Received: 11 November 2019 | Revised: 18 January 2021 | Accepted: 13 February 2021

DOI: 10.1002/sim.8933

TUTORIAL IN BIOSTATISTICS

Statistics in Medicine WILEY

Bayesian survival analysis with BUGS

Danilo Alvares[1] | Elena Lázaro[2] | Virgilio Gómez-Rubio[3] | Carmen Armero[4]

[1]Department of Statistics, Pontificia Universidad Católica de Chile, Santiago, Chile

[2]Plant Protection and Biotechnology Centre, Instituto Valenciano de Investigaciones Agrarias, Valencia, Spain

[3]Department of Mathematics, School of Industrial Engineering-Albacete, Universidad de Castilla-La Mancha, Ciudad Real, Spain

[4]Department of Statistics and Operational Research, Universitat de València, Valencia, Spain

Correspondence
Danilo Alvares, Department of Statistics, Pontificia Universidad Católica de Chile, Santiago 7820436, Chile.
Email: dalvares@mat.uc.cl

Funding information
Consejería de Educación, Cultura y Deportes (JCCM, Spain), Grant/Award Number: PPIC-2014-001-P; FEDER, Grant/Award Number: SBPLY/17/180501/000491; Fondo Nacional de Desarrollo Científico y Tecnológico, Grant/Award Number: 11190018; Ministerio de Ciencia e Innovación, Grant/Award Number: PID2019-106341GB-I00

Survival analysis is one of the most important fields of statistics in medicine and biological sciences. In addition, the computational advances in the last decades have favored the use of Bayesian methods in this context, providing a flexible and powerful alternative to the traditional frequentist approach. The objective of this article is to summarize some of the most popular Bayesian survival models, such as accelerated failure time, proportional hazards, mixture cure, competing risks, multi-state, frailty, and joint models of longitudinal and survival data. Moreover, an implementation of each presented model is provided using a BUGS syntax that can be run with JAGS from the R programming language. Reference to other Bayesian R-packages is also discussed.

KEYWORDS
Bayesian inference, JAGS, R-packages, time-to-event analysis

1 | INTRODUCTION

Survival analysis, sometimes referred to as *failure-time analysis*, is one of the most important fields of statistics, mainly in medicine and biological sciences.[1,2] Survival times are data that measure follow-up time from a defined starting point to the occurrence of a given event of interest or endpoint, for instance, onset of disease, cure, death, and so on.[3]

Continuous survival times are defined as non-negative random variables, T, whose probabilistic behavior can be equivalently described by its hazard function, *survival function*, or *density function*. The hazard function of T at time t represents the instantaneous rate of the event occurrence for the population group that is still at risk at time t. It is defined as

$$h(t|\boldsymbol{\theta}) = \lim_{\Delta t \to 0} \frac{P(t \leq T < t + \Delta t | T \geq t, \boldsymbol{\theta})}{\Delta t} = \frac{f(t|\boldsymbol{\theta})}{S(t|\boldsymbol{\theta})}, \quad t > 0, \tag{1}$$

where $\boldsymbol{\theta}$ is a set of unknown quantities, $f(t|\boldsymbol{\theta})$ is the density function of T given $\boldsymbol{\theta}$, and $S(t|\boldsymbol{\theta}) = P(T > t|\boldsymbol{\theta})$ is the survival function of T given $\boldsymbol{\theta}$. Note that $h(t|\boldsymbol{\theta}) \geq 0$ and in general $\int h(t|\boldsymbol{\theta})\, \mathrm{d}t = \infty$. The hazard function is defined in terms of a conditional probability but is not a probability. It could be interpreted as the approximate instantaneous probability of having the event given that the target individual has not yet experienced it at time t.

Usual relationships between $h(t|\boldsymbol{\theta}), f(t|\boldsymbol{\theta})$, and $S(t|\boldsymbol{\theta})$ derived from (1) that will be useful throughout this article include spimergestart

$$H(t|\boldsymbol{\theta}) = \int_0^t h(u|\boldsymbol{\theta})\, \mathrm{d}u, \quad t > 0, \tag{2}$$

$$S(t|\boldsymbol{\theta}) = \exp\{-H(t|\boldsymbol{\theta})\}, \quad t > 0, \tag{3}$$

$$f(t|\boldsymbol{\theta}) = h(t|\boldsymbol{\theta})\, \exp\{-H(t|\boldsymbol{\theta})\}, \quad t > 0, \tag{4}$$

spimergeend where $H(t|\boldsymbol{\theta})$ denotes the *cumulative hazard function* of T.

Standard statistical techniques cannot usually be applied to survival data because in most cases they are not fully recorded, mainly due to censoring and/or truncation schemes.[4] Also, normal distributions are usually inappropriate for analyzing survival data at their original scale, since they are positive and often asymmetrical. Distributions such as the Weibull, gamma, or log-normal are the ones that now occupy the leading role.

Censoring mechanisms affect the likelihood function, which factorizes in general as the product of the likelihood function corresponding to each individual i in the sample as $L(\boldsymbol{\theta}) = \prod_{i=1}^n L_i(\boldsymbol{\theta})$. In general terms, censored observations can be classified as *right-censored*, *left-censored*, and *interval-censored*. See Klein and Moeschberger[5] for technical details, interpretations, and examples. The contribution of the survival observation t of individual i to the likelihood function, $L_i(\boldsymbol{\theta})$, depends on the type of censoring as follows:

Exact survival time:	$f_i(t\|\boldsymbol{\theta})$
Right-censored observation:	$S_i(C_r\|\boldsymbol{\theta})$
Left-censored observation:	$1 - S_i(C_l\|\boldsymbol{\theta})$
Interval-censored observation:	$S_i(C_l\|\boldsymbol{\theta}) - S_i(C_r\|\boldsymbol{\theta})$

Right censoring here is type I, where the end of study period C_r is known and prefixed before it begins and/or random dropout is allowed. In the case of left-censored observations, the survival time is known to be below a certain censoring time C_l. When survival times are interval-censored in $[C_l, C_r]$, the survival time is somewhere in this interval.

Due to these peculiar characteristics and their extreme relevance to different scientific subjects, survival models/methods have been extensively developed over the past 50 years, which includes many R-packages.[*]

In this article, the inference is performed from the Bayesian methodology, which is an appealing alternative to the traditional frequentist approach, since the Bayesian probability conception allows to measure the uncertainty associated with parameters, models, hypotheses, latent variables, and missing data in probabilistic terms. We can also highlight that the Bayesian approach incorporates prior knowledge in a natural way, whereas the frequentist approach does not. For example, when we have historical clinical trials data, we can use prior distributions to formally incorporate this information into the current analysis.[6,7] Another potential limitation of the frequentist paradigm is the calculation of variance estimates, which can be complicated to derive (or even impracticable) since it is based on asymptotic arguments and therefore require large sample sizes. In contrast, Bayesian variance estimates, as well as any other quantity of interest based on the Bayesian sampler, are trivial to obtain once samples from the posterior distribution are available.[8] For those interested in comparative performance between both methodologies, several authors[9-12] have contrasted frequentist and Bayesian approaches for survival models.

There are a number of R-packages to fit different types of Bayesian survival models which can be used in specific contexts. For example, if one is interested in non-standard accelerated failure time (AFT) modeling, the `bayesSurv`[13] package provides implementations of mixed-effects AFT models with various censored data specifications; while the `spBayesSurv`[14] package includes AFT and proportional hazards (PH) models (among others) for spatial/non-spatial

[*]https://cran.r-project.org/web/views/Survival.html

survival data. There are also many packages that fit Cox PH models, we highlight some non-standard ones like the `dynsurv`[15] that provides time-varying coefficient models for interval-censored and right-censored survival data; the `ICBayes`[16] that offers semi-parametric regression survival models to interval-censored time-to-event data; and the `spatsurv`[17] that fits parametric PH spatial survival models. For those that want to do Bayesian model averaging, the `BMA`[18] package implements this approach for Cox PH models. For multi-state problems, we give emphasis to the `CFC`[19] package that implements a cause-specific framework for competing-risk analysis; and `SemiCompRisks`[20] package that provides a broader specification using of independent/clustered semi-competing risks models.[21] When a joint approach of longitudinal and survival data is required, the most popular package is the `JMbayes`[22] one, where the longitudinal process is modeled through a linear mixed framework, a Cox PH model is assumed for the survival process, and various association structures between both processes are provided. Other generic packages for Bayesian inference based on standalone programs such as, for example, `BayesX`,[23] `RStan`,[24,25] `rstanarm`,[26] or `INLA`[27] can be used to fit different models.

This article is intended as a guide for those readers who, having a solid background in Bayesian statistics, want to enter into learning the Bayesian treatment of survival scenarios. It may also be of interest to those experts in survival analysis who want to implement standard survival models from the Bayesian perspective. These objectives determine the structure of this article, whose sections are organized around each survival model introduced. Sections 2 to 7, respectively, describe AFT, PH, mixture cure, competing risks, multi-state, frailty, and joint models of longitudinal and survival data. These sections have a common logical scheme that aims to facilitate a relatively independent reading of each one of them. In each one of these sections, we briefly introduce notation and basic concepts of the models under discussion. Then, the survival dataset used (available in an R-package) is described and each survival modeling is exemplified using Markov chain Monte Carlo (MCMC) methods[28] implemented in BUGS syntax[29] with the support of JAGS[30] and the `rjags`[31] package (version 4-10) for the R language (version 4.0.2).[32] Finally, posterior summaries, and graphs of quantities of interest derived from the posterior distribution are provided. In particular, in Sections 2.1 (first example), we present in detail the use of auxiliary functions for graphical summaries of the model parameters as well as a MCMC convergence diagnostic. To conclude, Section 8 ends with an overview of the Bayesian survival models introduced and implemented in this article, motivating the use and adaptation of the codes provided. Theoretical and methodological aspects of survival models and Bayesian inference are not discussed in depth in this article, so we recommend that readers unfamiliar with these topics review specific references, such as Klein et al[4] and Gelman et al.[28] Although the examples presented in this article use the R environment, these models can also be fit using other programming languages with interfaces to JAGS, such as for example Python,[33] STATA,[34] Matlab,[35] and JASP.[36]

2 | SURVIVAL REGRESSION MODELS

Regression models focus on the association between survival elements (defined in terms of time-to-event random variables, hazard rates, etc.) and covariates (explanatory/predictor variables or risk factors). They allow the comparison of survival times between groups both with regard to the general behavior of the individuals in each group and prediction for new members of them. The most popular approaches to survival regression models are the AFT and the PH models, also known as Cox models.[37]

2.1 | Accelerated failure time models

AFT models are the survival counterpart of linear models. Survival time T in logarithmic scale is expressed in terms of a linear combination of covariates $\boldsymbol{x}$ with regression coefficients $\boldsymbol{\beta}$ and a measurement error ξ as follows:

$$\log(T) = \boldsymbol{x}^\top \boldsymbol{\beta} + \sigma\,\epsilon, \tag{5}$$

where σ is a scale parameter and ϵ is an error term usually expressed via a normal, logistic, or a standard Gumbel probabilistic distribution. The particular case of a standard Gumbel distribution for ϵ implies a conditional (on $\boldsymbol{\beta}$ and σ) Weibull survival model for T with shape $\alpha = 1/\sigma$ and scale $\lambda(\boldsymbol{\beta}, \sigma) = \exp\left\{-\boldsymbol{x}^\top \boldsymbol{\beta}/\sigma\right\}$ parameters.[38]

2.1.1 | *larynx* dataset

We consider a larynx cancer dataset, referred to as *larynx*. It is available from the `KMsurv` package:[39]

```
R> library("KMsurv")
R> data("larynx")
R> str(larynx)
'data.frame': 90 obs. of 5 variables:
 $ stage : int 1 1 1 1 1 1 1 1 1 1 ...
 $ time : num 0.6 1.3 2.4 2.5 3.2 3.2 3.3 3.3 3.5 3.5 ...
 $ age : int 77 53 45 57 58 51 76 63 43 60 ...
 $ diagyr: int 76 71 71 78 74 77 74 77 71 73 ...
 $ delta : int 1 1 1 0 1 0 1 0 1 1 ...
```

This dataset provides observations of 90 male larynx-cancer patients, diagnosed and treated in the period 1970 to 1978.[40] The following variables were observed for each patient:

- `stage`: disease stage (1 to 4).
- `time`: time (in months) from first treatment until death, or end of study.
- `age`: age (in years) at diagnosis of larynx cancer.
- `diagyr`: year of diagnosis of larynx cancer.
- `delta`: death indicator (1: if patient died; 0: otherwise).

2.1.2 | Model specification

Survival times are analyzed through the following AFT model:

$$\log(T) = \beta_1 + \beta_2 \mathrm{I}_{(\texttt{stage}=2)} + \beta_3 \mathrm{I}_{(\texttt{stage}=3)} + \beta_4 \mathrm{I}_{(\texttt{stage}=4)} + \beta_5 \texttt{age} + \beta_6 \texttt{diagyr} + \sigma\epsilon, \tag{6}$$

where T represents death time for each individual; β_1 is an intercept; $\mathrm{I}_{(\texttt{stage}=\cdot)}$ is an indicator variable for `stage`=2, 3, 4 with regression coefficients β_2, β_3, and β_4, respectively (`stage`=1 is considered as the reference category); and β_5 and β_6 are regression coefficients for `age` and `diagyr` covariates, respectively. The errors ϵ's are i.i.d. random variables which follow a standard Gumbel distribution and σ is a scale parameter.

The Bayesian model is completed with the specification of a prior distribution for their corresponding parameters. A non-informative prior independent default scenario is considered. The marginal prior distribution for each regression coefficient β_k, $k = 1, \dots, 6$, is elicited as a normal distribution centered at zero and with a small precision, N(0,0.001). A uniform distribution, Un(0,10), is selected as the marginal prior distribution for α (ie, the Weibull shape parameter). However, the use of another continuous distributions, for example, Gamma(0.01,0.01) is also common under a non-informative prior framework.[41,42]

2.1.3 | Model implementation

Censoring is handled in JAGS via the function `dinterval` and the specification of an auxiliary ordinal variable `is.censored` $\in \{0, \dots, M\}$ together with cutpoints $\mathrm{cen}_1 < \dots < \mathrm{cen}_M$. Assuming the vector of cutpoints $\mathrm{cen} = (\mathrm{cen}_1, \dots, \mathrm{cen}_M)^\top$, associated to the `is.censored` value, the statement `is.censored` $\sim$ `dinterval(time,cen)` specifies the following for the censored variable `time`:[12,43]

$$\begin{array}{lll}
\texttt{is.censored} = 0: & \texttt{time} \le \mathrm{cen}_1 & \\
\texttt{is.censored} = m: & \mathrm{cen}_m < \texttt{time} \le \mathrm{cen}_{m+1}, & m = 1, \dots, M-1 \\
\texttt{is.censored} = M: & \texttt{time} > \mathrm{cen}_M &
\end{array}$$

In principle, `dinterval` is a distribution to represent general interval-censored data,[43] but it can be used to manage jointly the specification of other types of censoring as well as uncensored observations as follows:

- For right-censored observations, `is.censored` should be set at 1, `time` as NA and only one cut point should be specified with the right-censored time as the lower limit of the interval in the cut point vector **cen** (ie, $\mathbf{cen} = (\text{cen}_1, \text{NA})^\top$).
- For left-censored observations, `is.censored` should be set at 0, `time` as NA and only one cut point should be specified with the left-censored time as the upper limit of the interval in the cut point vector **cen** (ie, $\mathbf{cen} = (\text{NA}, \text{cen}_2)^\top$).
- For interval-censored observations, `is.censored` should be set at 1, `time` as NA and two cut points should be specified with the interval-censored times as the lower and the upper limit of the interval in the cut point vector **cen** (ie, $\mathbf{cen} = (\text{cen}_1, \text{cen}_2)^\top$).
- For uncensored observations, `is.censored` should be set at 0, `time` as the observed time and only one cut point should be specified with the observed time as the lower limit of the interval in the cut point vector **cen** (ie, $\mathbf{cen} = (\text{time}, \text{NA})^\top$).

Listing 1 shows a generic implementation in JAGS on how to specify jointly right-censored, left-censored, interval-censored and uncensored observations. Truncation, although is not covered in this article, is represented in JAGS using the `T(,)` construct on the right hand side of a stochastic relation. For example, a late entry at time 10 in a Weibull model can be specified as: `time ~ dweib(alpha, lambda)T(10,)`.[44] Note that censoring and truncation is addressed differently in JAGS than in WinBUGS or OpenBUGS.[31]

```
# Illustrative data
list(time = c(NA,NA,NA,1), is.censored = c(1,0,1,0), cen1 = c(1.5,NA,0.5,1),
     cen2 = c(NA,0.5,1.2,NA))
model {
  # Right-censored observation: is.censored[1] = 1
  is.censored[1] ~ dinterval(time[1], cen1[1])
  # Left-censored observation: is.censored[2] = 0
  is.censored[2] ~ dinterval(time[2], cen2[2])
  # Interval-censored observation: is.censored[3] = 1
  is.censored[3] ~ dinterval(time[3], c(cen1[3],cen2[3]))
  # Uncensored observation: is.censored[4] = 0
  is.censored[4] ~ dinterval(time[4], cen1[4])
}
```

Listing 1: Censored observations specification in JAGS syntax

Thus, for the `larynx` dataset which contains uncensored and right-censored observations the following code is dedicated to manage the creation and specification of `is.censored`, `time`, and `cen` variables as well as the design matrix `X` of the covariates `age`, `diagyr`, and `stage`. Note that we have scaled the `age` and `diagyr`, and encoded the `stage` as a factor.

```
[commandchars=\\\{\}]
R> # Covariates
R> larynx$age <- as.numeric(scale(larynx$age))
R> larynx$diagyr <- as.numeric(scale(larynx$diagyr))
R> larynx$stage <- as.factor(larynx$stage)
R> X <- model.matrix(~ stage + age + diagyr, data = larynx)

R> # Uncensored observations
R> time_{u}n <- larynx$time[un]
```

```
R> cen_un <- larynx$time[un]
R> is.censored_{u}n <- rep(0, length(cen_{u}n))
R> X_{u}n <- X[un, ]

R> # Right-censored observations
R> cen_{c}en <- larynx$time[-un]
R> time_cen <- rep(NA, length(cen_cen))
R> is.censored_cen <- rep(1, length(cen_cen))
R> X_cen <- X[-un, ]

R> # Uncensored and right-censored observations
R> time <- c(time_un, time_cen)
R> cen <- c(cen_un, cen_cen)
R> is.censored <- c(is.censored_un, is.censored_cen)
R> Xnew <- rbind(X_un, X_cen)
```

Listing 2 shows a generic implementation of an AFT model in BUGS syntax using *larynx* data.

```
model {
# Uncensored and right−censored observations
  for(i in 1:n) {
    is.censored[i] ~ dinterval(time[i], cen[i])
    time[i] ~ dweib(alpha, lambda[i])
    lambda[i] <− exp(−mu[i] ∗ alpha)
    mu[i] <− inprod(beta[], X[i, ])
  }
  # Prior distributions
  for(l in 1:Nbetas) {
    beta[l] ~ dnorm(0, 0.001)
  }
  alpha ~ dunif(0, 10)
}
```

Listing 2: AFT model in BUGS syntax (file named as **AFT.txt**)

Once the BUGS syntax and its corresponding variables has been created, JAGS requires specifying some elements to run the MCMC simulation:

- `d.jags`: a list with all the elements/data specified in the model.
- `i.jags`: a function that returns a list of initial random values for each model parameters.
- `p.jags`: a character vector with the parameter names to be monitored/saved.

These elements are defined for our AFT model as follows:

```
[commandchars=\\\{\}]
R> d.jags <- list(n = nrow(X), time = time, cen = cen, X = Xnew, is.censored =
is.censored,
+ Nbetas = ncol(X))
R> i.jags <- function() list(beta = rnorm(ncol(X)), alpha = runif(1))
R> p.jags <- c("beta", "alpha")
```

Then, MCMC algorithm is run in three steps. First, the JAGS model is compiled by means of the `jags.model` function available from the `rjags` package:

```
R> library("rjags")
R> m1 <- jags.model(data = d.jags, file = "AFT.txt", inits = i.jags, n.chains = 3)
```

The `n.chains` argument indicates the number of Markov chains selected. Second, a burn-in period is considered (here the first 1000 simulations) using the `update` function:

```
R> update(m1, 1000)
```

Thirdly, the model is run using `coda.samples` function for a specific number of iterations to monitor (here `n.iter`=50000) as well as a specific thinning value (here `thin`=10) in order to reduce autocorrelation in the saved samples:

```
R> res <- coda.samples(m1, variable.names = p.jags, n.iter = 50000, thin = 10)
```

A posterior distributions summary can be obtained through the `summary` function:

```
R> summary(res)
Iterations = 7010:57000
Thinning interval = 10
Number of chains = 3
Sample size per chain = 5000

1. Empirical mean and standard deviation for each variable,
 plus standard error of the mean:
 Mean SD Naive SE Time-series SE
alpha 1.03246 0.1370 0.001118 0.001382
beta[1] 2.55478 0.2983 0.002436 0.003947
beta[2] -0.12645 0.4725 0.003858 0.004794
beta[3] -0.64608 0.3659 0.002987 0.004099
beta[4] -1.65929 0.4449 0.003633 0.005046
beta[5] -0.21062 0.1581 0.001291 0.001352
beta[6] 0.07108 0.1627 0.001329 0.001491

2. Quantiles for each variable:
 2.5
alpha 0.7815 0.9358 1.02815 1.1211 1.31636
beta[1] 2.0412 2.3484 2.52976 2.7370 3.21216
beta[2] -1.0211 -0.4474 -0.14098 0.1794 0.84315
beta[3] -1.4101 -0.8750 -0.63335 -0.4058 0.04360
beta[4] -2.5599 -1.9465 -1.64556 -1.3544 -0.81834
beta[5] -0.5382 -0.3107 -0.20500 -0.1047 0.08638
beta[6] -0.2313 -0.0417 0.06641 0.1764 0.40063
```

Markov chains must reach stationarity and, to check this condition, several convergence diagnostics can be applied. Trace plots (`traceplot` function) and the calculation of the Gelman-Rubin diagnostic (`gelman.diag` function) are used for illustrating this issue[45,46] with the `coda` package,[47] which is already loaded as `rjags` depends on it:

```
R> par(mfrow = c(2,4))
R> traceplot(res)
```

```
R> gelman.diag(res)
Potential scale reduction factors:
```

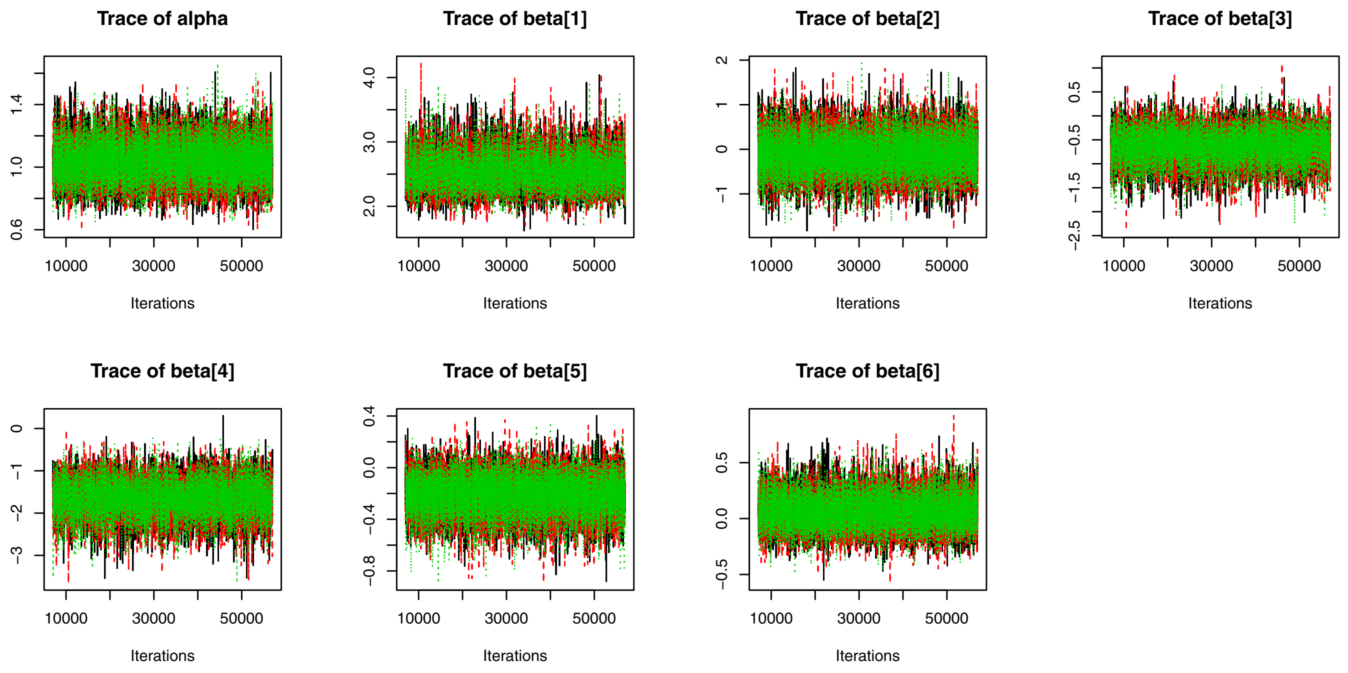

FIGURE 1 Trace plots for α and β's of the AFT model (6) [Colour figure can be viewed at wileyonlinelibrary.com]

```
 Point est. Upper C.I.
alpha 1 1.01
beta[1] 1 1.00
beta[2] 1 1.00
beta[3] 1 1.00
beta[4] 1 1.00
beta[5] 1 1.00
beta[6] 1 1.00

Multivariate psrf
1
```

Trace plots (see Figure 1) of the sampled values for each parameter in the chain appear overlapping one another and Gelman-Rubin values are very close to 1, which indicates that convergence has been achieved.

From the `densplot` function, also available from the `coda` package, we can draw the marginal posterior distributions (using kernel smoothing) for all the model parameters (see Figure 2):

```
R> par(mfrow = c(2,4))
R> densplot(res, xlab = "")
```

Simulation-based Bayesian inference requires using simulated samples to summarize posterior distributions or calculate any relevant quantities of interest. So, posterior samples from the three Markov chains can be merged together using the following code:

```
R> result <- as.mcmc(do.call(rbind, res))
```

Next, the posterior samples of each parameter are extracted from the `result` object as follows:

```
R> alpha <- result[,1]; beta1 <- result[,2]; beta2 <- result[,3]; beta3 <-
result[,4]
```

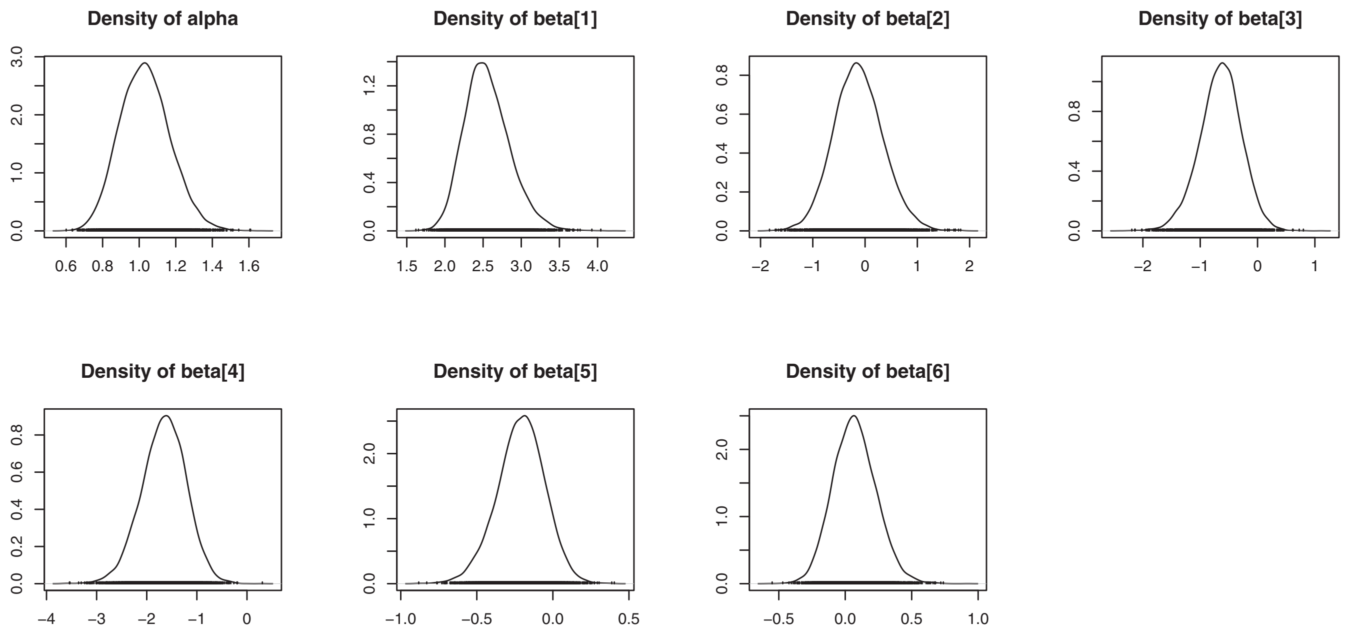

FIGURE 2 Density plots for α and β's of the AFT model (6)

```
R> beta4 <- result[,5]; beta5 <- result[,6]; beta6 <- result[,7]
```

The median survival time is an important summary measure for survival data. This quantity can be easily estimated for any AFT model.[38] A common effect measure derived from this quantity is the *relative median* (RM) between two individuals with covariate vectors $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. This measure compares the median of the survival time between both individuals and is defined as:

$$\mathrm{RM}(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{\beta}) = \exp\left\{(\boldsymbol{x}_1 - \boldsymbol{x}_2)^\top \boldsymbol{\beta}\right\}.$$

As an illustration, we can easily summarize the posterior distribution of the RM between two men of the same `age` and `diagyr` (year of diagnosis) but one in `stage`=3 and the other in `stage`=4:

```
R> RM.s3_s4 <- exp(beta3 - beta4)
R> summary(RM.s3_s4)
Iterations = 1:15000
Thinning interval = 1
Number of chains = 1
Sample size per chain = 15000

1. Empirical mean and standard deviation for each variable,
 plus standard error of the mean:

 Mean SD Naive SE Time-series SE
 3.01495 1.36246 0.01112 0.01145

2. Quantiles for each variable:

 2.5
1.209 2.084 2.756 3.613 6.435
```

2.2 Proportional hazards models

The Cox PH model[37] expresses the hazard function $h(t)$ of the survival time of each individual of the target population as the product of a common baseline hazard function $h_0(t)$, which determines the shape of $h(t)$, and an exponential term which includes the relevant covariates $\boldsymbol{x}$ with regression coefficients $\boldsymbol{\beta}$ as follows:

$$h(t|h_0, \boldsymbol{\beta}) = h_0(t) \exp\left\{\boldsymbol{x}^\top \boldsymbol{\beta}\right\}. \tag{7}$$

The estimation of the regression coefficients in the Cox model under the frequentist approach can be obtained without specifying a model for the baseline hazard function by using partial likelihood methodology.[37] This is not the case of Bayesian analysis which in general needs to specify some model for the baseline hazard.[38] Depending on the context of the study, baseline hazard misspecification can imply a loss of valuable model information that makes it impossible to fully report the estimation of the outcomes of interest, such as probabilities or survival curves for relevant covariate patterns.[48] This is specially important in survival studies where $h_0(t)$ represents the natural course of a disease or an infection, or even the control group when comparing several treatments.

Baseline hazard functions can be modeled considering some of the usual probability distributions in the survival analysis framework such as exponential, Weibull, Gompertz, and so on. However, these specifications give restricted shapes which do not allow the presence of irregular behaviors. In addition, it is also possible to specify more flexible hazard shapes that allow for multimodal patterns by means of piecewise constant functions or spline functions, among other proposals.[49] Theoretical and methodological aspects of different flexible approaches to define the baseline hazard function can be found in several specific references such as Ibrahim et al,[8] Lin et al,[50] Mitra and Müller,[51] Bogaerts et al,[12] and Lázaro et al.[49] The BUGS manual[43] shows a very flexible model implementation, using a Poisson approach, that allows the hazard to change at every observed event time.

2.2.1 Model specification

The PH model implemented here is also illustrated with the *larynx* dataset (see Section 2.1). Survival time for each individual is modeled by means of the hazard function:

$$h(t|h_0, \boldsymbol{\beta}) = h_0(t) \exp\left\{\beta_2 \mathrm{I}_{(\texttt{stage=2})} + \beta_3 \mathrm{I}_{(\texttt{stage=3})} + \beta_4 \mathrm{I}_{(\texttt{stage=4})} + \beta_5 \texttt{age} + \beta_6 \texttt{diagyr}\right\}, \tag{8}$$

with baseline hazard function defined as a mixture of piecewise constant functions, $h_0(t|\lambda) = \sum_{k=1}^{K} \lambda_k \, I_{(a_{k-1},a_k]}(t)$, $t > 0$, where $\lambda = (\lambda_1, \dots, \lambda_K)$ and $I_{(a_{k-1},a_k]}(t)$ is the indicator function defined as 1 when $t \in (a_{k-1}, a_k]$ and 0 otherwise. We consider a total number of knots $K = 3$ and an equally-spaced partition of the time axis from $a_0 = 0$ to $a_4 = 10.70$ which corresponds to the longest survival time observed. We select equally-spaced partitions to allow the hazard may still exhibit interesting features if some interval presents a dearth of event times. In turn, this partition ensured a sufficient number of observations at each interval to estimate each λ_k. Prior scenario is set under a non-informative independent framework with a N(0,0.001) for β's and an independent gamma distributions, Ga(0.01,0.01), for each λ.

Piecewise constant baseline hazard functions can accommodate different shapes of the hazard depending on the particular characteristics of the partition of the time axis (see Breslow,[52] Murray et al,[53] and Lee et al[54] for different proposals in the subject). Within the Bayesian framework, the specification of K and $\boldsymbol{a} = (a_0, a_1, \dots, a_{K+1})$ can be avoided by treating them as unknown parameters and estimating them into the inferential process.[55] However, the additional computational burden typically does not justify the marginal gains in approximation accuracy over a reasonable prespecified partition.[53] Similarly, there is a wide range of approaches to define marginal prior distributions for h_0 parameters, from prior independence to prior correlation. Correlated scenarios account for shape restrictions and also avoid overfitting and strong irregularities in the estimation process.[49]

The likelihood function of this specific model is not implemented in JAGS, so the "zeros trick" approach using a Poisson distribution has been used to specify it indirectly.[43,56]

2.2.2 | Model implementation

The variables `age`, `diagyr`, `stage`, and `X` are defined as in Section 2.1. In addition, as previously commented, piecewise constant is handled in JAGS considering the "zeros trick."[43,56] To execute it, it is necessary to reformat the individual times (both observed and right-censored) to build an auxiliary variable (`int.obs`) that identifies the interval in which the ith individual experiences the event of interest. So, the following code is dedicated to define a time axis partition (ie, intervals) and the `int.obs` variable:

```
R> # Time axis partition
R> K <- 3 # number of intervals
R> a <- seq(0, max(larynx$time) + 0.001, length.out = K + 1)
R> # int.obs: vector that tells us at which interval each observation is
R> int.obs <- matrix(data = NA, nrow = nrow(larynx), ncol = length(a) - 1)
R> d <- matrix(data = NA, nrow = nrow(larynx), ncol = length(a) - 1)
R> for(i in 1:nrow(larynx)) {
+ for(k in 1:(length(a) - 1)) {
+ d[i, k] <- ifelse(time[i] - a[k] > 0, 1, 0) * ifelse(a[k + 1] - time[i] > 0, 1,
0)
+ int.obs[i, k] <- d[i, k] * k
+ }
+ }
R> int.obs <- rowSums(int.obs)
```

Listing 3 shows a generic implementation of a PH piecewise constant model in BUGS syntax using *larynx* data.

```
model {
  for(i in 1:n) {
    # Pieces of the cumulative hazard function
    for(k in 1:int.obs[i]) {
      cond[i, k] <- step(time[i] - a[k + 1])
      HH[i, k] <- cond[i, k] * (a[k + 1] - a[k]) * lambda[k] +
        (1 - cond[i, k]) * (time[i] - a[k]) * lambda[k]
    }
    # Cumulative hazard function
    H[i] <- sum(HH[i, 1:int.obs[i]])
  }
  for(i in 1:n) {
    # Linear predictor
    elinpred[i] <- exp(inprod(beta[], X[i, ]))
    # Log-hazard function
    logHaz[i] <- log(lambda[int.obs[i]] * elinpred[i])
    # Log-survival function
    logSurv[i] <- -H[i] * elinpred[i]
    # Definition of the log-likelihood using zeros trick
    phi[i] <- 100000 - delta[i] * logHaz[i] - logSurv[i]
    zeros[i] ~ dpois(phi[i])
  }
  # Prior distributions
  for(l in 1:Nbetas) {
    beta[l] ~ dnorm(0, 0.001)
  }
  for(k in 1:m) {
    lambda[k] ~ dgamma(0.01, 0.01)
  }
}
```

Listing 3: PH model in BUGS syntax (file named as **PH.txt**)

Once the variables have been defined, a list with all the elements required in the model is created:

```
R> d.jags <- list(n = nrow(larynx), m = length(a) - 1, delta = larynx$delta,
+ time = larynx$time, X = X[, -1], a = a, int.obs = int.obs, Nbetas = ncol(X) - 1,
+ zeros = rep(0, nrow(larynx)))
```

The initial values for each PH model parameter are passed to JAGS using a function that returns a list of random values:

```
R> i.jags <- function() {
+ list(beta = rnorm(ncol(X) - 1), lambda = runif(3, 0.1))
+ }
```

The vector of monitored/saved parameters is:

```
R> p.jags <- c("beta", "lambda")
```

Next, the JAGS model is compiled:

```
R> library("rjags")
R> m2 <- jags.model(data = d.jags, file = "PH.txt", inits = i.jags, n.chains = 3)
```

We now run the model for 1000 burn-in simulations:

```
R> update(m2, 1000)
```

Finally, the model is run for 50 000 additional simulations to keep one in 10 so that a proper thinning is done:

```
R> res <- coda.samples(m2, variable.names = p.jags, n.iter = 50000, n.thin = 10)
```

Similar to the first example (Section 2.1), numerical and graphical summaries of the model parameters can be obtained using the `summary` and `densplot` functions, respectively. Gelman and Rubin's convergence diagnostic can be calculated with the `gelman.diag` function, and the `traceplot` function provides a visual way to inspect sampling behavior and assesses mixing across chains and convergence.

Next, simulations from the three Markov chains are merged together for inference:

```
R> result <- as.mcmc(do.call(rbind, res))
```

The posterior samples of each parameter are obtained by:

```
R> beta2 <- result[,1]; beta3 <- result[,2]; beta4 <- result[,3]; beta5 <-
result[,4]
R> beta6 <- result[,5]; lambda1 <- result[,6]; lambda2 <- result[,7]; lambda3 <-
result[,8]
```

Table 1 shows posterior summaries for the PH model parameters using *larynx* data.

The last column of Table 1 contains the posterior probability that the corresponding parameter is positive. A probability equal to 0.5 indicates that a positive value of the parameter is equally likely than a negative one. Once we have

TABLE 1 Posterior summaries for the PH model parameters

Parameter	Mean	SD	2.5%	50%	97.5%	P(· > 0| data)
β_2 (stage=2)	0.152	0.474	−0.811	0.163	1.050	0.633
β_3 (stage=3)	0.672	0.363	−0.037	0.672	1.388	0.969
β_4 (stage=4)	1.804	0.443	0.924	1.809	2.659	1.000
β_5 (age)	0.215	0.155	−0.086	0.214	0.523	0.918
β_6 (diagyr)	−0.042	0.167	−0.368	−0.042	0.288	0.400
λ_1	0.069	0.021	0.035	0.066	0.116	1.000
λ_2	0.104	0.035	0.048	0.100	0.185	1.000
λ_3	0.079	0.064	0.008	0.062	0.246	1.000

the posterior sample of each parameter stored, the calculation of this posterior probability, for example for β_2, is given by `mean(beta2>0)`.

A relevant quantity for PH models is the *hazard ratio* (HR), also called *relative risk*, between two individuals with covariate vectors $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. This measure is defined as:

$$\mathrm{HR}(\boldsymbol{x}_1, \boldsymbol{x}_2, h_0, \boldsymbol{\beta}) = \frac{h(t|\boldsymbol{x}_1, h_0, \boldsymbol{\beta})}{h(t|\boldsymbol{x}_2, h_0, \boldsymbol{\beta})} = \exp\left\{(\boldsymbol{x}_1 - \boldsymbol{x}_2)^\top \boldsymbol{\beta}\right\},$$

and it is time independent. As an illustration, we can easily summarize the posterior distribution of the HR between two men of the same `age` and `diagyr` (year of diagnosis) but one in `stage=3` and the other in `stage=4`:

```
R> HR.s3_{s}4 <- exp(beta3 - beta4)
R> summary(HR.s3_{s}4)
Iterations = 1:15000
Thinning interval = 1
Number of chains = 1
Sample size per chain = 15000

1. Empirical mean and standard deviation for each variable,
 plus standard error of the mean:
 Mean SD Naive SE Time-series SE
 0.354210 0.163810 0.001338 0.001338

2. Quantiles for each variable:
 2.5
0.1384 0.2404 0.3217 0.4297 0.7667
```

3 | MIXTURE CURE MODELS

3.1 | Cure models

Cure models deal with target populations in which a part of the individuals cannot experience the event of interest. This type of models has widely been matured as a consequence of the discovery and development of new treatments against cancer. The rationale of considering a cure subpopulation comes from the idea that a successful treatment removes totally the original tumor and the individual cannot experience any recurrence of the disease. These models allow to estimate the probability of cure, a key and valuable outcome in cancer research.

Mixture cure models are the most popular cure models.[57] They consider the target population as a mixture of susceptible and non-susceptible individuals for the event of interest. Let Z be a cure random variable defined as $Z = 0$ for susceptible and $Z = 1$ for cured or immune individuals. Cure and non cure probabilities are $P(Z = 1) = \eta$ and $P(Z = 0) =$

$1 - \eta$, respectively. The survival function for each individual in the cured and uncured subpopulation, $S_c(t)$ and $S_u(t)$, $t > 0$, respectively, is

$$S_u(t) = P(T > t|Z = 0), \quad S_c(t) = P(T > t|Z = 1), \tag{9}$$

and the general survival function for T can be expressed as $S(t) = P(T > t) = \eta + (1 - \eta)S_u(t)$. It is important to point out that $S_u(t)$ is a proper survival function but $S(t)$ is not. It goes to η and not to zero when t goes to infinity.

The effect of a baseline covariate vector $\boldsymbol{x}$ on the cure fraction η for each individual is typically modeled by means of a logistic link function, logit(η), but the probit link or the complementary log-log link could also be used. Covariates for modeling T in the uncured subpopulation are usually considered via Cox models. Cure fraction model is usually known as the *incidence model* and the survival model (ie, time-to-event T in the uncured subpopulation) as the *latency model*.[4]

3.2 | *bmt* dataset

We consider a bone marrow transplant dataset, referred to as *bmt*. It is available from the `smcure` package:[58]

```
R> library("smcure")
R> data("bmt")
R> str(bmt)
'data.frame': 91 obs. of 3 variables:
 $ Time : num 11 14 23 31 32 35 51 59 62 78 ...
 $ Status: num 1 1 1 1 1 1 1 1 1 1 ...
 $ TRT : num 0 0 0 0 0 0 0 0 0 0 ...
```

This dataset refers to a bone marrow transplant study for the refractory acute lymphoblastic leukemia patients, in which 91 patients were divided into two treatment groups.[59] The following variables were observed for each patient:

- `Time`: time to death (in days).
- `Status`: censoring indicator (1: if patient is uncensored; 0: otherwise).
- `TRT`: treatment group indicator (0: allogeneic; 1: autologous).

3.3 | Model specification

The (cure subpopulation) incidence model for each individual is expressed in terms of a logistic regression:

$$\text{logit}\,[\eta(\beta_{C1}, \beta_{C2})] = \beta_{C1} + \beta_{C2}\texttt{TRT}, \tag{10}$$

where β_{C1} represents an intercept and β_{C2} is the regression coefficient for the `TRT` covariate.

Survival time for each individual in the uncured subpopulation is modeled from a PH specification:

$$h_u(t|h_0, \beta_U) = h_0(t) \exp\{\beta_U \texttt{TRT}\}, \tag{11}$$

with $h_0(t) = \lambda\ \alpha\ t^{\alpha-1}$ specified as a Weibull baseline hazard function, where α and λ are the shape and scale parameters, respectively, and β_U is the regression coefficient for the `TRT` covariate.

We assume prior independence and specify prior marginal distributions based on non-informative distributions commonly employed in the literature. The β's follow a N(0,0.001), while λ and α follow a Gamma(0.01,0.01) and a Un(0,10), respectively.

The likelihood function for mixture cure models is not implemented in JAGS, so the "zeros trick" approach using a Poisson distribution has also been used to specify it indirectly.[43,56]

3.4 | Model implementation

We have created two design matrices, one for model (10) and another for model (11), with the `TRT` covariate:

```
R> XC <- model.matrix(~ TRT, data = bmt) # Reference = allogeneic
R> XU <- model.matrix(~ TRT, data = bmt)
R> XU <- matrix(XU[, -1], ncol = 1) # Remove intercept
```

Listing 4 shows a generic implementation of a mixture cure model in BUGS syntax using *bmt* data.

```
model {
  for(i in 1:n) {
    # Logistic regression model (cured subpopulation)
    logit(eta[i]) <- inprod(betaC[], XC[i, ])
    # PH model (uncured subpopulation)
    # Weibull baseline
    base[i] <- lambda * alpha * pow(t[i], alpha - 1)
    elinpred[i] <- exp(inprod(betaU[], XU[i, ]))
    # Log-hazard function
    logHaz[i] <- log(base[i] * elinpred[i])
    # Log-survival function
    logSurv[i] <- -lambda * pow(t[i], alpha) * elinpred[i]
    # Definition of the log-likelihood using zeros trick
    logLike[i] <- delta[i] * (log(1 - eta[i]) + logHaz[i] + logSurv[i]) +
      (1 - delta[i]) * log(eta[i] + (1 - eta[i]) * exp(logSurv[i]))
    phi[i] <- 100000 - logLike[i]
    zeros[i] ~ dpois(phi[i])
  }
  # Prior distributions
  for(l in 1:NbetasC) {
    betaC[l] ~ dnorm(0, 0.001)
  }
  for(l in 1:NbetasU) {
    betaU[l] ~ dnorm(0, 0.001)
  }
  lambda ~ dgamma(0.01, 0.01)
  alpha ~ dunif(0, 10)
}
```

Listing 4: Mixture cure model in BUGS syntax (file named as **Cure.txt**)

Once the variables have been defined, a list with all the elements required in the model is created:

```
R> d.jags <- list(n = nrow(bmt), t = bmt$Time, XC = XC, XU = XU,
+ delta = bmt$Status, zeros = rep(0, nrow(bmt)), NbetasC = ncol(XC), NbetasU =
ncol(XU))
```

The initial values for each mixture cure model parameter are passed to JAGS using a function that returns a list of random values:

```
R> i.jags <- function() {
```

TABLE 2 Posterior summaries for the mixture cure model parameters

Parameter	Mean	SD	2.5%	50%	97.5%	P(· > 0| data)
β_{C1} (`intercept`)	−1.015	0.349	−1.731	−1.002	−0.365	0.001
β_{C2} (`TRT`)	−0.419	0.519	−1.445	−0.417	0.591	0.208
β_U (`TRT`)	0.762	0.269	0.239	0.760	1.294	0.998
α	1.143	0.105	0.943	1.140	1.354	1.000
λ	0.002	0.001	0.000	0.002	0.006	1.000

```
+ list(betaC = rnorm(ncol(XC)), betaU = rnorm(ncol(XU)), lambda = runif(1), alpha =
runif(1))
+ }
```

The vector of monitored/saved parameters is:

```
R> p.jags <- c("betaC", "betaU", "alpha", "lambda")
```

Next, the JAGS model is compiled:

```
R> library("rjags")
R> m3 <- jags.model(data = d.jags, file = "Cure.txt", inits = i.jags, n.chains =
3)
```

We now run the model for 10000 burn-in simulations:

```
R> update(m3, 10000)
```

Finally, the model is run for 100 000 additional simulations to keep one in 100 so that a proper thinning is done:

```
R> res <- coda.samples(m3, variable.names = p.jags, n.iter = 100000, n.thin = 100)
```

Similar to the first example (Section 2.1), numerical and graphical summaries of the model parameters can be obtained using the `summary` and `densplot` functions, respectively. Gelman and Rubin's convergence diagnostic can be calculated with the `gelman.diag` function, and the `traceplot` function provides a visual way to inspect sampling behavior and assesses mixing across chains and convergence.

Next, simulations from the three Markov chains are merged together for inference:

```
R> result <- as.mcmc(do.call(rbind, res))
```

The posterior samples of each parameter are obtained by:

```
R> alpha <- result[,1]; betaC1 <- result[,2]; betaC2 <- result[,3]
R> betaU <- result[,4]; lambda <- result[,5]
```

Table 2 shows posterior summaries for the mixture cure model parameters using *bmt* data.

As we discussed earlier, the cure fraction (η) is a relevant quantity for mixture cure models. For allogeneic (`TRT`=0) autologous and (`TRT`=1) treated patients it is modeled as:

$$\eta(\texttt{TRT}, \beta_{C1}, \beta_{C2}) = \frac{\exp(\beta_{C1} + \beta_{C2}\texttt{TRT})}{1 + \exp(\beta_{C1} + \beta_{C2}\texttt{TRT})}.$$

We can easily summarize the posterior distribution of the cure fraction for individuals in both groups:

```
R> CP.allo <- exp(betaC1) / (1 + exp(betaC1))
R> CP.auto <- exp(betaC1 + betaC2) / (1 + exp(betaC1 + betaC2))
```

```
R> summary(cbind(CP.allo, CP.auto))
 CP.allo CP.auto
 Min. :0.01604 Min. :0.02277
 1st Qu.:0.22426 1st Qu.:0.15680
 Median :0.26850 Median :0.19477
 Mean :0.27146 Mean :0.19925
 3rd Qu.:0.31515 3rd Qu.:0.23699
 Max. :0.65573 Max. :0.55898
```

The uncured survival curve based on posterior samples is another relevant information in this type of studies. So, from the posterior samples obtained above, we can summarize the posterior distribution of the mean value of the uncured survival curve for allogeneic and autologous treated patients in a grid of points as follows:

```
R> grid <- 100
R> time <- seq(0, bmt$Time, len = grid)
R> surv.allo <- surv.auto <- vector()
R> for(l in 1:grid) {
+ surv.allo[l] <- mean(exp(-lambda * time[l]ˆ{a}lpha))
+ surv.auto[l] <- mean(exp(-lambda * exp(betaU) * time[l]ˆ{a}lpha))
+ }
```

Figure 3 shows the difference between both curves using the code below:

```
R> library("ggplot2")
R> treat.col <- rep(0:1, each = grid)
R> treat.col[treat.col == 0] <- "allogeneic"
R> treat.col[treat.col == 1] <- "autologous"
R> df <- data.frame(time = rep(time, 2), survival = c(surv.allo, surv.auto),
+ treatment = treat.col)
R> ggplot(data = df, aes(x = time, y = survival, group = treatment, colour =
treatment)) +
+ geom_{l}ine() + theme_{b}w() + theme(legend.position = "top")
```

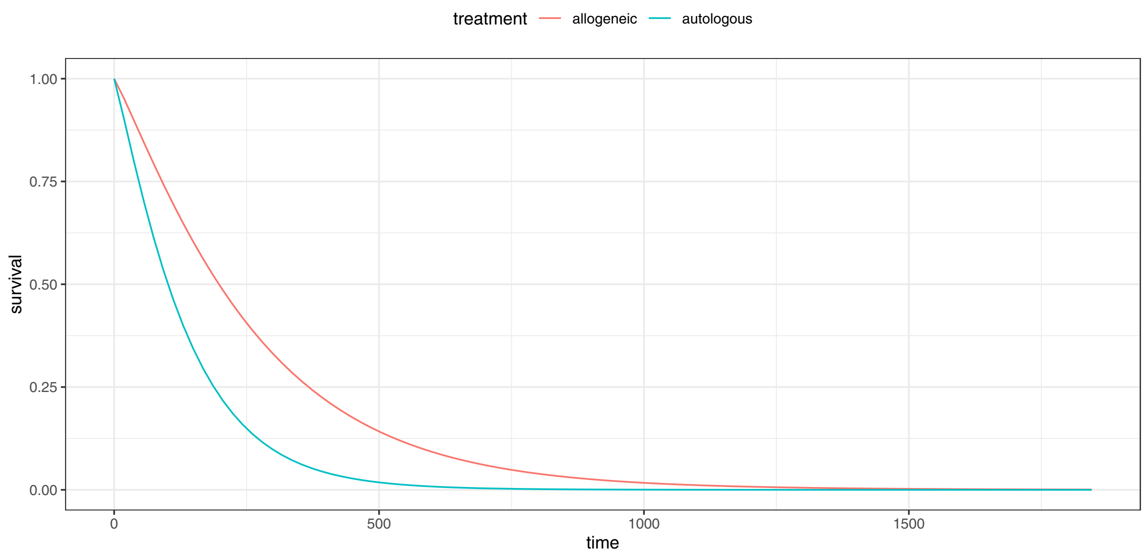

FIGURE 3 Posterior mean of the uncured survival function from the mixture cure model (11) [Colour figure can be viewed at wileyonlinelibrary.com]

4 | COMPETING RISKS MODELS

Competing risks occur when the survival process includes more than one cause of failure. In the case of different causes of death it is only possible to report the first event to occur.[60] There are different approaches for competing risk models: multivariate time to failure model, the cause-specific hazards model, the mixture model, the subdistribution model, and the full specified subdistribution model.[61] We will only consider here the cause-specific hazards model, possibly the most popular of them.

Let T_k be the random variable that represents the time-to-event from cause k, for $k = 1, \dots, K$, where K is the total number of different events. The only survival time observed $T = \min\{T_1, T_2, \dots, T_K\}$ usually corresponds to the earliest cause together with their indicator $\delta = k$ when the subsequent individual experiences the event due to cause k. The key concept in a competing risk model is the cause-specific hazard function for cause k, which assesses the hazard of experiencing the event k in the presence of the rest of competing events, and it is defined by:

$$h_k(t) = \lim_{\Delta t \to 0} \frac{P(t \le T < t + \Delta t, \delta = k | T \ge t)}{\Delta t}. \tag{12}$$

Inference for each $h_k(t)$ considers the observed failure times for cause k as censored observations for the rest of events. Another relevant concept in the competing framework is the cumulative incidence function for cause k, defined as follows:

$$F_k(t) = P(T \le t, \delta = k) = \int_0^t h_k(u)\ S(u)\ \mathrm{d}u, \tag{13}$$

where $S(t)$ is the overall survival function. It is typically expressed in terms of the different cause-specific hazard functions in accordance with

$$S(t) = P(T > t) = \exp\left\{ -\sum_{k=1}^{K} \int_0^t h_k(u)\ \mathrm{d}u \right\}.$$

The cumulative incidence function is not a proper cumulative distribution function, that is, the probability that the subsequent individual fails from cause k is $F_k(\infty) = P(\delta = k) \neq 1$. The sub-survival function for cause k, defined as $S_k(t) = P(T > t, \delta = k)$, is also not a proper survival function.

4.1 | *okiss* dataset

We consider a stem-cell transplanted patients dataset, referred to as *okiss*. It is available from the `compeir` package:[62]

```
R> library("compeir")
R> data("okiss")
R> str(okiss)
'data.frame': 1000 obs. of 4 variables:
 $ time : 'times' num 21 6 14 12 11 18 10 8 17 10 ...
 ..- attr(*, "format")= chr "h:m:s"
 $ status: num 2 1 2 2 2 2 7 2 2 2 ...
 $ allo : num 1 1 1 1 1 1 1 0 1 1 ...
 $ sex : Factor w/ 2 levels "f","m": 1 1 1 2 2 2 2 1 2 2 ...
```

This dataset provides information about a sub-sample of 1000 patients enrolled in the ONKO-KISS programme, which is part of the German National Reference Centre for Surveillance of Hospital-Acquired Infections.[63] These patients have been treated by peripheral blood stem-cell transplantation, which has become a successful therapy for severe hematologic diseases. After transplantation, patients are neutropenic, that is, they have a low count of white blood cells, which are the cells that primarily avert infections.[64] Occurrence of bloodstream infection during neutropenia is a severe complication. The following variables were observed for each patient:

- `time`: time (in days) of neutropenia until first event.
- `status`: event status indicator (1: infection; 2: end of neutropenia; 7: death; 11: censored observation).
- `allo`: transplant type indicator (0: autologous; 1: allogeneic).
- `sex`: sex of each patient (m: if patient is male; f: if patient is female).

We have redefined the `status` variable using an auxiliary one (`delta`) in a matrix format:

```
R> delta <- matrix(c(as.integer(okiss$status == 1), as.integer(okiss$status == 2),
+ as.integer(okiss$status == 7)), ncol = 3)
R> head(delta)
 [,1] [,2] [,3]
[1,] 0 1 0
[2,] 1 0 0
[3,] 0 1 0
[4,] 0 1 0
[5,] 0 1 0
[6,] 0 1 0
```

where the events 1 (infection), 2 (end of neutropenia), and 7 (death) are indicated with a value of 1 in column 1, 2, or 3, respectively, and a row with only 0's represents a censored observation.

4.2 | Model specification

Cause-specific hazard functions for infection ($k = 1$), end of neutropenia ($k = 2$), and death ($k = 3$) are modeled from a PH specification:

$$h_k(t|h_{0k}, \boldsymbol{\beta}_k) = h_{0k}(t) \exp\{\beta_{1k}\texttt{allo} + \beta_{2k}\texttt{sex}\}, \quad k = 1, 2, 3, \tag{14}$$

with $h_{0k}(t) = \lambda_k\, \alpha_k\, t^{\alpha_k - 1}$ for event k specified as a Weibull baseline hazard function, where α_k and λ_k are the shape and scale parameters, respectively, and $\boldsymbol{\beta}_k = (\beta_{1k}, \beta_{2k})^\top$ are regression coefficients for the `allo` and `sex` covariates, respectively, for $k = 1, 2, 3$. We assume prior independence and specify prior marginal based on non-informative distributions commonly employed in the literature. The β's follow a N(0,0.001), while λ's and α's follow a Gamma(0.01,0.01) and a Un(0,10), respectively.

4.3 | Model implementation

We have created a design matrix `X` with the covariates `allo` and `sex`:

```
R> X <- model.matrix(~ allo + sex, data = okiss) # Reference = female
R> X <- X[, -1] # Remove intercept
```

Listing 5 shows a generic implementation of a competing risks model in BUGS syntax using *okiss* data.

```
model {
  for(i in 1:n) {
    for(k in 1:Nrisks) {
      # Weibull baseline
      base[i, k] <- lambda[k] * alpha[k] * pow(t[i], alpha[k] - 1)
      elinpred[i, k] <- exp(inprod(beta[, k], X[i, ]))
      # Log-hazard functions
      logHaz[i, k] <- log(base[i, k] * elinpred[i, k])
      # Log-survival functions
      logSurv[i, k] <- -lambda[k] * pow(t[i], alpha[k]) * elinpred[i, k]
    }
    # Definition of the log-likelihood using zeros trick
    phi[i] <- 100000 - inprod(delta[i, ], logHaz[i, ]) - sum(logSurv[i, ])
    zeros[i] ~ dpois(phi[i])
  }
  # Prior distributions
  for(k in 1:Nrisks) {
    for(l in 1:Nbetas) {
      beta[l, k] ~ dnorm(0, 0.001)
    }
    lambda[k] ~ dgamma(0.01, 0.01)
    alpha[k] ~ dunif(0, 10)
  }
}
```

Listing 5: Competing risks model in BUGS syntax (file named as **CR.txt**)

Once the variables have been defined, a list with all the elements required in the model is created:

```
R> d.jags <- list(n = nrow(okiss), t = as.vector(okiss$time), X = X,
+ delta = delta, zeros = rep(0, nrow(okiss)), Nbetas = ncol(X), Nrisks =
ncol(delta))
```

The initial values for each competing risks model parameter are passed to JAGS using a function that returns a list of random values:

```
R> i.jags <- function() {
+ list(beta = matrix(rnorm(ncol(X) * ncol(delta)), ncol = ncol(delta)),
+ lambda = runif(ncol(delta)), alpha = runif(ncol(delta)))
+ }
```

The vector of monitored/saved parameters is:

```
R> p.jags <- c("beta", "alpha", "lambda")
```

Next, the JAGS model is compiled:

```
R> library("rjags")
R> m4 <- jags.model(data = d.jags, file = "CR.txt", inits = i.jags, n.chains = 3)
```

We now run the model for 1000 burn-in simulations:

```
R> update(m4, 1000)
```

Finally, the model is run for 10 000 additional simulations to keep one in 10 so that a proper thinning is done:

```
R> res <- coda.samples(m4, variable.names = p.jags, n.iter = 10000, n.thin = 10)
```

Similar to the first example (Section 2.1), numerical and graphical summaries of the model parameters can be obtained using the `summary` and `densplot` functions, respectively. Gelman and Rubin's convergence diagnostic can be calculated with the `gelman.diag` function, and the `traceplot` function provides a visual way to inspect sampling behavior and assesses mixing across chains and convergence.

Next, simulations from the three Markov chains are merged together for inference:

```
R> result <- as.mcmc(do.call(rbind, res))
```

The posterior samples of each parameter are obtained by:

```
R> alpha1 <- result[,1]; alpha2 <- result[,2]; alpha3 <- result[,3]; beta11 <-
result[,4]
R> beta21 <- result[,5]; beta12 <- result[,6]; beta22 <- result[,7]; beta13 <-
result[,8]
R> beta23 <- result[,9]; lambda1 <- result[,10]; lambda2 <- result[,11]; lambda3 <-
result[,12]
```

Table 3 shows posterior summaries for the competing risks model parameters using *okiss* data.

As discussed in Section 4, the cumulative incidence function $F_k(t)$ in (13) is the most appropriate way to analyze the evolution of each cause k over time. For our PH specification (14), it is given by:

$$F_k(t) = \int_0^t h_k(t|h_{0k}, \boldsymbol{\beta}_k) \ \exp\left\{ -\sum_{l=1}^{3} \int_0^u h_l(v|h_{0l}, \boldsymbol{\beta}_l) \, \mathrm{d}v \right\} \, \mathrm{d}u, \tag{15}$$

where $h_k(t|h_{0k}, \boldsymbol{\beta}_k)$ is defined in (14).

The integral in (15) has no closed form, so some approximate method of integration is required. To do this, we first have created a function `fk` which describes the integrand of (15):

```
R> fk <- function(u.vect, lambda, alpha, beta, x, k) {
+ res <- sapply(u.vect, function(u) {
+ # Cause-specific hazard
+ hk <- lambda[k] * alpha[k] * (u^(alpha[k] - 1)) * exp(sum(unlist(beta[,k]) * x))
+ # Cumulative cause-specific hazard
+ Hk <- lambda * (rep(u, length(lambda))^{a}lpha) * exp((t(beta)
```

TABLE 3 Posterior summaries for the competing risks model parameters

Parameter	Mean	SD	2.5%	50%	97.5%	P(· > 0| data)
Infection (h_1)						
β_{11} (allo)	−0.523	0.151	−0.817	−0.523	−0.225	0.000
β_{21} (sex)	0.158	0.148	−0.130	0.156	0.451	0.857
λ_1	0.014	0.003	0.009	0.014	0.022	1.000
α_1	1.137	0.070	1.003	1.136	1.278	1.000
End of neutropenia (h_2)						
β_{12} (allo)	−1.193	0.075	−1.340	−1.193	−1.046	0.000
β_{22} (sex)	−0.102	0.073	−0.244	−0.103	0.042	0.081
λ_2	0.008	0.001	0.006	0.008	0.010	1.000
α_2	2.033	0.045	1.947	2.034	2.120	1.000
Death (h_3)						
β_{13} (allo)	−0.617	0.747	−2.001	−0.650	0.945	0.193
β_{23} (sex)	0.446	0.730	−0.859	0.408	1.986	0.718
λ_3	0.000	0.000	0.000	0.000	0.000	1.000
α_3	2.628	0.418	1.813	2.623	3.466	1.000

```
+ # Cause-specific hazard x Overall survival
+ aux <- hk * exp(-sum(Hk))
+ return(aux)
+ })
+ return(res) }
```

Next, we have created a function `cif` that computes $F_k(t)$ in (15) by integrating out the `fk` function using the `integral` function available from the `pracma` package.[65]

```
R> library("pracma")
R> cif <- function(tt, lambda, alpha, beta, x, k) {
+ return(integral(fk, xmin = 0, xmax = tt, method = "Simpson", lambda = lambda,
alpha = alpha,
+ beta = beta, x = x, k = k)) }
```

Finally, we have constructed a function `mcmc_cif` that takes the output from JAGS (variable `obj`) and computes $F_k(t)$ in (15) for a vector of times (variable `t.pred`) using covariates `x`. Note that `mcmc_cif` is based on the `mclapply` function, available from the `parallel` package,[32] to speed computations up.

```
R> library("parallel")
R> options(mc.cores = detectCores())
R> mcmc_{c}if <- function(obj, t.pred, x) {
+ var.names <- names(obj)
+ # Indices of beta's, alpha's, and lambda's
+ b.idx <- which(substr(var.names, 1, 4) == "beta")
+ a.idx <- which(substr(var.names, 1, 5) == "alpha")
+ l.idx <- which(substr(var.names, 1, 6) == "lambda")
+ # Number of causes and number of covariates
+ K <- length(a.idx)
+ n.b <- length(b.idx) / K
+ # Sub-sample to speed up computations
+ samples.idx <- sample(1:nrow(obj), 200)
+
+ res <- lapply(1:K, function(k) {
+ sapply(t.pred, function(tt) {
+ aux <- mclapply(samples.idx, function(i) {
+ cif(tt, alpha = unlist(obj[i, a.idx]), lambda = unlist(obj[i, l.idx]),
+ beta = matrix(unlist(c(res[i, b.idx])), nrow = n.b), x = x, k = k)
+ })
+ return(mean(unlist(aux)))
+ })
+ })
+ return(res)
+ }
```

Hence, we redefine the MCMC output as a `data.frame` and set a vector of times to evaluate the cumulative incidence function. In this example, we are interested in this function when both covariates are 1 (ie, allogeneic transplant and male).

```
R> res <- as.data.frame(result)
R> t.pred <- seq(0, 100, by = 2.5)
R> cum_{i}nc <- mcmc_{c}if(res, t.pred, c(1, 1))
```

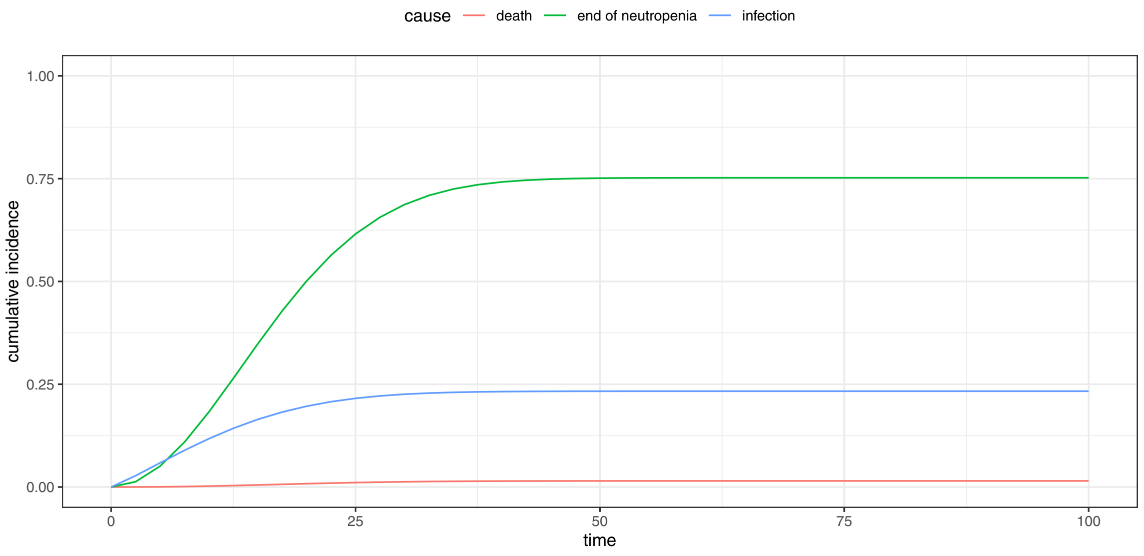

FIGURE 4 Posterior mean of the cumulative incidence function for `allo=1` (allogeneic transplant) and `sex=1` (male) from the competing risks model (14) [Colour figure can be viewed at wileyonlinelibrary.com]

Figure 4, generated with the code below, shows the posterior mean of the cumulative incidence function for a man with an allogeneic transplant for the three types of events considered in the *okiss* data.

```
R> library("ggplot2")
R> df <- data.frame(cif = unlist(cum_{i}nc), time = t.pred,
+ cause = rep(c("infection", "end of neutropenia", "death"), each =
length(t.pred)))
R> ggplot(data = df, aes(x = time, y = cif, group = cause)) + geom_{l}ine(aes(color
= cause)) +
+ ylab("cumulative incidence") + ylim(c(0,1)) + theme_{b}w() + theme(legend.position
= "top")
```

5 | MULTI-STATE MODELS

Multi-state models are a class of stochastic processes which account for event history data, with individuals who may experience different events in time. Relevant data are the events and the time elapsed between them. Multi-state models allow for different structures depending on the number and relationships between the states.[66] Outputs of interest are the usual in survival analysis (sometimes with a specific vocabulary, eg, the hazard function which is now called *transition intensity*) to which transition probabilities are added.

We concentrate on the illness-death model (also known as *disability model*) which is a particular multi-state model. This is a relevant model in irreversible diseases where a significant illness' progression increases the risk of a terminal event.[67,68] The underlying stochastic process $\{Z(t), t \geq 0\}$ describes the state of an individual at time t, where t is time from entry into the initial state (state 1). Each transition has its respective hazard function, for example, $h_{12}(t|\boldsymbol{\theta})$ is associated with time T_{12} from state 1 to state 2 ($1 \rightarrow 2$), while $h_{13}(t|\boldsymbol{\theta})$ and $h_{23}(t|\boldsymbol{\theta})$ represent the hazard functions for the time T_{13} ($1 \rightarrow 3$) and T_{23} ($2 \rightarrow 3$), respectively. If $h_{23}(t|\boldsymbol{\theta})$ does not depend on the time in which transition $1 \rightarrow 2$ occurs, the process will be known as *Markovian*. Otherwise, when transition $2 \rightarrow 3$ depends not only on the current state but also on the time elapsed since it was reached the process is called *semi-Markovian*.

The probabilistic behavior of the process is determined by the subsequent hazard functions from which transitions probabilities between states, defined as $p_{jk}(s, t|\boldsymbol{\theta}) = P(Z(t) = k|Z(s) = j, \boldsymbol{\theta})$, can be derived, where $s \leq t$, j and k are states,

$\sum_{k=j}^{3} p_{jk}(s,t|\boldsymbol{\theta}) = 1$, for $j = 1, 2, 3$, and $\boldsymbol{\theta}$ is the vector of model parameters. In the case of a semi-Markovian process, the transition probabilities from state 2 also incorporate the specific value of the time of transition from state 1 to 2.

Transition probabilities between states and hazard functions for the semi-Markovian specification are connected as follows:[69] spimergestart

$$p_{11}(s,t|\boldsymbol{\theta}) = \exp\left\{-\int_s^t \left[h_{12}(u|\boldsymbol{\theta}) + h_{13}(u|\boldsymbol{\theta})\right] \mathrm{d}u\right\}, \tag{16}$$

$$p_{22}(s,t|\boldsymbol{\theta},t_{12}) = \exp\left\{-\int_s^t h_{23}(u - t_{12}|\boldsymbol{\theta},t_{12})\, \mathrm{d}u\right\}, \quad t_{12} < s, \tag{17}$$

$$p_{12}(s,t|\boldsymbol{\theta}) = \int_s^t p_{11}(s,u|\boldsymbol{\theta})\, h_{12}(u|\boldsymbol{\theta})\, p_{22}(u,t|\boldsymbol{\theta},u)\, \mathrm{d}u, \tag{18}$$

$$p_{13}(s,t|\boldsymbol{\theta}) = 1 - p_{11}(s,t|\boldsymbol{\theta}) - p_{12}(s,t|\boldsymbol{\theta}), \tag{19}$$

$$p_{23}(s,t|\boldsymbol{\theta},t_{12}) = 1 - p_{22}(s,t|\boldsymbol{\theta},t_{12}), \tag{20}$$

$$p_{33}(s,t|\boldsymbol{\theta}) = 1, \tag{21}$$

spimergeend where the marginal probability $p_{22}(s,t|\boldsymbol{\theta})$ is obtained by integrating out $p_{22}(s,t|\boldsymbol{\theta},t_{12})$ with regard to the density of T_{12} from 0 to s.

The standard specification of an illness-death model is through the hazard function of the relevant survival times, generally modeled by means of PH models. Consequently, the specification of prior distributions for $\boldsymbol{\theta}$ must be addressed to them in Section (2.2).

5.1 | *heart2* dataset

We consider a heart transplant dataset, referred to as *heart2*. It is available from the `p3state.msm` package:[70]

```
R> library("p3state.msm")
R> data("heart2")
R> str(heart2)
'data.frame': 103 obs. of 8 variables:
 $ times1 : num 50 6 1 36 18 3 51 40 85 12 ...
 $ delta : int 0 0 1 1 0 0 1 0 0 1 ...
 $ times2 : num 0 0 15 3 0 0 624 0 0 46 ...
 $ time : int 50 6 16 39 18 3 675 40 85 58 ...
 $ status : int 1 1 1 1 1 1 1 1 1 1 ...
 $ age : num -17.16 3.84 6.3 -7.74 -27.21 ...
 $ year : num 0.123 0.255 0.266 0.49 0.608 ...
 $ surgery: int 0 0 0 0 0 0 0 0 0 0 ...
```

This dataset provides information about a sample of 103 patients of the Stanford Heart Transplant Program.[71] The patients are initially on the waiting list (state 1) and can either be transplanted (state 2, non-terminal event) and then die (state 3, terminal event), or just one or none of them because they continue to be on the waiting list. The following variables were observed for each patient:

- `times1`: time of transplant/censoring time (state 2).
- `delta`: transplant indicator (1: yes; 0: no).
- `times2`: time to death since the transplant/censoring time (state 3).
- `time`: `times1` + `times2`.
- `status`: censoring indicator (1: dead; 0: alive).

- `age`: age - 48 years.
- `year`: year of acceptance (in years after 1 Nov 1967).
- `surgery`: prior bypass surgery (1: yes; 0: no).

The patients had the following characteristics: 4 were censored for both events (transplant and death), 24 moved from state 1 (waiting list) to state 2 (transplant) and survived, 30 moved from state 1 to state 3 (death) without going through state 2, and 45 moved from state 1 to state 2 and then to state 3.

We have redefined `delta` and `status` variables using an auxiliary one (`event`) in a matrix format:

```
R> event <- matrix(c(heart2$delta, heart2$status * (1 - heart2$delta),
+ heart2$delta * heart2$status), ncol = 3)
R> head(event)
 [,1] [,2] [,3]
[1,] 0 1 0
[2,] 0 1 0
[3,] 1 0 1
[4,] 1 0 1
[5,] 0 1 0
[6,] 0 1 0
```

where each row represents the transitions of a patient, in which a 1 in columns 1, 2 and 3 indicates transition $1 \rightarrow 2$, $1 \rightarrow 3$, and $2 \rightarrow 3$, respectively, and a row with only 0's represents a censored observation.

We will adopt the semi-Markovian specification, but it could also be the Markovian one.[21] In this case, the model can also be seen as a *clock-reset specification*,[1] in which time starts at zero again after each transition.

5.2 | Model specification

Hazard functions for survival times T_{12}, T_{13}, and T_{23} are modeled from a PH specification: spimergestart

$$h_{12}(t|h_{01},\boldsymbol{\beta}_1) = h_{01}(t)\exp\{\beta_{11}\texttt{age} + \beta_{21}\texttt{year} + \beta_{31}\texttt{surgery}\}, \quad t > 0, \tag{22}$$

$$h_{13}(t|h_{02},\boldsymbol{\beta}_2) = h_{02}(t)\exp\{\beta_{12}\texttt{age} + \beta_{22}\texttt{year} + \beta_{32}\texttt{surgery}\}, \quad t > 0, \tag{23}$$

$$h_{23}(t|h_{03},\boldsymbol{\beta}_3, T_{12} = t_{12}) = h_{03}(t - t_{12})\exp\{\beta_{13}\texttt{age} + \beta_{23}\texttt{year} + \beta_{33}\texttt{surgery}\}, \quad t > t_{12}, \tag{24}$$

spimergeend with $h_{0k}(t) = \lambda_k \alpha_k t^{\alpha_k - 1}$ specified as a Weibull baseline hazard function, where α_k and λ_k are the shape and scale parameters, respectively, for $k = 1, 2, 3$, and $\boldsymbol{\beta}_k = (\beta_{1k}, \beta_{2k}, \beta_{3k})^\top$ are regression coefficients for the `age`, `year` and `surgery` covariates, respectively. We assume prior independence and specify prior marginal based on non-informative distributions commonly employed in the literature. The β's follow a N(0,0.001), while λ's and α's follow a Gamma(0.01,0.01) and a Un(0,10), respectively.

5.3 | Model implementation

We have created a design matrix `X` with the covariates `age`, `year` and `surgery`, and defined a `time3` $= t - t_{12}$ variable according to semi-Markovian specification:

```
R> X <- model.matrix(~ age + year + surgery, data = heart2)
R> X <- X[, -1] # Remove intercept
R> time3 <- heart2$times2
R> time3[time3 == 0] <- 0.0001
```

The values of `time3` equal to zero have been replaced by 0.0001 to avoid computational problems when calculating $(t - t_{12})^{\alpha_3 - 1}$.

Listing 6 shows a generic implementation of an illness-death model in BUGS syntax using *heart2* data.

```
model {
  for(i in 1:n) {
    # Linear predictor
    elinpred[i, 1] <- exp(inprod(beta[, 1], X[i, ]))
    elinpred[i, 2] <- exp(inprod(beta[, 2], X[i, ]))
    elinpred[i, 3] <- exp(inprod(beta[, 3], X[i, ]))
    # Log-hazard functions
    logHaz[i, 1] <- log(lambda[1] * alpha[1] * pow(t1[i], alpha[1] - 1) *
      elinpred[i, 1])
    logHaz[i, 2] <- log(lambda[2] * alpha[2] * pow(t2[i], alpha[2] - 1) *
      elinpred[i, 2])
    logHaz[i, 3] <- log(lambda[3] * alpha[3] * pow(t3[i], alpha[3] - 1) *
      elinpred[i, 3])
    # Log-survival functions
    logSurv[i, 1] <- -lambda[1] * pow(t1[i], alpha[1]) * elinpred[i, 1]
    logSurv[i, 2] <- -lambda[2] * pow(t2[i], alpha[2]) * elinpred[i, 2]
    logSurv[i, 3] <- -lambda[3] * pow(t3[i], alpha[3]) * elinpred[i, 3]
    # Definition of the log-likelihood using zeros trick
    phi[i] <- 100000 - inprod(event[i, ], logHaz[i, ]) - sum(logSurv[i, ])
    zeros[i] ~ dpois(phi[i])
  }
  # Prior distributions
  for(k in 1:3) {
    for(l in 1:Nbetas) {
      beta[l, k] ~ dnorm(0, 0.001)
    }
    lambda[k] ~ dgamma(0.01, 0.01)
    alpha[k] ~ dunif(0, 10)
  }
}
```

Listing 6: Illness-death model in BUGS syntax (file named as **IllDeath.txt**)

Once the variables have been defined, a list with all the elements required in the model is created:

```
R> d.jags <- list(n = nrow(heart2), t1 = heart2$times1, t2 = heart2$time, t3 =
time3,
+ X = X, event = event, zeros = rep(0, nrow(heart2)), Nbetas = ncol(X))
```

The initial values for each illness-death model parameter are passed to JAGS using a function that returns a list of random values:

```
R> i.jags <- function() {
+ list(beta = matrix(rnorm(3 * ncol(X)), ncol = 3), lambda = runif(3), alpha =
runif(3))
+ }
```

The vector of monitored/saved parameters is:

```
R> p.jags <- c("beta", "alpha", "lambda")
```

Next, the JAGS model is compiled:

```
R> library("rjags")
R> m5 <- jags.model(data = d.jags, file = "IllDeath.txt", inits = i.jags, n.chains
= 3)
```

We now run the model for 1000 burn-in simulations:

```
R> update(m5, 1000)
```

Finally, the model is run for 10 000 additional simulations to keep one in 10 so that a proper thinning is done:

```
R> res <- coda.samples(m5, variable.names = p.jags, n.iter = 10000, n.thin = 10)
```

Similar to the first example (Section 2.1), numerical and graphical summaries of the model parameters can be obtained using the `summary` and `densplot` functions, respectively. Gelman and Rubin's convergence diagnostic can be calculated with the `gelman.diag` function, and the `traceplot` function provides a visual way to inspect sampling behavior and assesses mixing across chains and convergence.

Next, simulations from the three Markov chains are merged together for inference:

```
R> result <- as.mcmc(do.call(rbind, res))
```

The posterior samples of each parameter are obtained by:

```
R> alpha1 <- result[,1]; alpha2 <- result[,2]; alpha3 <- result[,3]
R> beta11 <- result[,4]; beta21 <- result[,5]; beta31 <- result[,6]
R> beta12 <- result[,7]; beta22 <- result[,8]; beta32 <- result[,9]
R> beta13 <- result[,10]; beta23 <- result[,11]; beta33 <- result[,12]
R> lambda1 <- result[,13]; lambda2 <- result[,14]; lambda3 <- result[,15]
```

Table 4 shows posterior summaries for the illness-death model parameters using *heart2* data.

As discussed in Section 5, the transition probabilities (16)-(21) are the most appropriate way to analyze the evolution of each state over time. The implementation of the posterior distribution for these transition probabilities requires auxiliary functions, similar to the calculation of the cumulative incidence function in Section 4. To avoid unnecessary repetitions on how to calculate these quantities of interest, we will omit their implementation here. However, the code to reproduce Figure 5 is available in Appendix A1.

6 | FRAILTY MODELS

Regression models include measurable covariates to improve the knowledge of the relevant failure times. However, in most survival studies, there are also sources of heterogeneity that are not known or measurable. These elements are known in the statistical framework as *random effects*, but in the context of survival models they are the *frailty* elements.[72] They can approach individual characteristics as well as heterogeneity in groups or clusters.[8]

The most popular type of frailty models is the multiplicative shared-frailty model. It is a generalization of the Cox regression model introduced by Clayton[73] and extensively studied in Hougaard.[74] Let T_i the survival time for each individual in group i with hazard function described by:

$$h_i(t|h_0, \boldsymbol{\beta}, w_i) = w_i \; h_0(t) \exp\left\{\boldsymbol{x}_i^\top \boldsymbol{\beta}\right\}, \tag{25}$$

where w_i is the frailty term associated to group i. The usual probabilistic model for the frailty term is a gamma distribution with mean equal to one for identifiability purposes but also the positive stable and log-normal distributions can be considered.[75] In addition, a unity mean can be considered as a neutral frontier because frailty values greater (lower) than one increases (decreases) the individual risk. An alternative way of incorporating a frailty term in the hazard function is

Parameter	Mean	SD	2.5%	50%	97.5%	P(· > 0| data)
From waiting list to heart transplant (h_{12})						
β_{11}	0.048	0.015	0.020	0.047	0.077	1.000
β_{21}	0.004	0.069	−0.130	0.004	0.140	0.521
β_{31}	0.220	0.321	−0.440	0.229	0.826	0.760
λ_1	0.042	0.016	0.018	0.039	0.079	1.000
α_1	0.778	0.069	0.645	0.777	0.916	1.000
From waiting list to death (h_{13})						
β_{12}	−0.001	0.018	−0.034	−0.002	0.035	0.462
β_{22}	−0.243	0.115	−0.476	−0.241	−0.027	0.013
β_{32}	−0.785	0.672	−2.223	−0.736	0.391	0.109
λ_2	0.101	0.047	0.034	0.092	0.217	1.000
α_2	0.379	0.058	0.272	0.376	0.500	1.000
From heart transplant to death (h_{23})						
β_{13}	0.055	0.022	0.014	0.055	0.099	0.997
β_{23}	−0.008	0.094	−0.196	−0.007	0.174	0.470
β_{33}	−0.986	0.469	−1.973	−0.961	−0.129	0.011
λ_3	0.034	0.021	0.008	0.029	0.087	1.000
α_3	0.602	0.074	0.463	0.598	0.756	1.000

TABLE 4 Posterior summaries for the illness-death model parameters

via an additive element as follows:

$$h_i(t|h_0, \boldsymbol{\beta}, b_i) = h_0(t) \exp\left\{\boldsymbol{x}_i^\top \boldsymbol{\beta} + b_i\right\},$$

where now b_i's are commonly assumed as normally distributed with zero mean and unknown variance.

The Bayesian framework deals with frailty models in a conceptually simpler way than the frequentist one due to the Bayesian probability conception, introduced in Section 1. Hence, the inclusion of randomness through frailties in a Bayesian perspective does not add any conceptual complexity because the information regarding the risk function is expressed in probabilistic terms through its posterior distribution. Survival modeling with frailty terms is a wide issue of research that applies to all type of regression, competing risks, multivariate survival models, and so on, and play a special role in joint models as we will discuss later.

6.1 | *kidney* dataset

We consider a kidney infection dataset, referred to as *kidney*. It is available from the `frailtyHL` package:[76]

```
R> library("frailtyHL")
R> data("kidney")
R> str(kidney)
'data.frame': 76 obs. of 10 variables:
 $ id : num 1 1 2 2 3 3 4 4 5 5 ...
 $ time : num 8 16 23 13 22 28 447 318 30 12 ...
 $ status : num 1 1 1 0 1 1 1 1 1 1 ...
 $ age : num 28 28 48 48 32 32 31 32 10 10 ...
 $ sex : num 1 1 2 2 1 1 2 2 1 1 ...
 $ disease: Factor w/ 4 levels "Other","GN","AN",..: 1 1 2 2 1 1 1 1 1 1 ...
 $ frail : num 2.3 2.3 1.9 1.9 1.2 1.2 0.5 0.5 1.5 1.5 ...
```

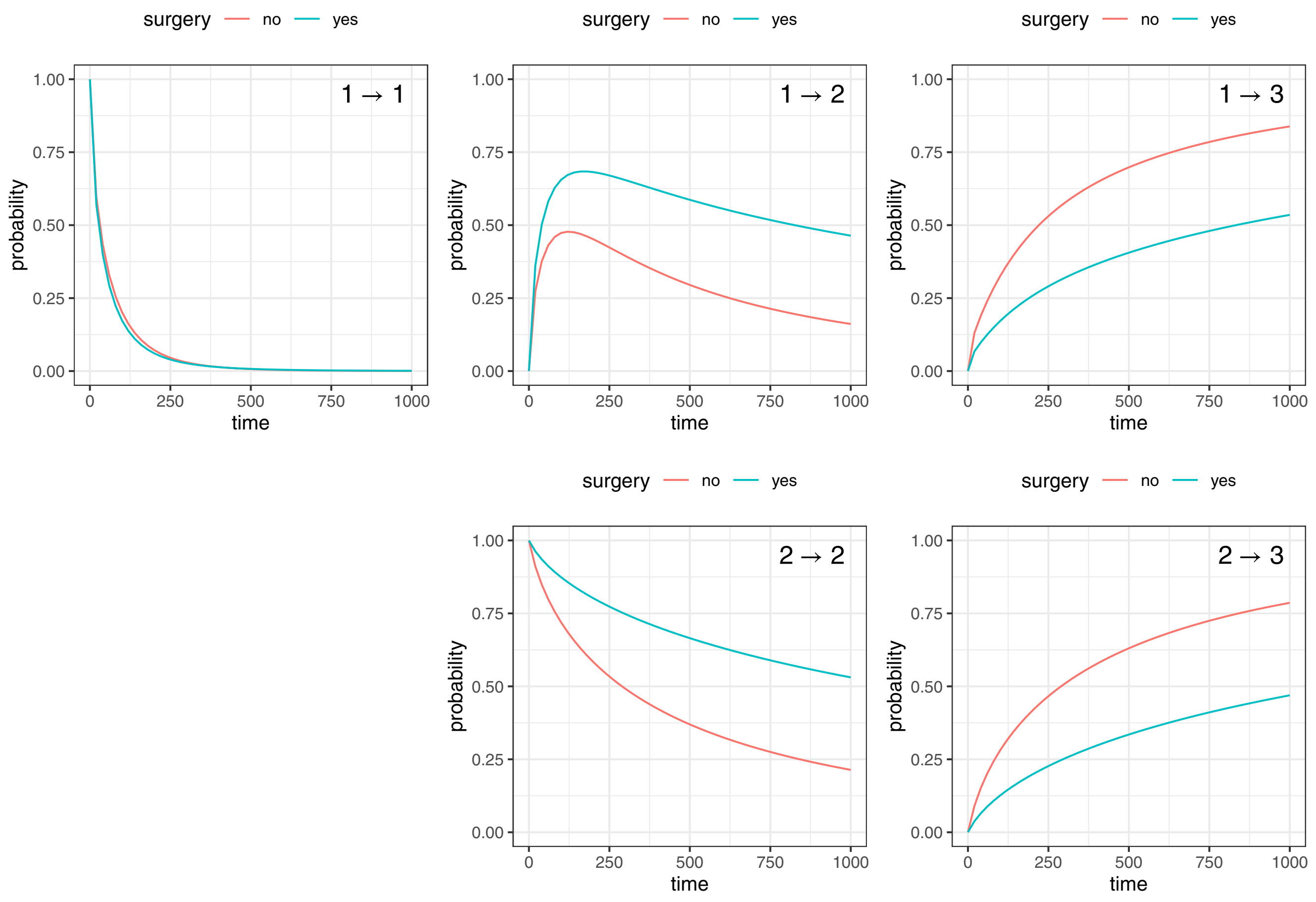

FIGURE 5 Posterior mean of all probability transitions from the illness-death model (22)-(24) for patients with and without prior bypass surgery and median values of `age` and `year`. Graphics on the top correspond to transitions from the initial state in the waiting list. Posterior probabilities for transitions from transplant (bottom row) assume the median time $T_{12} = 26$ recorded from the waiting list to heart transplant [Colour figure can be viewed at wileyonlinelibrary.com]

```
 $ GN  : num 0 0 1 1 0 0 0 0 0 0 ...
 $ AN  : num 0 0 0 0 0 0 0 0 0 0 ...
 $ PKD : num 0 0 0 0 0 0 0 0 0 0 ...
```

This dataset consists of times to the first and second recurrences of infection in 38 kidney patients using a portable dialysis machine. Infections can occur at the location of insertion of the catheter. The catheter is later removed if infection occurs and can be removed for other reasons, in which case the observation is censored.[77] The following variables were repeated twice for each patient:

- `id`: patient number.
- `time`: time (in days) from insertion of the catheter to infection in kidney patients using portable dialysis machine.
- `status`: censoring indicator (1: if patient is uncensored; 0: otherwise).
- `age`: age (in years) of each patient.
- `sex`: sex of each patient (1: if patient is male; 2: if patient is female).
- `disease`: disease type (GN; AN; PKD; Other).
- `frail`: frailty estimate from original paper.
- `GN`: indicator for disease type GN.

- AN: indicator for disease type AN.
- PKD: indicator for disease type PKD.

Note that the status variable equal to zero in any of the recurrences means that until that moment of observation there was no infection. Analogous to the multi-state example, we will use a modeling based on the *clock-reset approach*,[1] in which time starts at zero again after each recurrence.

6.2 | Model implementation

Time, in days, from insertion of a catheter in patient ith to infection is modeled trough a PH specification with multiplicative frailties:

$$h_i(t|h_0, \boldsymbol{\beta}, w_i) = w_i\ h_0(t) \exp\{\beta_2 \texttt{sex}\}, \tag{26}$$

where $w_i \sim \text{Gamma}(\psi, \psi)$ represents the frailty term for each individual and is set with unity mean in order for the parameters of the model to be identifiable;[8] $h_0(t) = \lambda\ \alpha\ t^{\alpha-1}$ is specified as a Weibull baseline hazard function, where α and $\lambda = \exp(\beta_1)$ are the shape and scale parameters, respectively; and β_2 is the regression coefficient for the sex covariate. As the purpose of this modelling is illustrative, the other covariates will not be considered. We assume prior independence and specify prior marginal based on non-informative distributions commonly employed in the literature. The β's follow a N(0,0.001), while α and ψ follow a Un(0,10) and a Gamma(0.01,0.01), respectively.

Our variable of interest is time, which represents the time from insertion of the catheter to infection. The status covariate plays an important role in the codification of the survival and censoring times:

```
R> # Number of patients and catheters
R> n <- length(unique(kidney$id))
R> J <- 2
R> # Survival and censoring times
R> time <- kidney$time
R> cens <- time
R> time[kidney$status == 0] <- NA # Censored
R> is.censored <- as.numeric(is.na(time))
R> # Matrix format
R> time <- matrix(time, n, J, byrow = TRUE)
R> cens <- matrix(cens, n, J, byrow = TRUE)
R> is.censored <- matrix(is.censored, n, J, byrow = TRUE)
```

Without loss of generality, we have created a design matrix X with the sex covariate:

```
R> sex <- kidney$sex[seq(1, 2 * n, 2)] - 1 # Reference = male
R> X <- model.matrix(~ sex)
```

Listing 7 shows a generic implementation of a frailty model in BUGS syntax using *kidney* data.

```
model {
  for(i in 1:n) {
    for(j in 1:J) {
      # Survival and censoring times
      is.censored[i, j] ~ dinterval(time[i, j], cens[i, j])
      time[i, j] ~ dweib(alpha, lbd[i, j])
      log(lbd[i, j]) <- inprod(beta[], X[i, ]) + log(w[i])
    }
    # Multiplicative frailties
    w[i] ~ dgamma(psi, psi)
  }
  # Prior distributions
  for(l in 1:Nbetas) {
    beta[l] ~ dnorm(0, 0.001)
  }
  alpha ~ dunif(0, 10)
  psi ~ dgamma(0.01, 0.01)
  # Derived quantity
  lambda <- exp(beta[1])
}
```

Listing 7: Frailty model in BUGS syntax (file named as **Frailty.txt**)

6.3 | Model estimation: JAGS from R

Once the variables have been defined, a list with all the elements required in the model is created:

```
R> d.jags <- list(n = n, J = J, time = time, cens = cens, X = X,
+ is.censored = is.censored, Nbetas = ncol(X))
```

The initial values for each frailty model parameter are passed to JAGS using a function that returns a list of random values:

```
R> i.jags <- function() { list(beta = rnorm(ncol(X)), alpha = runif(1), psi =
runif(1)) }
```

The vector of monitored/saved parameters is:

```
R> p.jags <- c("beta", "alpha", "lambda", "psi", "w")
```

Next, the JAGS model is compiled:

```
R> library("rjags")
R> m6 <- jags.model(data = d.jags, file = "Frailty.txt", inits = i.jags, n.chains =
3)
```

We now run the model for 10 000 burn-in simulations:

```
R> update(m6, 10000)
```

Finally, the model is run for 100 000 additional simulations to keep one in 100 so that a proper thinning is done:

```
R> res <- coda.samples(m6, variable.names = p.jags, n.iter = 100000, thin = 100)
```

Similar to the first example (Section 2.1), numerical and graphical summaries of the model parameters can be obtained using the `summary` and `densplot` functions, respectively. Gelman and Rubin's convergence diagnostic can be calculated with the `gelman.diag` function, and the `traceplot` function provides a visual way to inspect sampling behavior and assesses mixing across chains and convergence.

Next, simulations from the three Markov chains are merged together for inference:

```
R> result <- as.mcmc(do.call(rbind, res))
```

Parameter	Mean	SD	2.5%	50%	97.5%	P(· > 0| data)
β_2 (sex)	−1.908	0.555	−3.064	−1.889	−0.876	0.000
α	1.233	0.167	0.929	1.222	1.592	1.000
λ	0.019	0.012	0.004	0.017	0.050	1.000
ψ	2.417	2.101	0.779	1.878	7.844	1.000

TABLE 5 Posterior summaries for the frailty model parameters

The posterior samples of each parameter are obtained by:

```
R> alpha <- result[,1]; beta2 <- result[,3]; lambda <- result[,4]
R> psi <- result[,5]; w <- result[,6:ncol(result)]
```

Table 5 shows posterior summaries for the frailty model parameters using *kidney* data.

The individual survival curve based on posterior samples is a relevant information in this type of studies. Since we have taken a clock-reset approach, this curve represents the (posterior mean of the) probability of infection from any catheter insertion at each time considering the two replicates per patient. So, we can summarize the posterior distribution of the individual survival curve in a grid of points as follows:

```
R> grid <- 1000
R> time <- seq(0, max(kidney$time), len = grid)
R> surv <- matrix(NA, n, grid)
R> for(i in 1:n) {
+ for(k in 1:grid) {
+ surv[i, k] <- mean(exp(-w[i] * lambda * exp(beta2 * sex[i]) * time[k]^alpha))
+ }
+ }
```

Next, we can differentiate the survival curves by sex (code below). Figure 6 shows such curves for all patients in the *kidney* data.

```
R> library("ggplot2")
R> sex.col <- sex
R> sex.col[sex == 0] <- "male"
R> sex.col[sex == 1] <- "female"
R> df <- data.frame(time = rep(time, n), survival = c(t(surv)),
+ patient = rep(1:n, each = grid), sex = rep(sex.col, each = grid))
R> ggplot(data = df, aes(x = time, y = survival, group = patient, colour = sex)) +
+ geom_line() + theme_bw() + theme(legend.position = "top")
```

7 | JOINT MODELS OF LONGITUDINAL AND SURVIVAL DATA

Joint modeling of longitudinal and time-to-event data is an increasingly productive area of statistical research that examines the association between longitudinal and survival processes.[78] It enhances survival modeling with the inclusion of internal time-dependent covariates as well as longitudinal modeling by allowing for the inclusion of non-ignorable dropout mechanisms through survival tools. Joint models were introduced during the 90s[79-82] and since then, have been applied to a great variety of studies mainly in epidemiological and biomedical areas.

Bayesian joint models assume a full joint distribution for the longitudinal (y) and the survival processes (s) as well as the subject-specific random effects vector ($\boldsymbol{b}$) and the parameters and hyperparameters ($\boldsymbol{\theta}$) of the model.[83] Usually, they can be defined as follows:

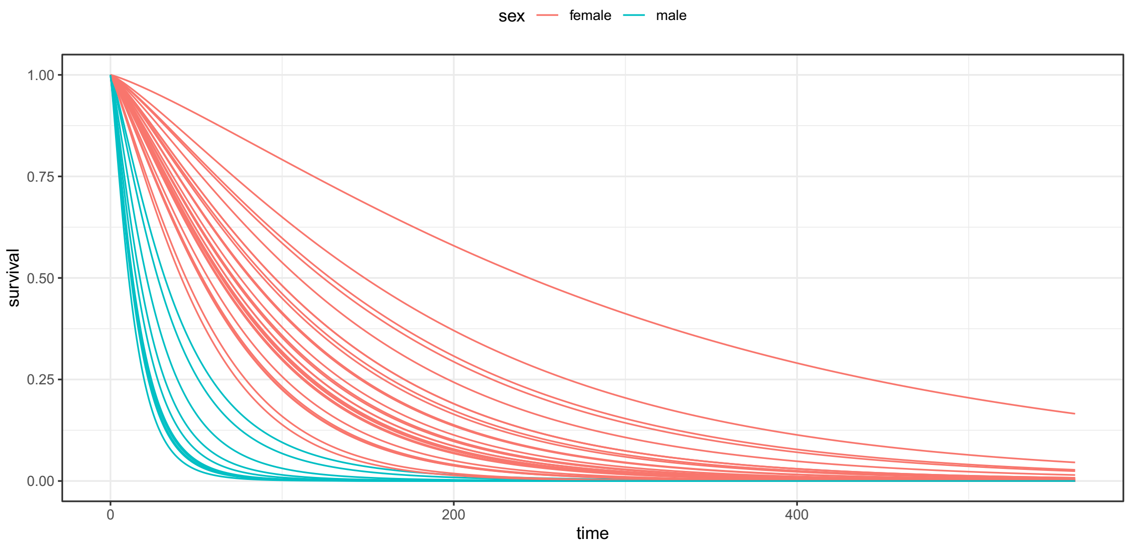

FIGURE 6 Posterior mean of the individual survival function from the frailty model (26) [Colour figure can be viewed at wileyonlinelibrary.com]

$$f(y,s,\boldsymbol{b},\boldsymbol{\theta}) = f(y,s|\boldsymbol{b},\boldsymbol{\theta})\ f(\boldsymbol{b}|\boldsymbol{\theta})\ \pi(\boldsymbol{\theta}), \tag{27}$$

which factorizes as the product of the joint conditional distribution $f(y,s|\boldsymbol{b},\boldsymbol{\theta})$, the conditional distribution $f(\boldsymbol{b}|\boldsymbol{\theta})$ of the random effects, and the prior distribution $\pi(\boldsymbol{\theta})$. There are different proposals for the specification of the conditional distribution $f(y,s|\boldsymbol{b},\boldsymbol{\theta})$. The most popular approaches are the share-parameter models and the joint latent class models.

Shared-parameter models are a type of joint models where the longitudinal and time-to-event processes are connected by means of a common set of subject-specific random effects. These models make possible to quantify both the population and individual effects of the underlying longitudinal outcome on the risk of an event and allow to obtain individualized time-dynamic predictions.[84-86] In particular, this approach postulates conditional independence between the longitudinal and survival processes given the random effects and the parameters:

$$f(y,s|\boldsymbol{b},\boldsymbol{\theta}) = f(y|\boldsymbol{b},\boldsymbol{\theta})f(s|\boldsymbol{b},\boldsymbol{\theta}).$$

The joint latent class model is based on finite mixtures.[87] Heterogeneity among the individuals is classified into a finite number G of homogeneous latent clusters which share the same longitudinal trajectory and the same risk function. Both elements are also conditionally independent within the subsequent latent group as follows:

$$f(y,s|L=g,\boldsymbol{b},\boldsymbol{\theta}) = f(y|L=g,\boldsymbol{b},\boldsymbol{\theta})f(s|L=g,\boldsymbol{\theta}), \tag{28}$$

where L is a random variable that quantifies the probability that individuals with certain characteristics belong to each of the groups, usually modeled by means of a multinomial logistic model. These models are possibly the most complex. Predicting observations from models with random effects is not easy. When the longitudinal submodel $f(y|L=g,\boldsymbol{b},\boldsymbol{\theta})$ in (28) includes random effects associated to individuals, predicting longitudinal observations will be computationally complex since there is a need to integrate out the random effects. However, when random effects are not used to model the longitudinal process, the joint latent class model may be particularly suited for prediction problems.[87] On the other hand, the prediction of survival times does seem to be computationally simpler because the conditional distribution $f(s|L=g,\boldsymbol{\theta})$ does not depend on individual random effects.

All these proposals account for a particular type of conditional independence between the longitudinal and the survival processes which facilitates the modeling into longitudinal and survival submodels with various types of connectors. This general structure allows any type of modeling for the survival process such as frailty survival regression models, competing risks with frailties, cure models with frailties as well as linear mixed models or generalized linear mixed models for the longitudinal process.[88-91] See Armero[92] for a short review on Bayesian joint models up to date.

7.1 | *prothro* dataset

We consider a liver cirrhosis dataset, referred to as *prothro* (longitudinal information) and *prothros* (survival information). It is available from the JMbayes package:[22]

```
R> library("JMbayes")
R> data("prothro")
R> data("prothros")
R> str(prothro); str(prothros)
'data.frame': 2968 obs. of 9 variables:
 $ id : num 1 1 1 2 2 2 2 2 2 2 ...
 $ pro : num 38 31 27 51 73 90 64 54 58 90 ...
 $ time : num 0 0.244 0.381 0 0.687 ...
 $ treat: Factor w/ 2 levels "placebo","prednisone": 2 2 2 2 2 2 2 2 2 2 ...
 $ Time : num 0.413 0.413 0.413 6.754 6.754 ...
 $ start: num 0 0.244 0.381 0 0.687 ...
 $ stop : num 0.244 0.381 0.413 0.687 0.961 ...
 $ death: num 1 1 1 1 1 1 1 1 1 1 ...
 $ event: num 0 0 1 0 0 0 0 0 0 0 ...
'data.frame': 488 obs. of 4 variables:
 $ id : num 1 2 3 4 5 6 7 8 9 10 ...
 $ Time : num 0.413 6.754 13.394 0.794 0.75 ...
 $ death: num 1 1 0 1 1 1 1 0 0 1 ...
 $ treat: Factor w/ 2 levels "placebo","prednisone": 2 2 2 2 2 2 1 1 1 1 ...
```

These datasets are part of a placebo-controlled randomized trial on 488 liver cirrhosis patients, where the longitudinal observations of a biomarker (prothrombin) are recorded.[93] For our illustrative purpose, only the following variables are relevant:

- `id`: patient number.
- `pro`: prothrombin measurements.
- `time`: time points at which the prothrombin measurements were taken.
- `treat`: randomized treatment (placebo or prednisone).
- `Time`: time (in years) from the start of treatment until death or censoring.
- `death`: censoring indicator (1: if patient is died; 0: otherwise).

7.2 | Model implementation

We assume a shared-parameter model specified by a linear mixed-effects model[94] for the longitudinal response and a Cox model for the survival part. The linear mixed-effects model which describes the subject-specific prothrombin evolution of individual i over time is given by:

$$y_i(t) = \beta_{L1} + b_{i1} + (\beta_{L2} + b_{i2})\, t + \beta_{L3}\texttt{treat} + \epsilon_i(t), \quad i = 1, \dots, n, \tag{29}$$

where $y_i(t)$ represents the prothrombin value at time t for individual i; β_{L1} and β_{L2} are fixed effects for intercept and slope, respectively, with b_{i1} and b_{i2} being the respective individual random effects; β_{L3} is the regression coefficient for `treat` covariate; and $\epsilon_i(t)$ is a measurement error for individual i at time t. We assume that the individual random effects, $\boldsymbol{b}_i = (b_{i1}, b_{i2})^\top$, given Σ follow a joint bivariate normal distribution with mean vector $(0,0)^\top$ and variance-covariance matrix Σ, and that the errors are conditionally i.i.d. as $(\epsilon_i(t)|\tau) \sim \mathrm{N}(0, \tau)$, where τ represents the error precision (defined as one divided by the variance). Random effects and error terms were assumed mutually independent.

Survival time for individual i is modeled from a PH specification which includes in the exponential term the random effects b_{i1} and b_{i2} in (29) as follows:

$$h_i(t|h_0, \boldsymbol{\beta}_S, \gamma, \boldsymbol{b}_i) = h_0(t) \exp\left\{\beta_{S2}\texttt{treat} + \gamma(b_{i1} + b_{i2}t)\right\}, \tag{30}$$

with $h_0(t) = \lambda\ \alpha\ t^{\alpha-1}$ specified as a Weibull baseline hazard function, where α and $\lambda = \exp(\beta_{S1})$ are the shape and scale parameters, respectively; β_{S2} is the regression coefficient for the `treat` covariate; and γ is an association parameter that measure the strength of the link between the random effects associated to individual i of the longitudinal submodel and their risk of death at time t.

We assume prior independence and specify prior marginal distributions based on non-informative distributions commonly employed in the literature. β_L's, β_S' and γ follow a N(0,0.001), α and τ follow a Gamma(0.01,0.01), and Σ follows an Inv-Wishart(V, r), where V is a 2×2 identity matrix and $r = 2$ is the degrees-of-freedom parameter. The position of the inverse-Wishart distribution in parameter space is specified by V, while r set the certainty about the prior information in the scale matrix.[95] The larger the r, the higher the certainty about the information in V, and the more informative is the distribution. Hence, in our application, the least informative specification then results when $r = 2$ (number of random effects), which is the lowest possible number of r. Additionally, V as an identity matrix has the appealing feature that each of the correlations in Σ has, marginally, a uniform prior distribution.[28]

Our variable of interest is `Time` (*prothros* file), which represents the time from the start of treatment until death or censoring. The `death`=1 covariate indicates an uncensored time:

```
R> # Number of patients and number of longitudinal observations per patient
R> n <- nrow(prothros)
R> M <- table(prothro$id)
R> # Survival and censoring times
R> Time <- prothros$Time
R> death <- prothros$death
```

The (log)prothrombin observations and their respective measurement times (*prothro* file) have been rearranged in matrix format:

```
R> # Longitudinal information in matrix format
R> time <- matrix(NA, n, max(M))
R> log.proth <- matrix(NA, n, max(M))
R> count <- 1
R> for(i in 1:n) {
+ log.proth[i, 1:M[i]] <- log(prothro$pro[count:(M[i] + count - 1)])
+ time[i, 1:M[i]] <- prothro$time[count:(M[i] + count - 1)]
+ count <- count + M[i]
+ }
```

We have created the survival design matrix composed of an intercept and a treatment variable:

```
R> treat <- as.numeric(prothros$treat) - 1 # Reference = placebo
R> XS <- model.matrix(~ treat) # Fixed effects
```

We have split the longitudinal design matrix into two parts, `XL` (fixed effects) and `ZL` (random effects):

```
R> XL <- array(1, dim = c(n, max(M), 3)) # Fixed effects
R> XL[, , 2] <- time; XL[, , 3] <- treat
R> ZL <- array(1, dim = c(n, max(M), 2)) # Random effects
R> ZL[, , 2] <- time
```

The survival function for individual i, $S_i(t|h_0, \boldsymbol{\beta}_S, \gamma, \boldsymbol{b}_i) = \exp\left\{-\int_0^t\ h_i(u|h_0, \boldsymbol{\beta}_S, \gamma, \boldsymbol{b}_i)\ \mathrm{d}u\right\}$can be efficiently approximated using some Gaussian quadrature method, available from the `statmod` package.[96] For our analysis, we have used 15-point Gauss-Legendre quadrature rule, as is done in Armero et al:[83]

```
R> # Gauss-Legendre quadrature (15 points)
R> library("statmod")
R> glq <- gauss.quad(15, kind = "legendre")
R> xk <- glq$nodes # Nodes
R> wk <- glq$weights # Weights
R> K <- length(xk) # K-points
```

Listing 8 shows a generic implementation of a joint model in BUGS syntax using *prothro*/*prothros* data.

```
model {
  for(i in 1:n) {
    # Longitudinal observations
    for(j in 1:M[i]) {
       log.proth[i, j] ~ dnorm(mu[i, j], tau)
          mu[i, j] <- inprod(betaL[], XL[i, j, ]) + inprod(b[i, ], ZL[i, j, ])
    }
    # Survival and censoring times
    # Hazard function at integration points
    for(j in 1:K) {
       haz[i, j] <- alpha * pow(Time[i] / 2 * (xk[j] + 1), alpha - 1) *
         exp(inprod(betaS[], XS[i, ]) +
           gamma * (b[i, 1] + b[i, 2] * (Time[i] / 2 * (xk[j] + 1))))
    }
    # Log-survival function with Gauss-Legendre quadrature
    logSurv[i] <- -Time[i] / 2 * inprod(wk, haz[i, ])
    # Definition of the survival log-likelihood using zeros trick
    phi[i] <- 100000 - death[i] * log(haz[i, K]) - logSurv[i]
    zeros[i] ~ dpois(phi[i])
    # Random effects
    b[i, 1:Nb] ~ dmnorm(mub[], Omega[, ])
  }
  # Prior distributions
        for(l in 1:NbetasL) {
    betaL[l] ~ dnorm(0, 0.001)
  }
  for(l in 1:NbetasS) {
    betaS[l] ~ dnorm(0, 0.001)
  }
  gamma ~ dnorm(0, 0.001)
  alpha ~ dgamma(0.01, 0.01)
  tau ~ dgamma(0.01, 0.01)
  Omega[1:Nb, 1:Nb] ~ dwish(V[, ], Nb)
  # Derived quantity
  lambda <- exp(betaS[1])
  sigma <- sqrt(1/tau)
  Sigma[1:Nb, 1:Nb] <- inverse(Omega[, ])
}
```

Listing 8: Joint model in BUGS syntax (file named as **JM.txt**)

Once the variables have been defined, a list with all the elements required in the model is created:

```
R> d.jags <- list(n = n, M = M, Time = Time, XS = XS, log.proth = log.proth, XL =
XL, ZL = ZL,
+ XS = XS, death = death, mub = rep(0, 2), V = diag(1, 2), Nb = 2, zeros = rep(0,
n),
+ NbetasL = dim(XL)[3], NbetasS = ncol(XS), K = length(xk), xk = xk, wk = wk)
```

The variables `mub` and `V` represent, respectively, the mean of the random effects normally distributed and the scale matrix of the Wishart distribution which models the precision of the random effects.

The initial values for each joint model parameter are passed to JAGS using a function that returns a list of random values:

```
R> i.jags <- function() {
+ list(betaS = rnorm(ncol(XS)), gamma = rnorm(1), alpha = runif(1),
+ betaL = rnorm(dim(XL)[3]), tau = runif(1), Omega = diag(runif(2)))
+ }
```

The vector of monitored/saved parameters is:

```
R> p.jags <- c("betaS", "gamma", "alpha", "lambda", "betaL", "sigma", "Sigma", "b")
```

Next, the JAGS model is compiled:

```
R> library("rjags")
R> m7 <- jags.model(data = d.jags, file = "JM.txt", inits = i.jags, n.chains = 3)
```

We now run the model for 1000 burn-in simulations:

```
R> update(m7, 1000)
```

Finally, the model is run for 10 000 additional simulations to keep one in 10 so that a proper thinning is done:

```
R> res <- coda.samples(m7, variable.names = p.jags, n.iter = 10000, thin = 10)
```

Similar to the first example in Section 2.1, numerical and graphical summaries of the model parameters can be obtained using the `summary` and `densplot` functions, respectively. Gelman and Rubin's convergence diagnostic can be calculated with the `gelman.diag` function, and the `traceplot` function provides a visual way to inspect sampling behavior and assesses mixing across chains and convergence.

Next, simulations from the three Markov chains are merged together for inference:

```
R> result <- as.mcmc(do.call(rbind, res))
```

The posterior samples of each parameter are obtained by:

```
R> Sigma2.11 <- result[,1]; Sigma2.12 <- result[,2]; Sigma2.22 <- result[,4]
R> alpha <- result[,5]; b1 <- result[,6:(n+5)]; b2 <- result[,(n+6):(2*n+5)]
R> betaL1 <- result[,(2*n+6)]; betaL2 <- result[,(2*n+7)]; betaL3 <-
result[,(2*n+8)]
R> betaS2 <- result[,(2*n+10)]; gamma <- result[,(2*n+11)]
R> lambda <- result[,(2*n+12)]; sigma <- result[,(2*n+13)]
```

Table 6 shows posterior summaries for the joint model parameters using *prothro*/*prothros* data.

Most of the parameters are interpreted similar to the ones in previous examples. However, the association parameter, γ, plays an important role in this type of models. In our illustration, the posterior mean of γ is negative, −2.269, and $P(\gamma > 0|\text{data}) = 0$, indicating a strong negative association of the prothrombin measurements with respect to vital status. In other words, a negative value for γ means that low values or decreasing trends of prothrombin increase the risk of death.

TABLE 6 Posterior summaries for the joint model parameters

Parameter	Mean	SD	2.5%	50%	97.5%	P($\cdot > 0$| data)
β_{S2} (`treat`)	0.071	0.137	−0.198	0.069	0.342	0.695
γ (`assoc`)	−2.276	0.179	−2.620	−2.275	−1.922	0.000
λ	0.187	0.023	0.146	0.186	0.235	1.000
α	0.929	0.051	0.833	0.928	1.033	1.000
β_{L1} (`intercept`)	4.275	0.022	4.232	4.277	4.315	1.000
β_{L2} (`slope`)	−0.002	0.008	−0.017	−0.002	0.013	0.395
β_{L3} (`treat`)	−0.098	0.030	−0.158	−0.098	−0.040	0.000
σ	0.258	0.004	0.250	0.258	0.265	1.000
Σ_{11}	0.099	0.008	0.084	0.098	0.115	1.000
Σ_{22}	0.013	0.002	0.011	0.013	0.017	1.000
Σ_{12}	−0.004	0.003	−0.010	−0.003	0.002	0.114

8 | CONCLUSIONS

The analysis of time until an event of interest requires a suitable and flexible modeling and has applications in several applied fields. The BUGS language offers the opportunity of easily use and adapt Bayesian hierarchical models without the need to manually implement MCMC methods. Hence, this article has summarized some of the most popular survival models and has focused on the Bayesian paradigm to make the inferential procedure. Furthermore, for each of the models proposed, we have provided the codes in BUGS syntax, so that model can be fit with the support of the `rjags` package from the R language.

We have discussed all the implementation details of the following Bayesian survival models: AFT, PH, mixture cure, competing risks, multi-state, frailty, and joint models of longitudinal and survival data. Moreover, the computation of quantities of interest derived from posterior samples has been provided as well as some graphs that assist in the interpretation of results and decision making. This article has also briefly presented other Bayesian R packages that handle time-to-event data.

In conclusion, we hope this article will encourage researchers who use survival models make their analyses based on the Bayesian paradigm from the BUGS codes we have provided and easily adapt them to novel settings. In addition, the descriptions of the survival models provided herein could also be used as a guidance to implement these models using other similar languages such as, for example, Stan.[24]

ACKNOWLEDGMENTS
We would like to thank the reviewers and editors for their positive suggestions and criticism. The paper has been greatly improved by the thorough and rigorous work of all of them. In addition, we would also like to thank the organizers of the Bayes Pharma Conference 2017 (held in Albacete, Spain) who invited us to give a Bayesian survival short course. This paper is based on the materials prepared for that course.

CONFLICT OF INTEREST
The authors declare no potential conflict of interests.

ORCID
Danilo Alvares https://orcid.org/0000-0003-3764-0397
Virgilio Gómez-Rubio https://orcid.org/0000-0002-4791-3072

REFERENCES
1. Kleinbaum DG, Klein M. *Survival Analysis: A Self-Learning Text*. 3rd ed. New York, NY: Springer; 2012.
2. Collett D. *Modelling Survival Data in Medical Research*. 3rd ed. Boca Raton, FL: Chapman & Hall/CRC Press; 2015.
3. Kalbfleisch JD, Prentice RL. *The Statistical Analysis of Failure Time Data*. 2nd ed. Hoboken, NJ: John Wiley & Sons; 2002.

4. Klein JP, van Houwelingen HC, Ibrahim JG, Scheike TH. *Handbook of Survival Analysis*. 1st ed. Boca Raton, FL: Chapman & Hall/CRC Press; 2013.
5. Klein JP, Moeschberger ML. *Survival Analysis: Techniques for Censored and Truncated Data*. 2nd ed. New York, NY: Springer; 2003.
6. Ibrahim JG, Chen MH, Lakshminarayanan M, Liu GF, Heyse JF. Bayesian probability of success for clinical trials using historical data. *Stat Med*. 2015;34(2):249-264. https://doi.org/10.1002/sim.6339.
7. Psioda MA, Ibrahim JG. Bayesian clinical trial design using historical data that inform the treatment effect. *Biostatistics*. 2019;20(3):400-415. https://doi.org/10.1093/biostatistics/kxy009.
8. Ibrahim JG, Chen MH, Sinha D. *Bayesian Survival Analysis*. 1st ed. New York, NY: Springer; 2001.
9. Serrat C, Rué M, Armero C, et al. Frequentist and Bayesian approaches for a joint model for prostate Cancer risk and longitudinal prostate-specific antigen data. *J Appl Stat*. 2015;42(6):1223-1239. https://doi.org/10.1080/02664763.2014.999032.
10. Nasejje JB, Mwambi HG, Achia TNO. Understanding the determinants of under-five child mortality in Uganda including the estimation of unobserved household and community effects using both Frequentist and Bayesian survival Analysis approaches. *BMC Public Health*. 2015;15(1003):1-12. https://doi.org/10.1186/s12889-015-2332-y.
11. Renganathan V. Overview of Frequentist and Bayesian approach to survival analysis. *Appl Med Inform*. 2016;38(1):25-38.
12. Bogaerts K, Komárek A, Lesaffre E. *Survival Analysis with Interval-Censored Data: A Practical Approach with Examples in R, SAS, and BUGS*. 1st ed. Boca Raton, FL: Chapman & Hall/CRC Press; 2017.
13. Komárek A. bayesSurv: Bayesian survival regression with flexible error and random effects distributions. R package version 3.2; 2018. https://CRAN.R-project.org/package=bayesSurv.
14. Zhou H, Hanson T. spBayesSurv: Bayesian modeling and analysis of spatially correlated survival data. R package version 1.1.3; 2018. https://CRAN.R-project.org/package=spBayesSurv.
15. Wang X, Chen MH, Wang W, Yan J. Dynsurv: dynamic models for survival data. R package version 0.3–6; 2017. https://CRAN.R-project.org/package=dynsurv.
16. Pan C, Cai B, Wang L, Lin X. ICBayes: Bayesian semiparametric models for interval-censored data. R package version 1.1; https://CRAN.R-project.org/package=ICBayes: 2017.
17. Taylor BM, Rowlingson BS, Zheng Z. spatsurv: Bayesian spatial survival analysis with parametric proportional hazards models. R package version 1.2; 2018. https://CRAN.R-project.org/package=spatsurv.
18. Raftery A, Hoeting J, Volinsky C, Painter I, Yeung KY. BMA: Bayesian model averaging. R package version 3.18.9; 2018. https://CRAN.R-project.org/package=BMA.
19. Sharabiani MTA, Mahani AS. CFC: cause-specific framework for competing-risk analysis. R package version 1.1.2; 2019. https://CRAN.R-project.org/package=CFC.
20. Lee KH, Lee C, Alvares D, Haneuse S. SemiCompRisks: hierarchical models for parametric and semi-parametric analyses of semi-competing risks data. R package version 3.2; 2019. https://CRAN.R-project.org/package=SemiCompRisks.
21. Alvares D, Haneuse S, Lee C, Lee KH. SemiCompRisks: an R package for the analysis of independent and cluster-correlated semi-competing risks data. *R J*. 2019;11(1):376-400. https://doi.org/10.32614/rj-2019-038.
22. Rizopoulos D. JMbayes: joint modeling of longitudinal and time-to-event data under a bayesian approach. R package version 0.8–83; 2019. https://CRAN.R-project.org/package=JMbayes.
23. Umlauf N, Kneib T, Klein N. BayesX: R utilities accompanying the software package BayesX. R package version 0.3–1; 2019. https://CRAN.R-project.org/package=BayesX.
24. Carpenter B, Gelman A, Hoffman M, et al. Stan: a probabilistic programming language. *J Stat Softw*. 2017;76(1):1-32. https://doi.org/10.18637/jss.v076.i01.
25. Stan Development Team RStan: the R Interface to Stan. Stan; 2020. http://mc-stan.org/.
26. Gabry J, Ali I, Brilleman S, et al. rstanarm: Bayesian applied regression modeling via Stan. R package version 2.21.1; 2020. https://CRAN.R-project.org/package=rstanarm.
27. INLA Development Team The R-INLA Project. INLA; 2020. http://www.r-inla.org/.
28. Gelman A, Carlin JB, Stern HS, Dunson DB, Vehtari A, Rubin DB. *Bayesian Data Analysis*. 3rd ed. Boca Raton, FL: Chapman & Hall/CRC Press; 2013.
29. Gilks WR, Thomas A, Spiegelhalter DJ. A language and program for complex Bayesian modelling. *The Statistician*. 1994;43(1):169-177. https://doi.org/10.2307/2348941.
30. Plummer M. JAGS: a program for analysis of Bayesian graphical models using gibbs sampling. Paper presented at: Proceedings of the 3rd International Workshop on Distributed Statistical Computing; 2003:1-10; Vienna, Austria.
31. Plummer M. rjags: Bayesian graphical models using MCMC. R package version 4–8; 2018. https://CRAN.R-project.org/package=rjags.
32. R Core Team R: a language and environment for statistical computing. R foundation for statistical computing; 2020. https://www.R-project.org.
33. Miasko T, Nowotny M. PyJAGS: the Python interface to JAGS. pyjags version 1.3.7; 2020. https://pypi.org/project/pyjags.
34. Grant RL, Carpenter B, Furr DC, Gelman A. Introducing the StataStan interface for fast, complex Bayesian modeling using Stan. *Stata J*. 2017;17(2):330-342. https://doi.org/10.1177/1536867x1701700205.
35. Steyvers M, Vincent BT, Carroll C. MATJAGS: a matlab interface for JAGS. matjags version 1.3.3; 2016. https://github.com/msteyvers/matjags.
36. Derks K, Wetzels R, Swart J. JAGS interface for JASP. audit version 1; 2019. https://github.com/vandenman/JAGS-for-JASP.

37. Cox DR. Regression models and life-tables. *J R Stat Soc B Methodol*. 1972;34(2):187-220. https://doi.org/10.1111/j.2517-6161.1972.tb00899.x.
38. Christensen R, Wesley J, Branscum A, Hanson TE. *Bayesian Ideas and Data Analysis: An Introduction for Scientists and Statisticians*. 1st ed. Boca Raton, FL: Chapman & Hall/CRC Press; 2011.
39. Klein JP, Moeschberger ML, Yan J. KMsurv: data sets from klein and moeschberger Survival analysis. R package version 0.1–5; 2012. https://CRAN.R-project.org/package=KMsurv
40. Kardaun O. Statistical survival analysis of male larynx-cancer patients - a case study. *Statistica Neerlandica*. 1983;37(3):103-125. https://doi.org/10.1111/j.1467-9574.1983.tb00806.x.
41. Sahu SK, Dey DK, Aslanidou H, Sinha D. A Weibull regression model with gamma frailties for multivariate survival data. *Lifetime Data Anal*. 1997;3(2):123-137. https://doi.org/10.1023/a:1009605117713.
42. Banerjee A, Kundu D. Inference based on type-II hybrid censored data from a Weibull distribution. *IEEE Trans Reliab*. 2008;57(2):369-378. https://doi.org/10.1109/TR.2008.916890.
43. Lunn D, Jackson C, Best N, Thomas A, Spiegelhalter D. *The BUGS Book: A Practical Introduction to Bayesian Analysis*. 1st ed. Boca Raton, FL: Chapman & Hall/CRC Press; 2012.
44. Plummer M. JAGS version 4.3.0 user manual. JAGS; 2017. https://sourceforge.net/projects/mcmc-jags/files/Manuals/4.x/jags_user_%manual.pdf.
45. Gelman A, Rubin DB. Inference from iterative simulation using multiple sequences. *Stat Sci*. 1992;7(4):457-472. https://doi.org/10.1214/ss/1177011136.
46. Brooks SP, Gelman A. General methods for monitoring convergence of iterative simulations. *J Comput Graph Stat*. 1998;7(4):434-455. https://doi.org/10.1080/10618600.1998.10474787.
47. Plummer M, Best N, Cowles K, et al. coda: output analysis and diagnostics for MCMC. R package version 0.19–3; 2019. https://CRAN.R-project.org/package=coda.
48. Royston P. Estimating a smooth baseline Hazard function for the cox model. Research report 314, MRC Clinical Trials Unit and University College London; 2011.
49. Lázaro E, Armero C, Alvares D. Bayesian regularization for flexible baseline Hazard functions in cox survival models. *Biom J*. 2021;63(1):7-26. https://doi.org/10.1002/bimj.201900211.
50. Lin X, Cai B, Wang L, Zhang Z. A Bayesian proportional hazards model for general interval-censored data. *Lifetime Data Anal*. 2015;21(3):470-490. https://doi.org/10.1007/s10985-014-9305-9.
51. Mitra R, Müller P, eds. *Nonparametric Bayesian Inference in Biostatistics*. 1st ed. New York, NY: Springer; 2015.
52. Breslow N. Covariance analysis of censored survival data. *Biometrics*. 1974;30(1):89-99. https://doi.org/10.2307/2529620.
53. Murray TA, Hobbs BP, Sargent DJ, Carlin BP. Flexible Bayesian survival modeling with semiparametric time-dependent and shape-restricted covariate effects. *Bayesian Anal*. 2016;11(2):381-402. https://doi.org/10.1214/15-BA954.
54. Lee KH, Dominici F, Schrag D, Haneuse S. Hierarchical models for semicompeting risks data with application to quality of end-of-life care for pancreatic cancer. *J Am Stat Assoc*. 2016;111(515):1075-1095. https://doi.org/10.1080/01621459.2016.1164052.
55. Sharef E, Strawderman RL, Ruppert D, Cowen M, Halasyamani L. Bayesian adaptive B-spline estimation in proportional hazards frailty models. *Electr J Stat*. 2010;4:606-642. https://doi.org/10.1214/10-EJS566.
56. Ntzoufras I. *Bayesian Modeling Using WinBUGS*. 1st ed. Boca raton, FL: John Wiley & Sons; 2009.
57. Berkson J, Gage RP. Survival curve for Cancer patients following treatment. *J Am Stat Assoc*. 1952;47(259):501-515. https://doi.org/10.1080/01621459.1952.10501187.
58. Cai C, Zou Y, Peng Y, Zhang J. smcure: fit semiparametric mixture cure models. R package version 2.0; 2012. https://CRAN.R-project.org/package=smcure.
59. Kersey JH, Weisdorf D, Nesbit ME, et al. Comparison of autologous and allogeneic bone marrow transplantation for treatment of high-risk refractory acute lymphoblastic leukemia. *N Engl J Med*. 1987;317(8):461-467. https://doi.org/10.1056/nejm198708203170801.
60. Putter H, Fiocco M, Geskus RB. Tutorial in biostatistics: competing risks and multi-state models. *Stat Med*. 2007;26(11):2389-2430. https://doi.org/10.1002/sim.2712.
61. Ge M, Chen MH. Bayesian inference of the fully specified subdistribution model for survival data with competing risks. *Lifetime Data Anal*. 2012;18(3):339-363. https://doi.org/10.1007/s10985-012-9221-9.
62. Grambauer N, Neudecker A. compeir: event-specific incidence rates for competing risks data. R package version 1.0; 2011. https://CRAN.R-project.org/package=compeir.
63. Dettenkofer M, Wenzler-Rottele S, Babikir R, et al. Surveillance of nosocomial Sepsis and pneumonia in patients with a bone marrow or peripheral blood stem cell transplant: a multicenter project. *Clin Infect Dis*. 2005;40(7):926-931. https://doi.org/10.1086/428046.
64. Beyersmann J, Dettenkofer M, Bertz H, Schumacher M. A competing risks Analysis of bloodstream infection after stem-cell transplantation using subdistribution hazards and cause-specific hazards. *Stat Med*. 2007;26(30):5360-5369. https://doi.org/10.1002/sim.3006.
65. Borchers HW. pracma: Practical Numerical Math Functions. R package version 2.2.5; 2019. https://CRAN.R-project.org/package=pracma.
66. Andersen PK, Keiding N. Multi-state Models for event history analysis. *Stat Methods Med Res*. 2002;11(2):91-115. https://doi.org/10.1191/0962280202sm276ra.
67. Han B, Yu M, Dignamc JJ, Rathouzb PJ. Bayesian approach for flexible modeling of Semicompeting risks data. *Stat Med*. 2014;33(29):5111-5125. https://doi.org/10.1002/sim.6313.

68. Armero C, Cabras S, Castellanos ME, et al. Bayesian analysis of a disability model for lung Cancer survival. *Stat Methods Med Res*. 2016;25(1):336-351. https://doi.org/10.1177/0962280212452803.
69. Andersen PK, Perme MP. Inference for outcome probabilities in multi-state Models. *Lifetime Data Anal*. 2008;14(4):405-431. https://doi.org/10.1007/s10985-008-9097-x.
70. Meira-Machado L, Roca-Pardinas J. p3state.msm: analyzing survival data from illness-death model. R package version 1.3; 2012. https://CRAN.R-project.org/package=p3state.msm.
71. Crowley J, Hu M. Covariance analysis of heart transplant survival data. *J Am Stat Assoc*. 1977;72(357):27-36. https://doi.org/10.1080/01621459.1977.10479903.
72. Aalen OO. Effects of frailty in survival analysis. *Stat Methods Med Res*. 1994;3(3):227-243. https://doi.org/10.1177/096228029400300303.
73. Clayton DG. A model for association in bivariate life tables and its application in epidemiological studies of familial tendency in chronic disease incidence. *Biometrika*. 1978;65(1):141-151. https://doi.org/10.1093/biomet/65.1.141.
74. Hougaard P. *Analysis of Multivariate Survival Data*. 1st ed. New York, NY: Springer; 2000.
75. Vaupel JW, Manton KG, Stallard E. The impact of heterogeneity in individual frailty on the dynamics of mortality. *Demography*. 1979;16(3):439-454. https://doi.org/10.2307/2061224.
76. Ha ID, Noh M, Kim J, Lee Y. frailtyHL: frailty models via hierarchical likelihood. R package version 2.2; 2018. https://CRAN.R-project.org/package=frailtyHL.
77. McGilchrist CA, Aisbett CW. Regression with frailty in survival Analysis. *Biometrics*. 1991;47(2):461-466. https://doi.org/10.2307/2532138.
78. Rizopoulos D. *Joint Models for Longitudinal and Time-To-Event Data: With Applications in R*. 1st ed. Boca Raton, FL: Chapman & Hall/CRC Press; 2012.
79. DeGruttola V. Tu XM. modelling progression of CD4-lymphocyte count and its relationship to survival time. *Biometrics*. 1994;50(4):1003-1014. https://doi.org/10.2307/2533439.
80. Tsiatis AA, DeGruttola V, Wulfsohn MS. Modeling the relationship of survival to longitudinal data measured with error. applications to survival and CD4 counts in patients with AIDS. *J Am Stat Assoc*. 1995;90(429):27-37. https://doi.org/10.1080/01621459.1995.10476485.
81. Faucett CL, Thomas DC. Simultaneously modelling censored survival data and repeatedly measured covariates: a Gibbs sampling approach. *Stat Med*. 1996;15(15):1663-1685. https://doi.org/10.1002/(sici)1097-0258(19960815)15:15<1663::aid-sim294>3.0.co;2-1.
82. Wulfsohn MS, Tsiatis AA. A joint model for survival and longitudinal data measured with error. *Biometrics*. 1997;53(1):330-339. https://doi.org/10.2307/2533118.
83. Armero C, Forte A, Perpiñán H, Sanahuja MJ, Agustí S. Bayesian joint modeling for assessing the progression of chronic kidney disease in children. *Stat Methods Med Res*. 2018;27(1):298-311. https://doi.org/10.1177/0962280216628560.
84. Wu MC, Carroll RJ. Estimation and comparison of changes in the presence of informative right censoring by modeling the censoring process. *Biometrics*. 1988;44(1):175-188. https://doi.org/10.2307/2531905.
85. Hogan JW, Laird NM. Mixture models for the joint distribution of repeated measures and event times. *Stat Med*. 1997;16(3):239-257. https://doi.org/10.1002/(sici)1097-0258(19970215)16:3<239::aid-sim483>3.0.co;2-x.
86. Hogan JW, Laird NM. Increasing efficiency from censored survival data by using random effects to model longitudinal covariates. *Stat Methods Med Res*. 1998;7(1):28-48. https://doi.org/10.1177/096228029800700104.
87. Proust-Lima C, Séne M, Taylor JMG, Jacqmin-Gadda H. Joint latent class Models for longitudinal and time-to-event data: a review. *Stat Methods Med Res*. 2014;23(1):74-90. https://doi.org/10.1177/0962280212445839.
88. Taylor JMG, Park Y, Ankerst DP, et al. Real-time individual predictions of prostate cancer recurrence using joint models. *Biometrics*. 2013;69(1):206-213. https://doi.org/10.1111/j.1541-0420.2012.01823.x.
89. Rizopoulos D, Taylor JMG, Rosmalen JV, Steyerberg EW, Takkenberg JJM. Personalized screening intervals for biomarkers using joint models for longitudinal and survival data. *Biostatistics*. 2015;17(1):149-164. https://doi.org/10.1093/biostatistics/kxv031.
90. Armero C, Forné C, Rué M, et al. Bayesian joint ordinal and survival modeling for breast cancer risk assessment. *Stat Med*. 2016;35(28):5267-5282. https://doi.org/10.1002/sim.7065.
91. Rué M, Andrinopoulou ER, Alvares D, Armero C, Forte A, Blanch L. Bayesian joint modeling of bivariate longitudinal and competing risks data: an application to study patient-ventilator asynchronies in critical care patients. *Biom J*. 2017;59(6):1184-1203. https://doi.org/10.1002/bimj.201600221.
92. Armero C. Bayesian joint models for longitudinal and survival data. Wiley Statistics Reference Online 2020 Doi: arXiv:2005.12822
93. Andersen PK, Borgan Ø, Gill RD, Keiding N. *Statistical Models Based on Counting Processes*. 1st ed. New York, NY: Springer; 1993.
94. Laird NM, Ware JH. Random-effects models for longitudinal data. *Biometrics*. 1982;38(4):963-974. https://doi.org/10.2307/2529876.
95. Schuurman NK, Grasman RPPP, Hamaker EL. A comparison of inverse-wishart prior specifications for covariance matrices in multilevel autoregressive models. *Multivar Behav Res*. 2016;51(2–3):185-206. https://doi.org/10.1080/00273171.2015.1065398.
96. Smyth G, Hu Y, Dunn P, Phipson B, Chen Y. statmod: statistical modeling. R package version 1.4.34; 2020. https://CRAN.R-project.org/package=statmod.
97. Auguie B, Antonov A. gridExtra: miscellaneous functions for "Grid" graphics. R package version 2.3; 2017. https://CRAN.R-project.org/package=gridExtra.

How to cite this article: Alvares D, Lázaro E, Gómez-Rubio V, Armero C. Bayesian survival analysis with BUGS. *Statistics in Medicine*. 2021;40:2975–3020. https://doi.org/10.1002/sim.8933

APPENDIX A. IMPLEMENTATION OF THE TRANSITION PROBABILITIES OF THE SEMI-MARKOV MULTI-STATE MODEL BASED ON POSTERIOR SAMPLES

Figure 5 was generated from the transition probabilities (16)-(21) based on posterior samples of the illness-death model parameter (22)-(24). In particular, we set a grid of time values to evaluate the transition probabilities and used the median values of `age` and `year` covariates. In this example, we are interested in the transition probabilities for patients with and without prior bypass surgery (ie, `surgery` equal to 1 or 0, respectively):

```
R> grid <- seq(0, 1000, length.out = 51)
R> x0 <- c(median(heart2$age), median(heart2$year), 0)
R> x1 <- c(median(heart2$age), median(heart2$year), 1)
```

For our Weibull baseline hazard specification, the transition probability $p_{11}(s,t|\boldsymbol{\theta})$ (see Equation (16)) can be calculated analytically and is expressed by:

$$p_{11}(s,t|\boldsymbol{\theta}) = \exp\left\{-\lambda_1\eta_1\left[t^{\alpha_1} - s^{\alpha_1}\right] - \lambda_3\eta_3\left[t^{\alpha_3} - s^{\alpha_3}\right]\right\}, \tag{A1}$$

where $\eta_k = \exp\left(\boldsymbol{x}^\top \boldsymbol{\beta}_k\right)$, for $k = 1, 3$. Equation (A1) is implemented as follows:

```
R> p11_{s}_t <- function(tt, ss, l1, a1, b1, l2, a2, b2, x) {
+ H1 <- l1 * exp(sum(b1 * x)) * (tt^{a}1 - ss^{a}1)
+ H2 <- l2 * exp(sum(b2 * x)) * (tt^{a}2 - ss^{a}2)
+ return(exp(-H1 - H2))
+ }
```

Next, we have created a function `mcmc_p11_s_t` that takes posterior samples of each parameter from JAGS and computes $p_{11}(s,t|\boldsymbol{\theta})$ in (A1) for a grid of time values (variable `grid`) using covariates `x` (here `x0` or `x1`). Note that `mcmc_p11_s_t` is based on the `mclapply` function, available from the `parallel` package,[32] to speed computations up.

```
R> library("parallel")
R> options(mc.cores = detectCores())
R> mcmc_{p}11_{s}_t <- function(t.pred, s.pred, l1, a1, b1, l2, a2, b2, x) {
+ # Sub-sample to speed up computations
+ samples.idx <- sample(1:length(l1), 200)
+ res <- sapply(t.pred, function(tt) {
+ aux <- mclapply(samples.idx, function(i) {
+ p11_{s}_t(tt, ss = s.pred, l1 = l1[i], a1 = a1[i], b1 = b1[i, ],
+ l2 = l2[i], a2 = a2[i], b2 = b2[i, ], x = x)
+ })
+ return(mean(unlist(aux)))
+ })
+ return(res)
+ }
```

As previously mentioned, we are interested in the transition probabilities for patients with and without prior bypass surgery. In particular, $p_{11}(s,t|\boldsymbol{\theta})$ for $s = 0$ can be calculated as follows:

```
R> p11_{0}_t_{s}urgery0 <- mcmc_{p}11_{s}_t(t.pred = grid, s.pred = 0, l1 =
lambda1, a1 = alpha1,
+ b1 = cbind(beta11, beta21, beta31), l2 = lambda2, a2 = alpha2,
+ b2 = cbind(beta12, beta22, beta32), x = x0)
R> p11_{0}_t_{s}urgery1 <- mcmc_{p}11_{s}_t(t.pred = grid, s.pred = 0, l1 =
lambda1, a1 = alpha1,
+ b1 = cbind(beta11, beta21, beta31), l2 = lambda2, a2 = alpha2,
+ b2 = cbind(beta12, beta22, beta32), x = x1)
```

Finally, these transition probabilities for `x0` and `x1` are plotted in Figure 5 using the following code:

```
R> library("ggplot2")
R> surg.colour <- rep(c("no", "yes"), each = length(grid))
R> df11 <- data.frame(time = rep(grid, 2), surgery = factor(surg.colour),
+ transition = c(p11_{0}_t_{s}urgery0, p11_{0}_t_{s}urgery1))
R> p11 <- ggplot(data = df11, aes(x = time, y = transition, group = surgery, colour
= surgery)) +
+ geom_{l}ine() + annotate("text", x = 880, y = 1, label=expression(1
+ ylab("probability") + theme(legend.position = "top") + ylim(0,1) + theme_{b}w()
```

The transition probability $p_{22}(s, t|\boldsymbol{\theta})$ (see Equation (17)) using the Weibull baseline hazard specification is written as:

$$p_{22}(s, t|\boldsymbol{\theta}) = \frac{\lambda_1\alpha_1\eta_1}{1 - \exp\left\{-\lambda_1\eta_1 s^{\alpha_1}\right\}} \int_0^s u^{\alpha_1 - 1} \exp\left\{-\lambda_1\eta_1 u^{\alpha_1} - \lambda_3\eta_3\left[(t-u)^{\alpha_3} - (s-u)^{\alpha_3}\right]\right\} \,du, \tag{A2}$$

where $\eta_k = \exp\left(\boldsymbol{x}^\top\boldsymbol{\beta}_k\right)$, for $k = 1, 3$. The integral in (A2) has no closed form, so some approximate method of integration is required. To do this, we first have created a function `int_p22` to calculate this integral:

```
R> int_{p}22 <- function(u, tt, ss, l1, a1, b1, l3, a3, b3, x) {
+ F1 <- 1 - exp(-l1 * exp(sum(b1 * x)) * ssˆ{a}1)
+ u1 <- ifelse(u > 0, uˆ(a1 - 1),0)
+ h1 <- l1 * a1 * exp(sum(b1 * x)) * u1
+ H1 <- l1 * exp(sum(b1 * x)) * uˆ{a}1
+ H3 <- l3 * exp(sum(b3 * x)) * ((tt - u)ˆ{a}3 - (ss - u)ˆ{a}3)
+ return(h1 * exp(-H1 - H3) / F1)
+ }
```

Next, we have created a function `p22_s_t` that computes $p_{22}(s, t|\boldsymbol{\theta})$ in (A2) by integrating out the `int_p22` function using the `integral` function available from the `pracma` package.[65]

```
R> library("pracma")
R> p22_{s}_t <- function(tt, ss, l1, a1, b1, l3, a3, b3, x) {
+ return(integral(int_{p}22, xmin = 0, xmax = ss, method = "Kronrod", tt = tt,
+ ss = ss, l1 = l1, a1 = a1, b1 = b1, l3 = l3, a3 = a3, b3 = b3, x = x))
+ }
```

Analogous to the previous case, we have created a function `mcmc_p22_s_t` that takes posterior samples of each parameter from JAGS and computes $p_{22}(s, t|\boldsymbol{\theta})$ in (A2) for a grid of time values (variable `grid`) using covariates `x` (here `x0` or `x1`).

```
R> mcmc_{p}22_{s}_t <- function(t.pred, s.pred, l1, a1, b1, l3, a3, b3, x) {
+ # Sub-sample to speed up computations
+ samples.idx <- sample(1:length(l1), 200)
+ res <- sapply(t.pred, function(tt) {
+ aux <- mclapply(samples.idx, function(i) {
+ p22_{s}_t(tt, ss = s.pred, l1 = l1[i], a1 = a1[i], b1 = b1[i, ],
+ l3 = l3[i], a3 = a3[i], b3 = b3[i, ], x = x)
+ })
+ return(mean(unlist(aux)))
+ })
+ return(res)
+ }
```

Note that the transition time from state 1 to state 2 prevents the case $s = 0$ in $p_{22}(s, t|\boldsymbol{\theta})$. To realistically get around this problem, we have set s as the median time of patients who transitioned from state 1 to 2 ($s = 26$) and add this amount to the grid of time values. So, we have calculated the transition probabilities (A2) for `x0` and `x1`:

```
R> m <- median(heart2$times1[heart2$delta == 1])
R> grid2 <- grid + m
R> p22_{m}_t_{s}urgery0 <- mcmc_{p}22_{s}_t(t.pred = grid2, s.pred = m, l1 =
lambda1, a1 = alpha1,
+ b1 = cbind(beta11, beta21, beta31), l3 = lambda3, a3 = alpha3,
+ b3 = cbind(beta13, beta23, beta33), x = x0)
R> p22_{m}_t_{s}urgery1 <- mcmc_{p}22_{s}_t(t.pred = grid2, s.pred = m, l1 =
lambda1, a1 = alpha1,
+ b1 = cbind(beta11, beta21, beta31), l3 = lambda3, a3 = alpha3,
+ b3 = cbind(beta13, beta23, beta33), x = x1)
```

Finally, these transition probabilities are plotted in Figure 5 using the following code:

```
R> df22 <- data.frame(time = rep(grid, 2), surgery = factor(surg.colour),
+ transition = c(p22_{m}_t_{s}urgery0, p22_{m}_t_{s}urgery1))
R> p22 <- ggplot(data = df22, aes(x = time, y = transition, group = surgery, colour
= surgery)) +
+ geom_{l}ine() + annotate("text", x = 880, y = 1, label=expression(2
+ ylab("probability") + theme(legend.position = "top") + ylim(0,1) + theme_{b}w()
```

The transition probability $p_{12}(s, t|\boldsymbol{\theta})$ (see Equation (18)) using the Weibull baseline hazard specification is written as:

$$p_{12}(s, t|\boldsymbol{\theta}) = \lambda_1\alpha_1\eta_1 \int_s^t u^{\alpha_1-1} \exp\left\{-\lambda_1\eta_1\left[u^{\alpha_1} - s^{\alpha_1}\right] - \lambda_2\eta_2\left[u^{\alpha_2} - s^{\alpha_2}\right] - \lambda_3\eta_3(t-u)^{\alpha_3}\right\}\, du, \tag{A3}$$

where $\eta_k = \exp\left(\boldsymbol{x}^\top \boldsymbol{\beta}_k\right)$, for $k = 1, 2, 3$. Analogous to the previous case, the integral in (A3) has no closed form, so some approximate method of integration is required. To do this, we first have created a function `int_p12` to calculate this integral:

```
R> int_{p}12 <- function(u, tt, ss, l1, a1, b1, l2, a2, b2, l3, a3, b3, x) {
+ u1 <- ifelse(u > 0, û(a1 - 1), 0)
+ h1 <- l1 * a1 * exp(sum(b1 * x)) * u1
+ H1 <- l1 * exp(sum(b1 * x)) * (û{a}1 - sŝ{a}1)
+ H2 <- l2 * exp(sum(b2 * x)) * (û{a}2 - sŝ{a}2)
+ H3 <- l3 * exp(sum(b3 * x)) * (tt - u)ˆ{a}3
+ return(h1 * exp(-H1 - H2 - H3))
+ }
```

Next, we have created a function `p12_s_t` that computes $p_{12}(s, t|\boldsymbol{\theta})$ in (A3) by integrating out the `int_p12` function.

```
R> p12_{s}_t <- function(tt, ss, l1, a1, b1, l2, a2, b2, l3, a3, b3, x) {
+ return(integral(int_{p}12, xmin = ss, xmax = tt, method = "Kronrod", tt = tt,
+ ss = ss, l1 = l1, a1 = a1, b1 = b1, l2 = l2, a2 = a2, b2 = b2,
+ l3 = l3, a3 = a3, b3 = b3, x = x))
+ }
```

As a last step, we have created a function `mcmc_p12_s_t` that takes posterior samples of each parameter from JAGS and computes $p_{12}(s, t|\boldsymbol{\theta})$ in (A3) for a grid of time values (variable `grid`) using covariates `x` (here `x0` or `x1`).

```
R> mcmc_{p}12_{s}_t <- function(t.pred, s.pred, l1, a1, b1, l2, a2, b2, l3, a3, b3,
x) {
+ # Sub-sample to speed up computations
+ samples.idx <- sample(1:length(l1), 200)
+ res <- sapply(t.pred, function(tt) {
+ aux <- mclapply(samples.idx, function(i) {
+ p12_{s}_t(tt, ss = s.pred, l1 = l1[i], a1 = a1[i], b1 = b1[i, ], l2 = l2[i],
+ a2 = a2[i], b2 = b2[i, ], l3 = l3[i], a3 = a3[i], b3 = b3[i, ], x = x)
+ })
+ return(mean(unlist(aux)))
+ })
+ return(res)
+ }
```

The transition probabilities $p_{12}(s, t|\boldsymbol{\theta})$ of `x0` and `x1` for $s = 0$ can be calculated as follows:

```
R> p12_{0}_t_{s}urgery0 <- mcmc_{p}12_{s}_t(t.pred = grid, s.pred = 0, l1 =
lambda1, a1 = alpha1,
+ b1 = cbind(beta11, beta21, beta31), l2 = lambda2, a2 = alpha2,
+ b2 = cbind(beta12, beta22, beta32), l3 = lambda3, a3 = alpha3,
+ b3 = cbind(beta13, beta23, beta33), x = x0)
R> p12_{0}_t_{s}urgery1 <- mcmc_{p}12_{s}_t(t.pred = grid, s.pred = 0, l1 =
lambda1, a1 = alpha1,
+ b1 = cbind(beta11, beta21, beta31), l2 = lambda2, a2 = alpha2,
+ b2 = cbind(beta12, beta22, beta32), l3 = lambda3, a3 = alpha3,
+ b3 = cbind(beta13, beta23, beta33), x = x1)
```

Finally, these transition probabilities are plotted in Figure 5 using the following code:

```
R> df12 <- data.frame(time = rep(grid, 2), surgery = factor(surg.colour),
+ transition = c(p12_{0}_t_{s}urgery0, p12_{0}_t_{s}urgery1))
R> p12 <- ggplot(data = df12, aes(x = time, y = transition, group = surgery, colour
= surgery)) +
+ geom_{l}ine() + annotate("text", x = 880, y = 1, label=expression(1
+ ylab("probability") + theme(legend.position = "top") + ylim(0,1) + theme_{b}w()
```

As shown in (19) and (20), $p_{13}(s, t|\boldsymbol{\theta})$ and $p_{23}(s, t|\boldsymbol{\theta})$ are derived from the previously calculated transition probabilities. So, we can get them for `x0` and `x1` as follows:

```
R> p13_0_t_surgery0 <- 1 - p11_0_t_surgery0 - p12_0_t_surgery0
R> p13_0_t_surgery1 <- 1 - p11_0_t_surgery1 - p12_0_t_surgery1
R> p23_m_t_surgery0 <- 1 - p22_m_t_surgery0
R> p23_m_t_surgery1 <- 1 - p22_m_t_surgery1
```

These transition probabilities are plotted in Figure 5 using the following code:

```
R> df13 <- data.frame(time = rep(grid, 2), surgery = factor(surg.colour),
+ transition = c(p13_0_t_surgery0, p13_0_t_surgery1))
R> df23 <- data.frame(time = rep(grid, 2), surgery = factor(surg.colour),
+ transition = c(p23_m_t_surgery0, p23_m_t_surgery1))
```

```
R> p13 <- ggplot(data = df13, aes(x = time, y = transition, group = surgery, colour
= surgery)) +
+ geom_line() + annotate("text", x = 880, y = 1, label=expression(1
+ ylab("probability") + theme(legend.position = "top") + ylim(0,1) + theme_bw()
R> p23 <- ggplot(data = df23, aes(x = time, y = transition, group = surgery, colour
= surgery)) +
+ geom_line() + annotate("text", x = 880, y = 1, label=expression(2
+ ylab("probability") + theme(legend.position = "top") + ylim(0,1) + theme_bw()
```

To organize these graphs in two rows and three columns, as in Figure 5, we have used the `grid.arrange` function available from the `gridExtra` package.[97]

```
R> library("gridExtra")
R> grid.arrange(p11, p12, p13, p22, p23, nrow = 2, ncol = 3,
+ layout_matrix = rbind(c(1, 2, 3), c(NA, 5, 6)))
```