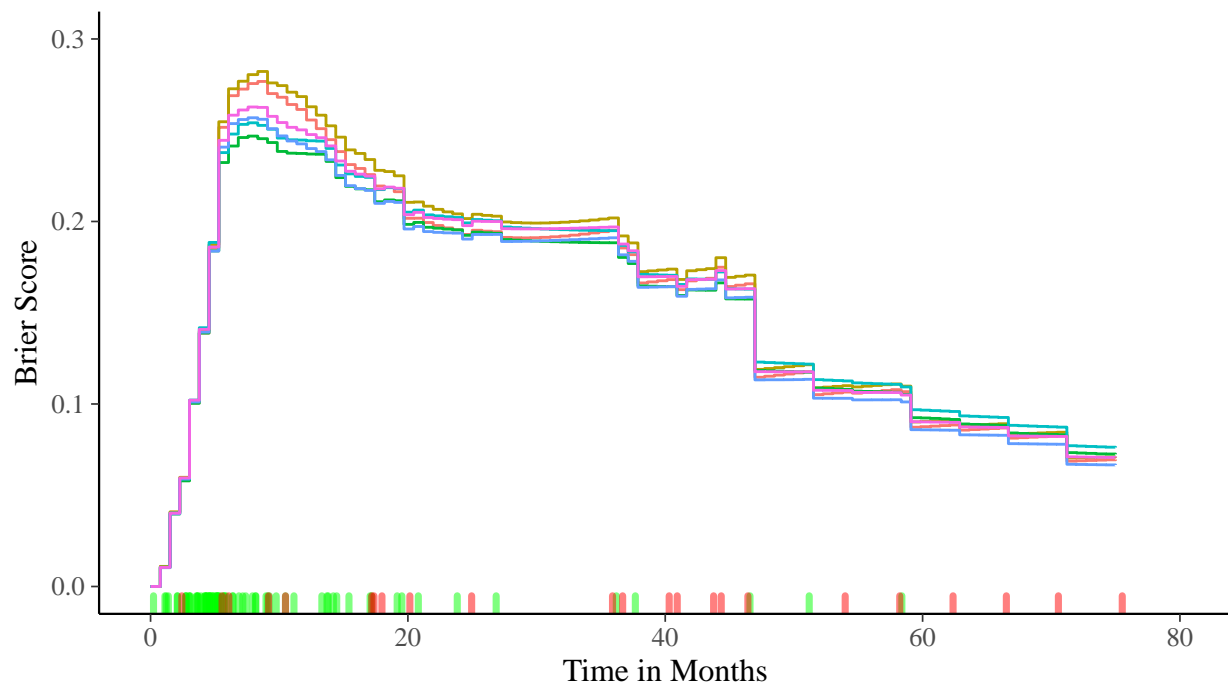
Second, we use the `pec()` function to calculate the BS estimator for the (loo) survival matrices:

```
pec_rslt_loo <- pec(surv_mats_loo,
  formula      = Surv(time, event) ~ arm,
  data         = df,
  times        = times_pec,
  reference    = FALSE,
  exact        = FALSE,
  cens.model   = "cox",
  splitMethod  = "none")
```

We can then plot the time-dependence of the BS estimates over all models considered:

```
map_dfr(names(fits_stan_for_brier),
  ~ bind_rows(tibble(time = pec_rslt$time,
    bs = pec_rslt$AppErr[[]],
    Model = .),
    tibble(time = pec_rslt_loo$time,
      bs = pec_rslt_loo$AppErr[[]],
      Model = paste(., "[Loo]")))) %>%

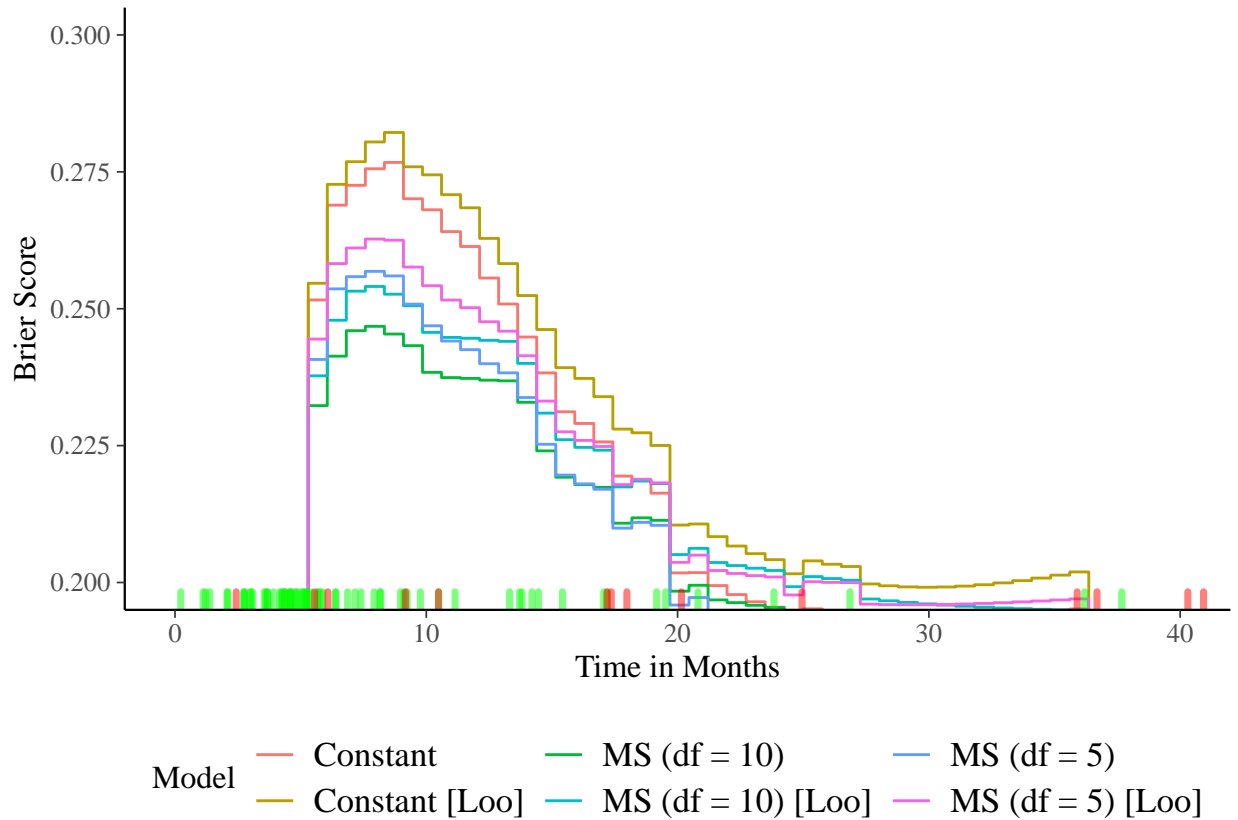
ggplot() +
  geom_step(aes(x = time, y = bs, color = Model)) +
  geom_rug(data = df %>% filter(event==1) %>% select(time),
    mapping = aes(x = time),
    color = "green",
    alpha = 0.5,
    size = 1.25) +
  geom_rug(data = df %>% filter(event==0) %>% select(time),
    mapping = aes(x = time),
    color = "red",
    alpha = 0.5,
    size = 1.25) +
  coord_cartesian(xlim = c(0, 80), ylim = c(0, 0.3)) +
  ylab("Brier Score") +
  xlab("Time in Months") +
  theme(legend.position = "bottom")
```



Model    — Constant    — MS (df = 10)    — MS (df = 5)  
           — Constant [Loo]    — MS (df = 10) [Loo]    — MS (df = 5) [Loo]

The green and red rugs above the horizontal axis are placed at times with uncensored and right-censored observations, respectively. The differences between the curves become more apparent when zoom in somewhat:

```
last_plot() + coord_cartesian(ylim = c(0.2, 0.3), xlim=c(0, 40))
```



We can make several observations from the plots of the BS estimates:

- Firstly, there is a clear difference between the loo and in-sample BS estimates – especially during the period between 5 and 20 months. The BS is larger when evaluated using loo.
- Secondly, between 5 and 20 months there is a clear difference between the models in terms of the BS estimates. Specifically, the constant model stands out as being the worst (higher BS) and the cubic M-spline model with 10 degrees of freedom stands out as being the best (lower BS).
- Thirdly, the added model complexity due to increasing the degrees of freedom for the M-splines from 5 to 10 (hence adding 5 additional interior knots) appears justified. This is because the  $df = 10$  model has a smaller BS (less prediction error) compared to the  $df = 5$  model. A closer inspection of the corresponding standardised survival curves shows that the implied placement of additional knots indeed helps to improve the fit of the survival curve.
- Lastly, and not surprisingly, our models are not particularly informative, as can be seen by the large BS-values (around 0.20 – 0.25).

## 9 Summary

Through a worked case study this notebook has provided an introduction to Bayesian survival analysis using the **rstanarm** R package. We hope that the new survival analysis functionality in **rstanarm** helps to make Bayesian survival analysis more accessible to applied researchers and analysts. As part of this notebook we have introduced some (but not all) of the methodology underpinning the software. Some of the features we haven't discussed include interval censoring, delayed entry, time-varying effects (e.g. non-proportional hazards), and accelerated failure time (AFT) models.

Our case study was based on clinical data from a randomized controlled trial in patients with cancer. Survival analysis is widely used in medical and health research, however this is not the only area in which it is relevant. Survival analysis methodology is applicable in any discipline where event times are observed and analysed.

## R Session info

```
sessionInfo()
```

```
## R version 3.6.0 (2019-04-26)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: macOS Mojave 10.14.6
##
## Matrix products: default
## BLAS: /Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats      graphics  grDevices  utils      datasets  methods   base
##
## other attached packages:
## [1] pec_2018.07.26      prodlim_2018.04.18 loo_2.1.0
## [4] rstanarm_2.18.2     Rcpp_1.0.1         survival_2.44-1.1
## [7] bayesplot_1.7.0     cowplot_0.9.4      forcats_0.4.0
## [10] stringr_1.4.0       dplyr_0.8.2        purrr_0.3.2
## [13] readr_1.3.1         tidyr_0.8.3        tibble_2.1.3
## [16] ggplot2_3.2.0       tidyverse_1.2.1    knitr_1.23
##
## loaded via a namespace (and not attached):
## [1] readxl_1.3.1        backports_1.1.4     Hmisc_4.2-0
## [4] plyr_1.8.4          igraph_1.2.4.1      lazyeval_0.2.2
## [7] splines_3.6.0       crosstalk_1.0.0     TH.data_1.0-10
## [10] rstantools_1.5.1    inline_0.3.15       digest_0.6.19
## [13] foreach_1.4.4       htmltools_0.3.6     rsconnect_0.8.13
## [16] fansi_0.4.0         magrittr_1.5         checkmate_1.9.3
## [19] cluster_2.0.8       modelr_0.1.4         matrixStats_0.54.0
## [22] sandwich_2.5-1      xts_0.11-2          prettyunits_1.0.2
## [25] colorspace_1.4-1    rvest_0.3.4         haven_2.1.0
## [28] xfun_0.8            callr_3.2.0         crayon_1.3.4
## [31] jsonlite_1.6        lme4_1.1-21         zeallot_0.1.0
## [34] zoo_1.8-6           iterators_1.0.10     glue_1.3.1
## [37] gtable_0.3.0        MatrixModels_0.4-1  pkgbuild_1.0.3
## [40] rstan_2.18.2        rms_5.1-3.1         SparseM_1.77
## [43] scales_1.0.0        mvtnorm_1.0-11      miniUI_0.1.1.1
## [46] xtable_1.8-4        htmlTable_1.13.1    foreign_0.8-71
## [49] Formula_1.2-3       stats4_3.6.0        splines2_0.2.8
## [52] lava_1.6.5          StanHeaders_2.18.1-10 DT_0.7
## [55] htmlwidgets_1.3     httr_1.4.0          threejs_0.3.1
## [58] RColorBrewer_1.1-2  acepack_1.4.1        pkgconfig_2.0.2
## [61] nnet_7.3-12         utf8_1.1.4          tidyselect_0.2.5
## [64] labeling_0.3        rlang_0.4.0         reshape2_1.4.3
## [67] later_0.8.0         munsell_0.5.0       cellranger_1.1.0
## [70] tools_3.6.0         cli_1.1.0           generics_0.0.2
## [73] broom_0.5.2         ggribges_0.5.1      evaluate_0.14
## [76] yaml_2.2.0          processx_3.4.0      timereg_1.9.3
## [79] packrat_0.5.0       nlme_3.1-139        quantreg_5.41
```

## [82]	mime_0.7	xml2_1.2.0	compiler_3.6.0
## [85]	shinythemes_1.1.2	rstudioapi_0.10	survRM2_1.0-2
## [88]	stringi_1.4.3	highr_0.8	ps_1.3.0
## [91]	lattice_0.20-38	Matrix_1.2-17	nloptr_1.2.1
## [94]	markdown_1.0	shinyjs_1.0	vctrs_0.1.0
## [97]	pillar_1.4.2	data.table_1.12.2	httpuv_1.5.1
## [100]	R6_2.4.0	latticeExtra_0.6-28	promises_1.0.1
## [103]	gridExtra_2.3	codetools_0.2-16	polyspline_1.1.15
## [106]	boot_1.3-22	colourpicker_1.0	MASS_7.3-51.4
## [109]	gtools_3.8.1	assertthat_0.2.1	withr_2.1.2
## [112]	shinystan_2.5.0	multcomp_1.4-10	parallel_3.6.0
## [115]	hms_0.4.2	grid_3.6.0	rpart_4.1-15
## [118]	minqa_1.2.4	rmarkdown_1.13	numDeriv_2016.8-1.1
## [121]	shiny_1.3.2	lubridate_1.7.4	base64enc_0.1-3
## [124]	dygraphs_1.1.1.6		

## References

- Carpenter, Bob, Andrew Gelman, Matthew D. Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. 2017. “Stan: A Probabilistic Programming Language.” *Journal of Statistical Software* 76 (1). <https://doi.org/10.18637/jss.v076.i01>.
- Cupples, L. A., D. R. Gagnon, R. Ramaswamy, and R. B. D’Agostino. 1995. “Age-adjusted Survival Curves with Application in the Framingham Study.” *Statistics in Medicine* 14 (16): 1731–44.
- Fu, K K, T L Phillips, I J Silverberg, C Jacobs, D R Goffinet, C Chun, M A Friedman, M Kohler, K McWhirter, and S K Carter. 1987. “Combined Radiotherapy and Chemotherapy with Bleomycin and Methotrexate for Advanced Inoperable Head and Neck Cancer: Update of a Northern California Oncology Group Randomized Trial.” *Journal of Clinical Oncology* 5 (9): 1410–8. <https://doi.org/10.1200/JCO.1987.5.9.1410>.
- Gerds, Thomas A. 2018. *Pec: Prediction Error Curves for Risk Prediction Models in Survival Analysis*. <https://cran.r-project.org/web/packages/pec/index.html>.
- Gerds, Thomas A, and Martin Schumacher. 2006. “Consistent Estimation of the Expected Brier Score in General Survival Models with Right-Censored Event Times.” *Biometrical Journal* 48 (6): 1029–40.
- Goodrich, Ben, Jonah Gabry, Imad Ali, and Sam Brilleman. 2018. *Rstanarm: Bayesian Applied Regression Modeling via Stan*. <http://mc-stan.org/>.
- Ramsay, J. O. 1988. “Monotone Regression Splines in Action.” *Statistical Science* 3 (4): 425–41. <https://doi.org/10.1214/ss/1177012761>.
- Royston, Patrick, and Mahesh KB Parmar. 2013. “Restricted Mean Survival Time: An Alternative to the Hazard Ratio for the Design and Analysis of Randomized Trials with a Time-to-Event Outcome.” *BMC Medical Research Methodology* 13 (1): 152.
- Sinha, D., J. G. Ibrahim, and M. Chen. 2003. “A Bayesian Justification of Cox’s Partial Likelihood.” *Biometrika* 90 (3): 629–41. <https://doi.org/10.1093/biomet/90.3.629>.
- Therneau, Terry M. 2019. *Survival: Survival Analysis*. <https://CRAN.R-project.org/package=survival>.
- Uno, Hajime, Lu Tian, Angel Cronin, Chakib Battioui, and Miki Horiguchi. 2017. *SurvRM2: Comparing Restricted Mean Survival Time*. <https://CRAN.R-project.org/package=survRM2>.
- Vehtari, Aki, Andrew Gelman, and Jonah Gabry. 2019. *Loo: Efficient Leave-One-Out Cross-Validation and Waic for Bayesian Models*. <https://cran.r-project.org/web/packages/loo/index.html>.
- Wang, Wenjie, and Jun Yan. 2018. *Splines2: Regression Spline Functions and Classes*. <https://CRAN.R-project.org/package=splines2>.



Zhou, Haiming, Timothy Hanson, and Jiajia Zhang. 2018. “SpBayesSurv: Fitting Bayesian Spatial Survival Models Using R.” *Journal of Statistical Software* to appear.