entropy

MDPI

Article

Bayesian Computational Methods for Sampling from the Posterior Distribution of a Bivariate Survival Model, Based on AMH Copula in the Presence of Right-Censored Data

Erlandson Ferreira Saraiva [1], Adriano Kamimura Suzuki [2,*] and Luis Aparecido Milan [3]

[1] Instituto de Matemática, Universidade Federal de Mato Grosso do Sul, Campo Grande 79070-900, Brazil; erlandson.saraiva@ufms.br

[2] Departamento de Matemática Aplicada e Estatística, Universidade de São Paulo, São Carlos 13566-590, Brazil

[3] Departamento de Estatística, Universidade de São Carlos, São Carlos 13565-905, Brazil; dlam@ufscar.br

* Correspondence: suzuki@icmc.usp.br; Tel.: +55-16-3373-8164

Received: 27 June 2018; Accepted: 23 August 2018; Published: 27 August 2018

Abstract: In this paper, we study the performance of Bayesian computational methods to estimate the parameters of a bivariate survival model based on the Ali–Mikhail–Haq copula with marginal distributions given by Weibull distributions. The estimation procedure was based on Monte Carlo Markov Chain (MCMC) algorithms. We present three version of the Metropolis–Hastings algorithm: Independent Metropolis–Hastings (IMH), Random Walk Metropolis (RWM) and Metropolis–Hastings with a natural-candidate generating density (MH). Since the creation of a good candidate generating density in IMH and RWM may be difficult, we also describe how to update a parameter of interest using the slice sampling (SS) method. A simulation study was carried out to compare the performances of the IMH, RWM and SS. A comparison was made using the sample root mean square error as an indicator of performance. Results obtained from the simulations show that the SS algorithm is an effective alternative to the IMH and RWM methods when simulating values from the posterior distribution, especially for small sample sizes. We also applied these methods to a real data set.

Keywords: Bayesian inference; Ali–Mikhail–Haq copula; MCMC; Metropolis-Hastings; slice sampling

1. Introduction

In survival studies, it is common to observe two or more lifetimes for the same client, patient or equipment. For instance, in a bivariate scenario, the lifetimes of a pair of organs can be observed, such as a pair of kidneys, liver, or eyes in patients; or the lifetimes of engines in a twin-engine airplane.

These variables are usually correlated and we are interested in the bivariate model that considers the dependence between them. The copula model is useful for modeling this kind of bivariate data. It has been used in several articles, including the following: [1] describes a comparison between bivariate frailty models, and models based on bivariate exponential and Weibull distributions; [2] proposes a copula model to study the association between survival time of individuals infected with HIV and persistence time of infection; [3] models the association of bivariate failure times by copula functions, and investigates two-stage parametric and semi-parametric procedures; and [4] considers a Gaussian copula model and estimates the copula association parameter using a two-stage estimation procedure.

According to [5,6], a copula is a joint distribution function of random variables for which the marginal probability distribution of each variable is uniformly distributed on the interval $[0, 1]$.

There are many parametric copula families in the literature, each one representing a different dependence structure between the random variables. One advantage of a copula model is its simplicity when applied to model bivariate data. This is explored by many authors in survival analysis. Among them are: Romeo et al. [7] and da Cruz et al. [8], who considered the Archimedean copula family; Louzada et al. [9] and Suzuki et al. [10], who considered the Farlie–Gumbel–Morgenstern (FGM) copula; and Romeo et al. [11], who considered the two-parameter Archimedean family of power variance function (PVF) copulas.

In this paper, we apply the Ali–Mikhail–Haq (AMH) copula to model bivariate survival data with random right-censored observations. From a practical point of view, the main reason for using the AMH copula is that it is an Archimedean copula that allows both positive and negative values for the dependence parameter, and whose mathematical formula is simpler than other Archimedean copulas. Another advantage is that assuming the AMH copula, the Kendall rank-order correlation τ between the bivariate lifetimes is a monotonic function of the dependence parameter ϕ. According to [12], the Kendall's τ can range from (approximately) -0.18 to 0.33, with $\tau = 0$ when $\phi = 0$; and the Spearman's ρ associated to ϕ can range (approximately) from -0.2711 to 0.4784, indicating that the AMH copula is adequate for modeling bivariate data with a weak correlation.

In order to proceed with the copula model it is necessary to specify the marginal distributions. At this point, several probability distributions could be considered. Generally, the choice for marginal distributions depends on the application. We restrict our analysis to the case where the marginal distributions are Weibull distributions. This is because it is a very flexible distribution for the modeling of various types of lifetime data. In addition, the parametrization of the Weibull distribution—as well as the mathematical expression of the AMH copula—is very attractive from the mathematical point of view, allowing the development of a Bayesian approach to estimate the parameters of interest in a clear and concise way.

As the conditional posterior distributions for parameters of interest does not follow any familiar distribution, the estimation procedure was carried out using versions of the Metropolis–Hastings algorithm, referred to here as Independent Metropolis–Hastings (IMH), Random Walk Metropolis (RWM) and Metropolis–Hastings (MH). MH refers to the Metropolis–Hastings algorithm with a natural candidate generating density whose parameters depend on the hyperparameter values and the observed data. Since the creation of a good candidate generating density in IMH and RWM can be difficult, we also used the slice sampling algorithm [13].

Combining IMH, RWM, MH and SS in different ways, we developed three MCMC algorithms to estimate the model parameters. A simulation study was carried out with the objective of investigating the behavior of each algorithm. The data sets were generated by considering different sample sizes and percentages of right-censored observations. Based on the root mean square error (RMSE), we identified the algorithms with the best performances when estimating the model parameters. We also compared the performances of the three algorithms using the effective sample size and the integrated autocorrelation time [14]. Results obtained from these simulations show that the algorithm that applied the SS algorithm is an effective alternative for standard MCMC methods (IMH and RWM) when simulating values from the posterior distribution of the model parameters, especially when the sample size is small.

We applied the three proposed algorithms to a real data set. This data set is related to diabetic retinopathy, described in The Diabetic Retinopathy Study Research Group [15], and is available in the 'survival' package [16] of the R software [17]. For this case, we compared the performance of the algorithms. Comparison was based on the RMSE relative to the empirical distribution function obtained from Kaplan–Meier estimates.

The remainder of the paper is organized as follows. In Section 2, we introduce the bivariate survival model based on the AMH copula with Weibull marginal distributions. The Bayesian approach and the three MCMC algorithms are described in Section 3. In Section 4, the simulation study is reported. In Section 5 we apply the three algorithms to the real data set. Section 6 summarizes our findings.

2. Bivariate Survival Model and Observed Data

Let (T_1, T_2) be the vector of bivariate lifetimes of an item (or an individual) with marginal density functions $(f(t_1|\theta_1), f(t_2|\theta_2))$ and the survival functions be $(S(t_1|\theta_1), S(t_2|\theta_2))$, where θ_1 and θ_2 are unknown parameters (scalars or vectors).

Consider that (T_1, T_2) comes from the copula $\tilde{C}_\phi$, where ϕ is a parameter showing dependence between T_1 and T_2. Then the joint survival function for (T_1, T_2) is given by

$$\mathbf{S}(t_1, t_2|\boldsymbol{\theta}, \phi) = \tilde{C}_\phi\left(S_1(t_1|\theta_1), S_2(t_2|\theta_2)\right),$$

where $\boldsymbol{\theta} = (\theta_1, \theta_2)$ and ϕ is a dependence parameter.

We also assume that copula $\tilde{C}_\phi$ is given by the Ali–Mikhail–Haq copula [18]. Thus, we have

$$\mathbf{S}(t_1, t_2|\boldsymbol{\theta}, \phi) = \tilde{C}_\phi\left(S_1(t_1|\theta_1), S_2(t_2|\theta_2)\right) = \frac{S_1(t_1|\theta_1)S_2(t_2|\theta_2)}{1-\phi\left(1-S_1(t_1|\theta_1)\right)\left(1-S_2(t_2|\theta_2)\right)}, \tag{1}$$

for $\phi \in [-1, 1)$. Note that under this assumption the survival functions and the dependence structure can be visualized separately with the dependence structure represented by the copula.

Let $(T_{11}, T_{12}), \ldots, (T_{n1}, T_{n2})$ and $(C_{11}, C_{12}), \ldots, (C_{n1}, C_{n2})$ be a sample of size n of bivariate lifetimes and censured bivariate lifetimes, respectively. Suppose (T_{i1}, T_{i2}) and (C_{i1}, C_{i2}) are independent, for $i = 1, \ldots, n$. Consider $t_{ij} = min(T_{ij}, C_{ij})$—the i-th observed value and δ_{ij}—a censorship indicator given by

$$\delta_{ij} = \begin{cases} 1, & \text{if the lifetime is uncensored, i.e., } T_{ij} = t_{ij}; \\ 0, & \text{if the lifetime is censored, i.e., } T_{ij} > t_{ij}, \end{cases}$$

for $j = 1, 2$ and $i = 1, \ldots, n$. We denote the observed values using $\mathbf{t} = (\mathbf{t}_1, \mathbf{t}_2)$ and $\boldsymbol{\delta} = (\boldsymbol{\delta}_1, \boldsymbol{\delta}_2)$, where $\mathbf{t}_1 = (t_{11}, \ldots, t_{n1})$, $\mathbf{t}_2 = (t_{12}, \ldots, t_{n2})$, $\boldsymbol{\delta}_1 = (\delta_{11}, \ldots, \delta_{n1})$ and $\boldsymbol{\delta}_2 = (\delta_{12}, \ldots, \delta_{n2})$.

The likelihood function for $(\boldsymbol{\theta}, \phi)$, given $(\mathbf{t}, \boldsymbol{\delta})$, is (see Lawless, [19])

$$L(\boldsymbol{\theta}, \phi|\mathbf{t}, \boldsymbol{\delta}) = \prod_{i=1}^{n} f(t_{i1}, t_{i2}|\boldsymbol{\theta}, \phi)^{\delta_{i1}\delta_{i2}} \mathbf{S}'^{\delta_{i1}(1-\delta_{i2})}_{(t_1)} \mathbf{S}'^{(1-\delta_{i1})\delta_{i2}}_{(t_2)} \mathbf{S}(t_{i1}, t_{i2}|\boldsymbol{\theta}, \phi)^{(1-\delta_{i1})(1-\delta_{i2})}$$

where $f(t_{i1}, t_{i2}|\boldsymbol{\theta}, \phi) = \frac{d^2\mathbf{S}(t_{i1}, t_{i2}|\boldsymbol{\theta}, \phi)}{dt_{i1}dt_{i2}}$ is the joint probability density function for (t_{i1}, t_{i2}), $\mathbf{S}'_{(t_1)} = \left(-\frac{d\mathbf{S}(t_{i1}, t_{i2}|\boldsymbol{\theta}, \phi)}{dt_{i1}}\right)$, $\mathbf{S}'_{(t_2)} = \left(-\frac{d\mathbf{S}(t_{i1}, t_{i2}|\boldsymbol{\theta}, \phi)}{dt_{i2}}\right)$, and $\mathbf{S}(t_{i1}, t_{i2}|\boldsymbol{\theta}, \phi)$ is the copula given by (1), for $i = 1, \ldots, n$.

From Equation (1), we have

$$\begin{aligned}
\frac{d^2\mathbf{S}(t_{i1}, t_{i2}|\boldsymbol{\theta}, \phi)}{dt_{i1}dt_{i2}} &= \frac{f_1(t_{i1}|\theta_1)f_2(t_{i2}|\theta_2)\left[(1+\phi)(1+\phi F_1(t_{i1}|\theta_1)F_2(t_{i2}|\theta_2)) - 2\phi(F_1(t_{i1}|\theta_1) + F_2(t_{i2}|\theta_2))\right]}{\left[1-\phi F_1(t_{i1}|\theta_1)F_2(t_{i2}|\theta_2)\right]^3}, \\
-\frac{d\mathbf{S}(t_{i1}, t_{i2}|\boldsymbol{\theta}, \phi)}{dt_{i1}} &= \frac{f_1(t_{i1}|\theta_1)S_2(t_{i2}|\theta_2)\left[1-\phi F_2(t_{i2}|\theta_2)\right]}{\left[1-\phi F_1(t_{i1}|\theta_1)F_2(t_{i2}|\theta_2)\right]^2}, \\
-\frac{d\mathbf{S}(t_{i1}, t_{i2}|\boldsymbol{\theta}, \phi)}{dt_{i2}} &= \frac{f_2(t_{i2}|\theta_2)S_1(t_{i1}|\theta_1)\left[1-\phi F_1(t_{i1}|\theta_1)\right]}{\left[1-\phi F_1(t_{i1}|\theta_1)F_2(t_{i2}|\theta_2)\right]^2},
\end{aligned}$$

where $F_j(t_{ij}|\theta_j) = 1 - S_j(t_{ij}|\theta_j)$ is the cumulative distribution function for $j = 1, 2$ and $i = 1, \ldots, n$.

Weibull Marginal Distribution

Assume that the marginal distributions for T_1 and T_2 are given by Weibull distributions [20], i.e.,

$$T_{i1}|\alpha_1,\beta_1 \sim Weibull(\alpha_1,\beta_1) \quad \text{and} \quad T_{i2}|\alpha_2,\beta_2 \sim Weibull(\alpha_2,\beta_2), \tag{2}$$

with shape parameter α_j and scale parameter $\beta_j^{-\alpha_j}$ [21], each one having a probability density function

$$f(t_{ij}|\alpha_j,\beta_j) = \beta_j\alpha_j t_{ij}^{\alpha_j-1} exp\{-\beta_j t_i^{\alpha_j}\}$$

for $j = 1,2$ and $i = 1,\ldots,n$.

The survival function $S_j(t_{ij}|\theta_j)$ and hazard function $h_j(t_{ij}|\theta_j)$ are

$$S_j(t_{ij}|\theta_j) = exp\left\{-\beta_j t_{ij}^{\alpha_j}\right\} \quad \text{and} \quad h_j(t_{ij}|\theta_j) = \beta_j\alpha_j t_{ij}^{\alpha_j-1}$$

respectively, where $\theta_j = (\alpha_j,\beta_j)$ for $j = 1,2$ and $i = 1,\ldots,n$.

Thus, the joint survival function in (1) is

$$\mathbf{S}(t_{i1},t_{i2}|\boldsymbol{\theta},\phi) = \frac{exp\left\{-\beta_1 t_{i1}^{\alpha_1}\right\} exp\left\{-\beta_2 t_{i2}^{\alpha_2}\right\}}{1-\phi\left(1-exp\left\{-\beta_1 t_{i1}^{\alpha_1}\right\}\right)\left(1-exp\left\{-\beta_2 t_{i2}^{\alpha_2}\right\}\right)}$$

where $\boldsymbol{\theta} = (\theta_1,\theta_2)$. The likelihood function for $(\boldsymbol{\theta},\phi)$ is

$$L(\boldsymbol{\theta},\phi|\mathbf{t},\delta) \propto \left[\prod_{j=1}^{2}\beta_j^{r_j}\alpha_j^{r_j}\exp\left\{\alpha_j\sum_{i=1}^{n}\delta_{ij}log(t_{ij}) - \beta_j\sum_{i=1}^{n}t_{ij}^{\alpha_j}\right\}\right]\prod_{i=1}^{n}\Psi_i(\boldsymbol{\theta},\phi|\mathbf{t},\delta), \tag{3}$$

where $r_j = \sum_{i=1}^{n}\delta_{ij}$ is the number of uncensored observations for $j = 1,2$, $\Psi(\boldsymbol{\theta},\phi|\mathbf{t},\delta) = \prod_{k=1}^{4}\Psi_{ik}(\boldsymbol{\theta},\phi|\mathbf{t},\delta)$, and

$$\begin{aligned}
\Psi_{i1}(\boldsymbol{\theta},\phi|\mathbf{t},\delta) &= [(1+\phi)(1+\phi F_1(t_{i1}|\theta_1)F_2(t_{i2}|\theta_2)) - 2\phi(F_1(t_{i1}|\theta_1)+F_2(t_{i2}|\theta_2))]^{\delta_{i1}\delta_{i2}}, \\
\Psi_{i2}(\boldsymbol{\theta},\phi|\mathbf{t},\delta) &= [1-\phi F_2(t_{i2}|\theta_2)]^{\delta_{i1}(1-\delta_{i2})}, \\
\Psi_{i3}(\boldsymbol{\theta},\phi|\mathbf{t},\delta) &= [1-\phi F_1(t_{i1}|\theta_1)]^{\delta_{i2}(1-\delta_{i1})}, \\
\Psi_{i4}(\boldsymbol{\theta},\phi|\mathbf{t},\delta) &= [1-\phi F_1(t_{i1}|\theta_1)F_2(t_{i2}|\theta_2)]^{-(\delta_{i1}+\delta_{i2}+1)},
\end{aligned}$$

for $i = 1,\ldots,n$.

3. Bayesian Approach

In order to develop the Bayesian approach, we need to specify the prior distributions for α_j, β_j and ϕ, for $j = 1,2$. We assume that priors are independent, i.e., $\pi(\boldsymbol{\theta},\phi) = \pi(\boldsymbol{\theta})\pi(\phi) = \left[\prod_{j=1}^{2}\pi(\alpha_j)\pi(\beta_j)\right]\pi(\phi)$. Therefore, we consider the following prior distributions

$$\alpha_j|a_{j1},a_{j2} \sim \Gamma(a_{j1},a_{j2}) \quad \text{and} \quad \beta_j|b_{j1},b_{j2} \sim \Gamma(b_{j1},b_{j2}),$$

where $\Gamma(\cdot)$ is the Gamma distribution and a_{j1}, a_{j2}, b_{j1} and b_{j2} are known hyperparameters, all of them with support on $(0,+\infty)$, for $j = 1,2$. The parametrization of the Gamma distribution is such that the mean is a_{j1}/a_{j2} and the variance is a_{j1}/a_{j2}^2, for $j = 1,2$. The choice of values for the hyperparameters depends on the application. In the remainder of the article, we set up the hyperparameters values that give prior distributions with large variances. In particular, we set $a_{j1} = b_{j1} = 0.01$, for $j = 1,2$. For ϕ we chose the uniform prior distribution on the interval $(-1,1)$, $\phi \sim \mathcal{U}(-1,1)$.

Using Bayes theorem, the joint posterior distribution for $(\boldsymbol{\theta}, \phi)$ is

$$\pi(\boldsymbol{\theta}, \phi | \mathbf{t}, \delta) \propto L(\boldsymbol{\theta}, \phi | \mathbf{t}, \delta) \pi(\boldsymbol{\theta}) \pi(\phi),$$

where $L(\boldsymbol{\theta}, \phi | \mathbf{t}, \delta)$ is given in Equation (3).

The conditional posterior distributions are

$$\begin{aligned} \pi(\alpha_j | \mathbf{t}, \delta, \boldsymbol{\theta}_{-\alpha_j}, \phi) &\propto \alpha_j^{a_{j1}+r_j-1} \exp\left\{ \alpha_j \left(\sum_{i=1}^{n} \delta_{ij} log(t_{ij}) - a_{j2} \right) - \beta_j \sum_{i=1}^{n} t_{ij}^{\alpha_j} \right\} \prod_{i=1}^{n} \Psi_i(\boldsymbol{\theta}, \phi | \mathbf{t}, \delta), \quad (4) \\ \pi(\beta_j | \mathbf{t}, \delta, \boldsymbol{\theta}_{-\beta_j}, \phi) &\propto \beta_j^{b_{j1}+r_j-1} \exp\left\{ -\beta_j \left[b_{j2} + \sum_{i=1}^{n} t_{ij}^{\alpha_j} \right] \right\} \prod_{i=1}^{n} \Psi_i(\boldsymbol{\theta}, \phi | \mathbf{t}, \delta) \text{ and} \quad (5) \\ \pi(\phi | \mathbf{t}, \delta, \boldsymbol{\theta}) &\propto L(\boldsymbol{\theta}, \phi | \mathbf{t}, \delta) \mathbb{I}_{\phi}(-1, 1), \quad (6) \end{aligned}$$

where $\boldsymbol{\theta}_{-\nu_j}$, for $\nu_j \in \{\alpha_j, \beta_j\}$, is the vector of parameters $\boldsymbol{\theta}$ without the parameter ν_j, $j = 1, 2$.

The conditional posterior distributions in Equations (4)–(6) are not familiar distributions. Thus, in order to simulate from conditional posterior distributions, we used the Metropolis–Hastings algorithm. At each iteration, the Metropolis–Hastings algorithm considers a value generated from a proposal distribution. This value is accepted according to a properly specified acceptance probability. This procedure guarantees the convergence of the Markov chain for the target density. More details on the Metropolis–Hastings algorithm can be found in [22–25] and their references.

3.1. MCMC for α_j

Without loss of generality, we describe here how to update parameter α_1 conditional on all other parameters, $\boldsymbol{\theta}_{-\alpha_1} = (\beta_1, \alpha_2, \beta_2)$ and ϕ. The update procedure for α_2 is similar.

Let $(\alpha_1, \boldsymbol{\theta}_{-\alpha_1}, \phi)$ be the current state of the Markov chain. Consider α_1^* a value generated from a candidate generating density $q[\alpha_1^* | \alpha_1]$. The value α_1^* is accepted with probability $\psi(\alpha_1^* | \alpha_1) = min(1, A_{\alpha_1})$, where

$$A_{\alpha_1} = \frac{L(\alpha_1^*, \boldsymbol{\theta}_{-\alpha_1}, \phi | \mathbf{t}, \delta) \pi(\alpha_1^*)}{L(\alpha_1, \boldsymbol{\theta}_{-\alpha_1}, \phi | \mathbf{t}, \delta) \pi(\alpha_1)} \frac{q[\alpha_1 | \alpha_1^*]}{q[\alpha_1^* | \alpha_1]}, \quad (7)$$

and $L(\cdot | \mathbf{y})$ is the likelihood function, given in Equation (3).

The Metropolis–Hastings algorithm is implemented as follows.

- **Metropolis–Hastings Algorithm**: Let the current state of the Markov chain be $\left(\alpha_1^{(l-1)}, \boldsymbol{\theta}_{-\alpha_1}^{(l-1)}, \phi^{(l-1)}\right)$, where l is the l-th iteration of the algorithm, $\alpha_1^{(l-1)}$, $\boldsymbol{\theta}_{-\alpha_1}^{(l-1)} = \left(\beta_1^{(l-1)}, \alpha_2^{(l-1)}, \beta_2^{(l-1)}\right)$ and $\phi^{(l-1)}$ are the values of α_1, $\boldsymbol{\theta}_{-\alpha_1}$ and ϕ in $(l-1)$-th iteration, respectively, for $l = 1, \ldots, L$, in which, $\alpha^{(0)}$, $\boldsymbol{\theta}_{-\alpha_1}^{(0)}$ and $\phi^{(0)}$ are the initial values. At the l-th iteration of the algorithm, we updated α_1 as follows:

  (1) Generate $\alpha_1^* \sim q[\alpha_1^* | \alpha_1]$;
  (2) Calculate $\psi(\alpha_1^* | \alpha_1) = min(1, A_{\alpha_1})$, where A_{α_1} is given by (7);
  (3) Generate $U \sim \mathcal{U}(0, 1)$. If $u \leq \psi(\alpha_1^* | \alpha_1)$ accept α_1^* and do $\alpha_1^{(l)} = \alpha_1^*$. Otherwise, reject α_1^* and set $\alpha_1^{(l)} = \alpha_1^{(l-1)}$.

3.1.1. Two Common Choices for $q[\cdot]$

To implement the Metropolis–Hastings algorithm, the candidate-generating density $q[\alpha_1^* | \alpha_1]$ needs to be specified. Generally, one may explore the form of the conditional posterior distribution to set the candidate-generating density. For example, if we can write $\pi(\alpha_1 | \mathbf{y}, \boldsymbol{\theta}_{-\alpha_1}, \phi)$ as $\pi(\alpha_1 | \mathbf{y}, \boldsymbol{\theta}_{-\alpha_1}, \phi) \propto \eta(\alpha_1) h(\alpha_1)$, where $h(\alpha_1)$ is a density that can be easily generated and $\eta(\alpha_1)$ is uniformly bounded, then we may set up $q(\alpha_1^* | \alpha_1) = h(\alpha_1^*)$. However, this is not the case for $\pi(\alpha_1 | \mathbf{y}, \boldsymbol{\theta}_{-\alpha_1})$.

Another option is to generate α_1^* from a candidate generating density that does not depend on the current α_1 value. That is, we may set up $q[\alpha_1^*|\alpha_1] = q[\alpha_1^*]$. Thus, we have a special case of the original MH algorithm, called Independent Metropolis–Hastings (IMH), where A_{α_1} is given in (7) and simplifies to

$$A_{\alpha_1} = \frac{L(\alpha_1^*, \boldsymbol{\theta}_{-\alpha_1}, \phi|\mathbf{t}, \delta)\pi(\alpha_1^*)}{L(\alpha_1, \boldsymbol{\theta}_{-\alpha}, \phi|\mathbf{t}, \delta)\pi(\alpha_1)} \frac{q[\alpha_1]}{q[\alpha_1^*]}.$$

In order to implement this case, one may set $q[\alpha_1^*]$ as the prior distribution, i.e., $q[\alpha_1^*] = \pi(\alpha_1^*)$. Then, A_{α_1} is given by the likelihood ratios,

$$A_{\alpha_1} = \frac{L(\alpha_1^*, \boldsymbol{\theta}_{-\alpha_1}, \phi|\mathbf{t}, \delta)}{L(\alpha_1, \boldsymbol{\theta}_{-\alpha}, \phi|\mathbf{t}, \delta)}. \tag{8}$$

This algorithm is implemented as follows.

- **Independent Metropolis–Hastings Algorithm**: Let the current state of the Markov chain be $\left(\alpha_1^{(l-1)}, \boldsymbol{\theta}_{-\alpha}^{(l-1)}, \phi^{(l)}\right)$. For the l-th iteration of the algorithm do the following:

  (1) Generate α_1^* from the prior distribution $\alpha_1^* \sim \Gamma(a_{11}, a_{12})$;
  (2) Calculate $\psi(\alpha_1^*|\alpha_1) = min(1, A_{\alpha_1})$, where A_{α_1} is given by (8);
  (3) Generate $U \sim \mathcal{U}(0,1)$. If $u \le \psi(\alpha_1^*|\alpha_1)$ accept α_1^* and set $\alpha_1^{(l)} = \alpha_1^*$. Otherwise, reject α_1^* and set $\alpha_1^{(l)} = \alpha_1^{(l-1)}$.

Although the choice of the prior distribution as the candidate generating density may be mathematically attractive, it usually leads to a slow convergence of the algorithm. This happens when vague prior information is available and prior distribution has large variance. As a consequence, many of the proposed values are rejected.

An alternative is to explore the neighborhood of the current value of the Markov chain to propose a new value. This method is termed the random walk Metropolis (RWM). In the RWM, the candidate value α_1^* is generated from a symmetric density $g(\cdot)$. That is, we set up $q[\alpha_1^*|\alpha_1] = g(|\alpha_1 - \alpha_1^*|)$ and the probability of generating a move from α_1 to α_1^* depends only on the distance between them. For this case, A_{α_1} given in (7) simplifies to

$$A_{\alpha_1} = \frac{L(\alpha_1^*, \boldsymbol{\theta}_{-\alpha_1}, \phi|\mathbf{t}, \delta)\pi(\alpha_1^*)}{L(\alpha_1, \boldsymbol{\theta}_{-\alpha_1}, \phi|\mathbf{t}, \delta)\pi(\alpha_1)} \tag{9}$$

since the proposal kernels from numerator and denominator cancel.

In order to implement the RWM it is necessary to simulate α_1^* setting $\alpha_1^* = \alpha_1 + \varepsilon$, where ε is a random perturbation generated from a Normal distribution with mean 0 and variance $\sigma^2_{\alpha_1}$, $\varepsilon \sim \mathcal{N}(0, \sigma^2_{\alpha_1})$, meaning that $\alpha_1^* \sim \mathcal{N}(\alpha_1, \sigma^2_{\alpha_1})$. This algorithm is implemented as follows.

- **Random Walk Metropolis Algorithm**: Let the current state of the Markov chain be $\left(\alpha_1^{(l-1)}, \boldsymbol{\theta}_{-\alpha_1}^{(l-1)}, \phi^{(l)}\right)$. For the l-th iteration of the algorithm, $l = 1, \ldots, L$, do the following:

  (1) Generate $\varepsilon \sim \mathcal{N}(0, \sigma^2_{\alpha_1})$ and set $\alpha_1^* = \alpha_1^{(l-1)} + \varepsilon$;
  (2) Calculate $\psi(\alpha_1^*|\alpha_1) = min(1, A_{\alpha_1})$, where A_{α_1} is given by (9);
  (3) Generate $U \sim \mathcal{U}(0,1)$. If $u \le \psi(\alpha_1^*|\alpha_1)$ accept α_1^* and set $\alpha_1^{(l)} = \alpha_1^*$. Otherwise, reject α_1^* and set $\alpha_1^{(l)} = \alpha_1^{(l-1)}$.

An issue in RWM is how to choose the value of $\sigma^2_{\alpha_1}$. It has a strong influence on the efficiency of the algorithm. If $\sigma^2_{\alpha_1}$ is too small, the random perturbations will be small in magnitude and almost all will be accepted. The consequence is that it will take a large number of iterations to explore the entire state-space. On the other hand, if $\sigma^2_{\alpha_1}$ is large there will be many rejections of the proposed values, slowing down the convergence. More details on this issue can be found in [23,26–28].

Typically, one may fix the value of $\sigma^2_{\alpha_1}$ by testing some values on a few pilot runs and then choosing a value whose acceptance ratio lies between 20% and 30% (see, for example, [24,25]). Thus, after a pilot run we set up $\sigma^2_{\alpha} = 1$.

3.1.2. Slice Sampling Algorithm

An alternative to the IMH and RWM sampling from some generic distribution is the slice sampling algorithm. This algorithm is a type of Gibbs sampling based on the simulation of specific uniform random variables. Here we explain the algorithm slice sampling in the context of the simulation of α_1. The sampling procedure for α_2 is similar. More details about SS can be found in [13].

In SS, an auxiliary variable U is introduced and the joint distribution $\pi(\alpha_1, U|\mathbf{t}, \delta, \theta_{-\alpha_1}, \phi)$ is given by a uniform distribution over the region $\mathbb{U} = \{(\alpha_1, u) : 0 < u < \kappa(\alpha_1)\}$ below the curve defined by $\kappa(\alpha_1)$. From (4), we have

$$\kappa(\alpha_1) = \alpha_1^{a_{11}+r_1-1} \exp\left\{\alpha_1 \left(\sum_{i=1}^{n} \delta_{i1} log(t_{i1}) - a_{12}\right) - \beta_1 \sum_{i=1}^{n} t_{i1}^{\alpha_1}\right\} \prod_{i=1}^{n} \Psi_i(\boldsymbol{\theta}, \phi|\mathbf{t}, \delta). \tag{10}$$

Marginalizing $\pi(\alpha_1, U|\mathbf{t}, \delta, \theta_{-\alpha_1}, \phi)$ over U yields $\pi(\alpha_1|\mathbf{t}, \delta, \theta_{-\alpha_1}, \phi)$, so sampling from $\pi(\alpha_1, U|\mathbf{t}, \delta, \theta_{-\alpha_1}, \phi)$ and discarding U is equivalent to sampling from $\pi(\alpha_1|\mathbf{t}, \delta, \theta_{-\alpha_1}, \phi)$.

As sampling from $\pi(\alpha_1, U|\mathbf{t}, \delta, \theta_{-\alpha_1}, \phi)$ is not straightforward, we implemented a Gibbs sampling algorithm where at every iteration l, we first generate $U^{(l)} \sim \mathcal{U}\left(0, \kappa\left(\alpha_1^{(l-1)}\right)\right)$ and then sample $\alpha_1^{(l)} \sim \mathcal{U}(A)$, where $A = \{\alpha_1 : u^{(l)} < \kappa(\alpha_1)\}$. However, as the inverse of $\kappa(\alpha_1)$ cannot be obtained analytically, we adopted the following procedure to update α_1:

(i) Let $\lambda = 0.01$ and $\tilde{A}$ be an empty set.

(a) For $m = 1, 2, \ldots$:
Set $\alpha_1^{-(m)} = \alpha_1^{(l-1)} - m\lambda$
If $u^{(l)} < \kappa\left(\alpha_1^{-(m)}\right)$ do $\tilde{A} = \tilde{A} \cup \left\{\alpha_1^{-(m)}\right\}$ else break

(b) For $m = 1, 2, \ldots$:
Set $\alpha_1^{+(m)} = \alpha_1^{(l-1)} + m\lambda$
If $u^{(l)} < \kappa\left(\alpha_1^{+(m)}\right)$ do $\tilde{A} = \tilde{A} \cup \left\{\alpha_1^{+(m)}\right\}$ else break

(ii) Generate $\alpha_1^{(l)} \sim \mathcal{U}(min(\widetilde{A}), max(\widetilde{A}))$.

This algorithm is implemented as follows.

- **Slice sampling algorithm**: Let the current state of the Markov chain be $\left(\alpha_1^{(l-1)}, \boldsymbol{\theta}_{-\alpha_1}^{(l-1)}, \phi^{(l-1)}\right)$ and $u^{(l-1)}$. For the l-th iteration of the algorithm, $l = 1, \ldots, L$:

  (1) Generate $U^{(l)} \sim \mathcal{U}\left(0, \kappa\left(\alpha_1^{(l-1)}\right)\right)$, where $\kappa(\cdot)$ is given by (10).
  (2) obtain $\widetilde{A}$, conditional on $u^{(l)}$.
  (3) Generate $\alpha_1^{(l)} \sim \mathcal{U}(min(\widetilde{A}), max(\widetilde{A}))$.

3.2. *MCMC for β_j and ϕ*

Note from (5) that the conditional posterior distribution for the scale parameter β_1, $\pi(\beta_1|\mathbf{t}, \delta, \boldsymbol{\theta}_{-\beta_1}, \phi)$, is given by the kernel of a Gamma distribution with parameters $b_{11} + r_{11}$ and

$b_{12} + \sum_{i=1}^{n} t_{i1}^{\alpha_1}$ multiplied by $\eta(\beta_1) = \prod_{i=1}^{n} \Psi_i(\boldsymbol{\theta}, \phi|\mathbf{t}, \delta)$. In other words, $\pi(\beta_1|\mathbf{t}, \delta, \boldsymbol{\theta}_{-\beta_1}, \phi)$ may be written as $\pi(\beta_1|\mathbf{y}, \theta_{-\beta_1}) \propto \eta(\beta_1)h(\beta_1)$, where $h(\beta_1)$ is the density of the Gamma distribution $\Gamma\left(b_{11} + r_{11}, b_{12} + \sum_{i=1}^{n} t_{i1}^{\alpha_1}\right)$ with $\eta(\beta_1)$ being uniformly bounded. Thus, we set up the candidate generating density for β_1 as $q(\beta_1^*|\beta_1) = h(\beta_1^*)$. The acceptance probability for the generated value β_1^* is given by $\psi(\beta_1^*|\beta_1) = min(1, A_{\beta_1})$, where

$$A_{\beta_1} = \frac{\eta(\beta_1^*)}{\eta(\beta_1)}. \tag{11}$$

This algorithm is implemented as follows.

- **Metropolis–Hastings Algorithm**: Let the current state of the Markov chain be $\left(\beta_1^{(l-1)}, \boldsymbol{\theta}_{-\beta_1}^{(l-1)}, \phi^{(l-1)}\right)$, where $\boldsymbol{\theta}_{-\beta_1}^{(l-1)} = \left(\alpha_1^{(l)}, \alpha_2^{(l-1)}, \beta_2^{(l-1)}\right)$. For the l-th iteration of the algorithm, $l = 1, \ldots, L$:

  (1) Generate $\beta_1^* \sim \Gamma\left(b_{11} + r_{11}, b_{12} + \sum_{i=1}^{n} t_{i1}^{\alpha_1^{(l)}}\right)$.

  (2) Calculate $\psi(\beta_1^*|\beta_1) = min(1, A_{\beta_1})$, where A_{β_1} is given by (11).

  (3) Generate $U \sim \mathcal{U}(0,1)$. If $u \leq \psi(\beta_1^*|\beta_1)$ accept β_1^* and set $\beta_1^{(l)} = \beta_1^*$. Otherwise, reject β_1^* and set $\beta_1^{(l)} = \beta_1^{(l-1)}$.

The Metropolis–Hastings algorithm for updating β_2 is similar. To update the dependence parameter ϕ conditional on the remaining parameters $\boldsymbol{\theta} = (\alpha_1, \beta_1, \alpha_2, \beta_2)$, we used the following IMH algorithm. Let $\mathcal{G}_\phi$ be a grid from -1 to 1 with increments of 0.1. Consider $[I_a, I_{a+1})$, an interval defined by two adjacent grid values of $\mathcal{G}_\phi$ where a is the index of the a-th value of the grid for $a = 1, \ldots, 20$. For example, for $a = 1$ we have the interval $[-1, -0.9)$; for $a = 11$, we have the interval $[0, 0.1)$; and for $a = 20$ we have the interval $[0.9, 1)$. Then generate the a candidate value ϕ^* as follows:

(i) If the current value of ϕ is in the interval (I_1, I_2), then generate ϕ^* from one of the two following Uniform distributions

$$\phi^* \sim \begin{cases} \mathcal{U}(I_1, I_2), & \text{with probability } 1/2, \\ \mathcal{U}(I_2, I_3), & \text{with probability } 1/2. \end{cases}$$

For this case, we generate an auxiliary variable $U \sim \mathcal{U}(0,1)$; if $u \leq 1/2$, then we generate ϕ^* from $\mathcal{U}(I_1, I_2)$, $\phi^* \sim \mathcal{U}(I_1, I_2)$, otherwise we generate ϕ^* from $\mathcal{U}(I_2, I_3)$, $\phi^* \sim U(I_2, I_3)$.

(ii) If the current value of ϕ is in (I_{20}, I_{21}), then generate ϕ^* from one of the two following uniform distributions

$$\phi^* \sim \begin{cases} \mathcal{U}(I_{19}, I_{20}), & \text{with probability } 1/2, \\ \mathcal{U}(I_{20}, I_{21}), & \text{with probability } 1/2, \end{cases}$$

Similarly to item (i), we generate an auxiliary variable $U \sim \mathcal{U}(0,1)$; if $u \leq 1/2$, then $\phi^* \sim \mathcal{U}(I_{20}, I_{21})$, otherwise $\phi^* \sim U(I_{19}, I_{20})$.

(iii) If the current value of ϕ is in the interval (I_a, I_{a+1}), for $a \neq 1$ and $a \neq 20$, then generate ϕ^* from one of three following uniform distributions

$$\phi^* \sim \begin{cases} \mathcal{U}(I_{a-1}, I_a), & \text{with probability } 1/3, \\ \mathcal{U}(I_a, I_{a+1}), & \text{with probability } 1/3, \\ \mathcal{U}(I_{a+1}, I_{a+2}), & \text{with probability } 1/3. \end{cases}$$

For this case, we generate an auxiliary variable $U \sim \mathcal{U}(0,1)$; if $u \leq 1/3$, then we generate ϕ^* from $\mathcal{U}(I_{a-1}, I_a)$, $\phi^* \sim \mathcal{U}(I_a, I_{a+1})$; if $1/3 < u \leq 2/3$, then we generate ϕ^* from $\mathcal{U}(I_a, I_{a+1})$, $\phi^* \sim \mathcal{U}(I_a, I_{a+1})$; and if $u > 2/3$, we generate ϕ^* from $\mathcal{U}(I_{a+1}, I_{a+2})$, $\phi^* \sim \mathcal{U}(I_{a+1}, I_{a+2})$.

The acceptance probability is given by $\psi[\phi^*|\phi] = min(1, A_\phi)$, where $A_\phi = \frac{L(\phi^*,\boldsymbol{\theta}|\mathbf{t},\delta)}{L(\phi,\boldsymbol{\theta}|\mathbf{t},\delta)} P_\phi$ for $P_\phi = 1$ or $P_\phi = \frac{1/2}{1/3}$ according to items (i)–(iii) described above. This algorithm is implemented as follows.

- **IMH algorithm for** ϕ: Let the current state of the Markov chain be $\left(\boldsymbol{\theta}^{(l)}, \phi^{(l-1)}\right)$. For the l-th iteration of the algorithm, $l = 2, \ldots, L$:

  (1) Generate ϕ^* according to one of the items (i), (ii) or (iii) described above.
  (2) Calculate $\psi(\phi^*|\phi) = min(1, A_\phi)$.
  (3) Generate $U \sim \mathcal{U}(0,1)$. If $u \le \psi(\phi^*|\phi)$ accept ϕ^* and set $\phi^{(l)} = \phi^*$. Otherwise, reject ϕ^* and set $\phi^{(l)} = \phi^{(l-1)}$.

3.3. MCMC Algorithms

Using the algorithms IMH, RWM, SS and MH described above, we implemented three MCMC algorithms:

- Algorithm A_1: Parameters α_j's are updated via IMH,
- Algorithm A_2: Parameters α_j's are updated via RWM,
- Algorithm A_3: Parameters α_j's are updated via SS.

For these three algorithms, the parameters β_j and ϕ are updated via MH and IMH, as described in Section 3.2, for $j = 1, 2$.

After defining the algorithms, we ran them for L iterations and a burn-in B. We also consider jumps of size J, i.e., only 1 drawn from every J was extracted from the original sequence obtaining a sub sequence of size $S = [(L-B)/J]$ to make inferences.

The estimates for parameters are given by

$$\tilde{\alpha}_j = \frac{1}{S}\sum_{l=1}^{L} \alpha_j^{(K(l))}; \quad \tilde{\beta}_j = \frac{1}{S}\sum_{l=1}^{L} \beta^{(K(l))} \quad \text{and} \quad \tilde{\phi} = \frac{1}{S}\sum_{l=1}^{L} \phi^{(K(l))}, \tag{12}$$

where $\theta^{(K(l))}$ is the value generated for θ in the $K(l) = [(B+1+lJ)]$-th iteration of the algorithm, for $j = 1, 2$ and $l = 1, \ldots, L$.

4. Simulation Study

In this section, we present the comparison between the performances of the three algorithms applied to simulated data sets. Simulated random samples of sizes $n = 25, 50, 100$ and 250 with 0%, 5%, 10%, 20% and 30% random right-censored were generated to represent small, medium and large data sets. Using these, we generated four simulated data sets with fixed parameters, as specified in Table 1.

Data set D_1 has two increasing hazard functions with a positive dependence parameter, while data set D_2 has a constant and increasing hazard function with a negative dependence parameter. Data set A_3 has parameters to produce a decreasing and a constant hazard function with weak dependence, while data set A_4 has strong dependence and two increasing hazard functions.

Table 1. Parameter values for simulated data sets.

Data Set	Parameters				
	α_1	β_1	α_2	β_2	ϕ
D_1	2.00	1.00	3.00	1.00	0.50
D_2	1.00	2.00	2.00	0.50	−0.75
D_3	0.75	1.50	1.00	2.00	0.05
D_4	1.80	2.40	2.20	1.20	0.95

The simulation procedure to generate n observations (t_{i1}, t_{i2}), for $i = 1, \cdots, n$, is given by the following steps:

(i) Set up the sample size n and set $i = 1$;
(ii) Generate the censoring times $C_{ij} \sim U(0, \tau_j)$, where τ_j controls the percentage of censored observations, for $j = 1, 2$;
(iii) Generate uniform values $u_{ij} \sim U(0, 1)$, $j = 1, 2$ and calculate w_i, the solution of the nonlinear equation $u_{i2} - \frac{w_i[1-\phi(1-w_i)]}{[1-\phi(1-u_{i1})(1-w_i)]^2} = 0$. Here we used the *rootsolve* package and the *uniroot.all* command from *R* software to solve the nonlinear equation and obtain w_i;
(iv) Calculate $T_{i1} = (-\log(u_{i1})/\beta_1)^{1/\alpha_1}$ and $T_{i2} = (-\log(w_i)/\beta_2)^{1/\alpha_2}$;
(v) Calculate the times $t_{ij} = \min(T_{ij}, C_{ij})$ and the censorship indicators δ_{ij}, which are equal to 1 if $t_{ij} < T_{ij}$ and 0 otherwise, for $j = 1, 2$;
(vi) Set $i = i + 1$. If $i = n$ stop. Otherwise, return to step (ii).

We generated $M = 200$ different simulated data sets according to steps (i)–(vi) described above and the parameters were estimated according to algorithms A_1, A_2 and A_3.

We used hyperparameters $a_{j1} = a_{j2} = b_{j1} = b_{j2} = 0.01$ to obtain prior distributions with large variance, for $j = 1, 2$. For the m-th generated data set, we applied algorithms A_1, A_2 and A_3 fixing L = 55,000 iterations, burn-in B = 5000 and $J = 10$.

Comparison of the algorithms was made using the sample Root Mean Square Error (RMSE), given by

$$\text{RMSE} = \sqrt{\frac{1}{M}\sum_{m=1}^{M}\left[\sum_{j=1}^{2}\left(\hat{\alpha}_j^{(m)} - \alpha_j\right)^2 + \left(\hat{\beta}_j^{(m)} - \beta_j\right)^2\right] + \left(\hat{\phi}^{(m)} - \phi\right)^2}.$$

A smaller RMSE indicates better overall quality of the estimates.

Table 2 presents the RMSE value for each simulated data set by algorithm, sample size and percentage of censorship. The smaller RMSE value for each sample size and percentage of censorship is highlighted in bold. For the three algorithms, by fixing the sample size and increasing the censuring percentage (% cens.), the RMSE values increased. When the sample size increases at a fixed percentage of censures, the RMSE values decrease, consequently improving the precision of the estimators.

Based on the results presented in Table 2, for the smaller sample size $n = 25$, the algorithm A_3 (with SS) outperformed algorithm A_1 (with IMH) and algorithm A_2 (with RWM), i.e., it gave a smaller RMSE value for all percentages of censures. This better performance also happened for data sets D_3 and D_4 for $n = 50$. For all other simulated cases, the algorithm A_2 outperformed algorithms A_1 and A_3. An exception is the case with $n = 250$ and 0% of censuring in data set D_2, in which algorithm A_1 had a better performance. These results suggest a possible complementarity between algorithms A_2 and A_3, where algorithm A_2 performs better for higher sample sizes and algorithm A_3 performs better for smaller sample sizes.

We verified the convergence of algorithms A_1, A_2 and A_3 using the effective sample size [14] and the integrated autocorrelation time (IAT). The effective sample size (ESS) is the number of effectively independent draws from the posterior distribution. Method with larger ESS are the most efficient. The IAT is a MCMC diagnostic that estimates the average number of autocorrelated samples required to produce one independent sample draw. Lower IAT is means more efficiency. The EES and IAT values were obtained using the *coda* and *LaplacesDemon*. Both packages are available in the *R* software.

Tables A1 and A2 in Appendix A show the average of ESS and IAT values for each algorithm by parameter for data set D_1. Algorithm A_3 showed a better performance than algorithms A_1 and A_2, i.e., it had the highest ESS values and smallest IAT values by parameter for all simulated cases. Note that algorithm A_1 had the worst results, especially for simulated values for α_j, $j = 1, 2$. Results for data sets D_2, D_3 and D_4 were similar.

Table 2. Root mean square error (RMSE) by algorithm for data sets D_1, D_2, D_3 and D_4.

Sample Size	**% of Censures**	**Data Set D_1**			**Data Set D_2**			**Data Set D_3**			**Data Set D_4**		
		Algorithm			**Algorithm**			**Algorithm**			**Algorithm**		
		A_1	A_2	A_3	A_1	A_2	A_3	A_1	A_2	A_3	A_1	A_2	A_3
$n = 25$	0%	0.3678	0.3717	**0.3581**	0.3774	0.3781	**0.3458**	0.3375	0.3370	**0.3368**	1.1085	1.0888	**1.0883**
	5%	0.4078	0.3869	**0.3597**	0.3861	0.3901	**0.3736**	0.3586	0.3573	**0.3523**	1.1325	1.1305	**1.1278**
	10%	0.4189	0.4012	**0.3670**	0.4144	0.4259	**0.4135**	0.3687	0.3675	**0.3611**	1.1428	1.1396	**1.1323**
	20%	0.4245	0.4153	**0.3772**	0.4472	0.4648	**0.4381**	0.3772	0.3729	**0.3727**	1.1726	1.1714	**1.1711**
	30%	0.4362	0.4543	**0.3989**	0.5335	0.5614	**0.5303**	0.3994	0.3990	**0.3944**	1.2078	1.1946	**1.1925**
$n = 50$	0%	0.2595	**0.2507**	0.2678	0.2633	**0.2552**	0.2573	0.2162	0.2112	**0.2048**	1.0397	1.0318	**1.0312**
	5%	0.2663	**0.2652**	0.2699	0.2641	**0.2601**	0.2719	0.2239	0.2283	**0.2233**	1.0470	1.0442	**1.0403**
	10%	0.2831	**0.2806**	0.2814	0.2959	**0.2683**	0.2844	0.2390	0.2457	**0.2269**	1.0483	1.0453	**1.0433**
	20%	0.2846	**0.2820**	0.2863	0.2966	**0.2820**	0.3026	0.2719	0.2546	**0.2366**	1.0517	1.0528	**1.0513**
	30%	0.2983	**0.2885**	0.3104	0.3245	**0.3170**	0.3182	0.2828	0.2776	**0.2736**	1.0915	1.0666	**1.0550**
$n = 100$	0%	0.1822	**0.1819**	0.1833	0.1917	**0.1816**	0.1878	0.1664	**0.1657**	0.1702	1.0153	**1.0041**	1.0124
	5%	0.1953	**0.1851**	0.1859	0.1925	**0.1857**	0.1914	0.1769	**0.1755**	0.1782	1.0228	**1.0063**	1.0152
	10%	0.1982	**0.1924**	0.1927	0.2026	**0.2019**	0.2023	0.1788	**0.1760**	0.1791	1.0239	**1.0088**	1.0157
	20%	0.1996	**0.1964**	0.2074	0.2029	**0.2028**	0.2047	0.1934	**0.1832**	0.1879	1.0282	**1.0092**	1.0177
	30%	0.2131	**0.2122**	0.2144	0.2463	**0.2112**	0.2211	0.2094	**0.1967**	0.2143	1.0291	**1.0128**	1.0265
$n = 250$	0%	0.1138	**0.1123**	0.1130	**0.1075**	0.1079	0.1115	0.1156	**0.1140**	0.1162	0.9934	**0.9923**	0.9936
	5%	0.1141	**0.1136**	0.1149	0.1206	**0.1141**	0.1129	0.1179	**0.1146**	0.1183	0.9970	**0.9963**	0.9968
	10%	0.1165	**0.1164**	0.1167	0.1244	**0.1199**	0.1237	0.1186	**0.1159**	0.1197	0.9985	**0.9977**	0.9972
	20%	0.1224	**0.1216**	0.1229	0.1258	**0.1252**	0.1287	0.1303	**0.1260**	0.1273	0.9991	**0.9984**	0.9991
	30%	0.1374	**0.1333**	0.1344	0.1677	**0.1398**	0.1458	0.1391	**0.1328**	0.1329	0.9999	**0.9993**	0.9997

Appendix B presents an empirical convergence check for the sampled values for α_1 for each algorithm. As shown in Figure A1, the generated values for α_1 by algorithm A_1 did not mix well and the stability for the ergodic mean and estimated autocorrelation were not satisfactory. On the other hand, the values generated by algorithms A_2 and A_3 were well mixed and present satisfactory stability for the ergodic mean and autocorrelation. As an illustration of convergence diagnostic, Figure A1(j–l) shows the Gelman plot for the sequence of α_1 values in two chains by each algorithm. As can be seen in the figure, the number of iterations was sufficient for algorithms A_2 and A_3 to reach convergence, but not for algorithm A_1. In addition, the scale reduction factor of the Gelman–Rubin diagnostic [29] for each parameter in algorithms A_2 and A_3 were smaller than 1.1, meaning that there is no indication of non-convergence. This implies a faster convergence of algorithms A_2 and A_3 in relation to algorithm A_1. For β_1 sampled values, the three algorithms present satisfactory properties, i.e., good mixing, and satisfactory stability for ergodic mean and autocorrelation (see Figure A2 in Appendix B).

The results indicate that algorithm A_3 (SS for α_j) is an effective alternative to algorithms A_1 (with IMH for α_j) and A_2 (with RWM for α_j) to simulate samples from the posterior distribution of bivariate survival models based on the Ali–Mikhail–Haq copula with marginal Weibull distributions.

5. Application to a Real Data Set

Next, we examine the performance of algorithms A_1, A_2 and A_3 on the diabetic retinopathy data set described in [15], which is available in the R software 'survival' package [16]. This data set consists of the follow-up times of 197 diabetic patients under 60 years of age. The main objective of the study was to evaluate the effectiveness of the photocoagulation treatment for proliferative retinopathy. The treatment was randomly assigned to one eye of each patient and the other eye was taken as a control.

Let (T_1, T_2) be the bivariate times, where T_1 is the time to visual loss for the treatment eye and T_2 is the time to visual loss for the control eye. The percentage of censure times for each variable is 72.59% (143 observations) for T_1 and 48.73% (96 observations) for T_2.

We used (1) to model this data with Weibull marginal distributions with parameters α_j and β_j and dependence parameter ϕ.

We compared the performances of the algorithms using the RMSE in relation to the empirical distribution function,

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{2}\left(\hat{F}_j(t_{ij}) - F_j(t_{ij})\right)^2},$$

where $\hat{F}_j(t_{ij})$ is obtained by substituting the estimates of α_j, β_j and ϕ (obtained by each algorithm); and $F_j(t_{ij})$ is the empirical distribution function obtained from the Kaplan–Meier estimates, for $j = 1, 2$ and $i = 1, \ldots, n$.

We ran the three algorithms using the same number of iterations, burn-in, thinning and hyperparameters values used with the simulation data. Table 3 shows the parameters estimates, the credibility intervals (95%) and RMSE values by algorithm. For this data set, the algorithm A_3 (with SS for α_j) gave the smaller RMSE value.

Figure 1 shows the estimated survival functions by algorithms A_1 (red line) and A_3 (blue line). The step functions (black lines) are the Kaplan–Meier estimates. The estimated curves by algorithms A_1 and A_2 are very close and so we show only the curve estimated by A_1, in order to provide a good visualization. The Kaplan–Meier estimates were obtained using the survival package and the survfit command in the *R* software.

Table 4 shows the ESS and IAT values for the sequences generated by algorithms A_1, A_2, and A_3. Algorithm A_3 had a better performance than algorithms A_1 and A_2, i.e., the highest ESS value and the lowest IAT value per parameter.

We also compared the performances of the algorithms in relation to the sequences generated for each parameter. Figure 2 shows the traceplots, the ergodic means, and the autocorrelations for sequences of α_1 values simulated by algorithms A_1, A_2 and A_3.

Table 3. Parameters estimates and RMSE by algorithm.

Algorithm	Parameter					RMSE Value
	α_1	β_1	α_2	β_2	ϕ	
A_1	0.7624 (0.5999,0.9361)	0.0186 (0.0087, 0.0338)	0.8399 (0.7607, 0.9353)	0.0294 (0.0195, 0.0414)	0.7159 (0.3765, 0.9637)	0.4227
A_2	0.7757 (0.5929, 0.9853)	0.0179 (0.0071, 0.0343)	0.8308 (0.6897, 0.9679)	0.0310 (0.0172, 0.0515)	0.7148 (0.3560, 0.9600)	0.4619
A_3	0.6438 (0.5103, 0.7967)	0.0289 (0.0142, 0.0482)	0.7015 (0.5910, 0.8273)	0.0494 (0.0293, 0.0746)	0.7266 (0.3675, 0.9715)	0.3562

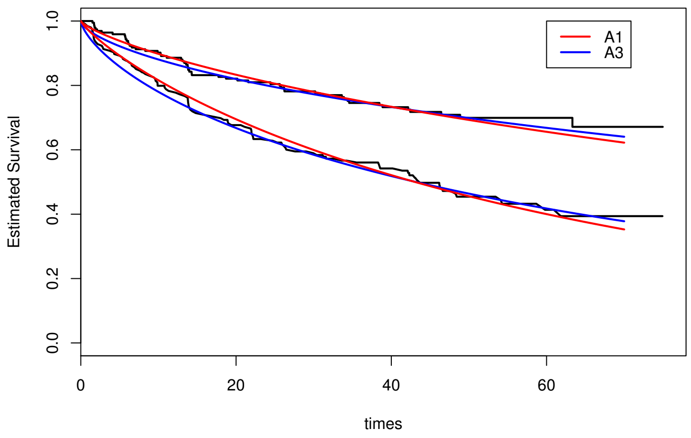
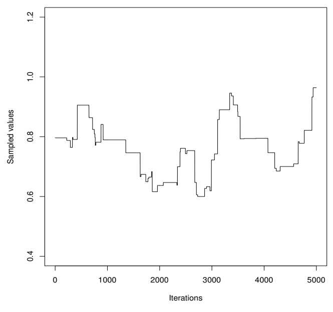


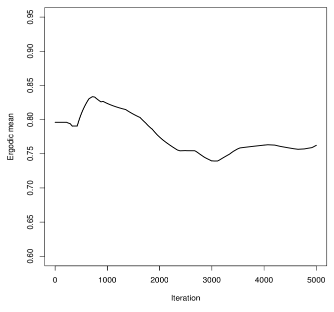
Figure 1. The estimated survival function for algorithms A_1 and A_3.

Table 4. Integrated autocorrelation time (IAT) and effective sample size (ESS) values for algorithms A_1, A_2 and A_3.

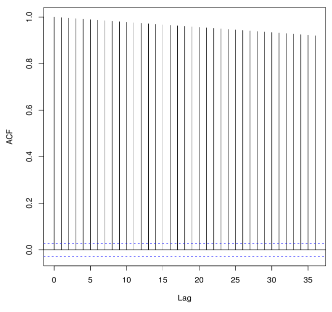
Parameter	ESS			IAT		
	A_1	A_2	A_3	A_1	A_2	A_3
α_1	5.4650	159.8655	791.0559	435.0485	34.2212	6.4039
β_1	6.5887	205.4812	880.9221	81.9980	26.8373	5.6359
α_2	8.1633	134.7412	227.6705	327.9376	35.6760	24.6754
β_2	16.1893	133.8282	230.9487	36.7590	30.5560	21.1668
ϕ	2443.3791	2400.0097	2461.1781	2.3426	2.3348	2.2813



(**a**) α_1 by A_1.

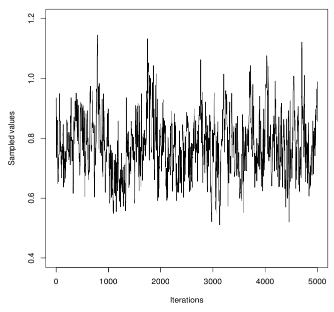


(**b**) β_1 by A_1.

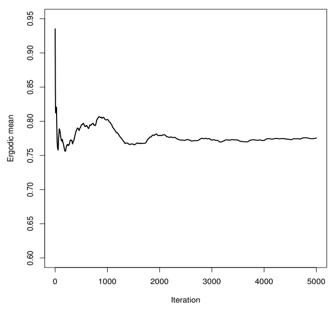


(**c**) ϕ by A_1.

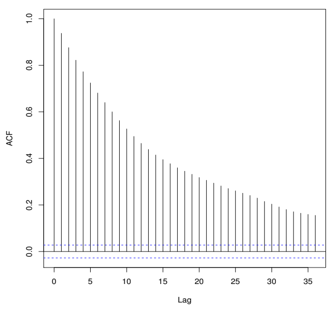
Figure 2. *Cont.*



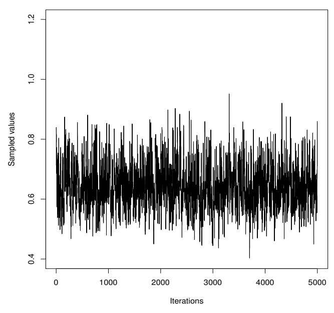
(**d**) α_1 by A_2.



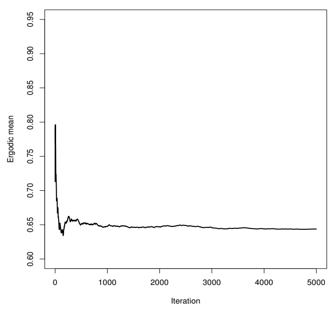
(**e**) β_1 by A_2.



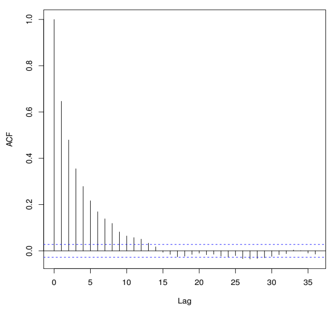
(**f**) ϕ by A_2.



(**g**) α_1 by A_3.



(**h**) β_1 by A_3.

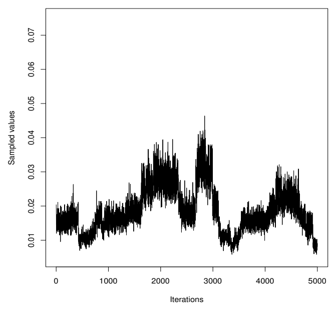


(**i**) ϕ by A_3.

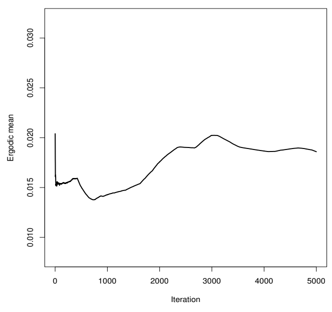
Figure 2. Traceplot, ergodic mean and autocorrelation for sequences produced by algorithms A_1, A_2 and A_3 for α_1.

It can be observed in these graphs that the α_1 values generated by the IMH (algorithm A_1) has poor mixing, does not show satisfactory stability for the ergodic mean, and the autocorrelation is high for long lags. On the other hand, the values generated by the RWM (algorithm A_2) and SS (algorithm A_3) are better mixed and present satisfactory stability for the ergodic mean. However, the sequence produced by the SS presents the steepest decreasing autocorrelation. Figure 3 shows the same graphs for parameter β_1. As can be seen, for β_1 the performances of the three algorithms are satisfactory. These results, together with those presented by the RMSE, show that for the data set analyzed here SS provides a better performance than IMH or RWM.

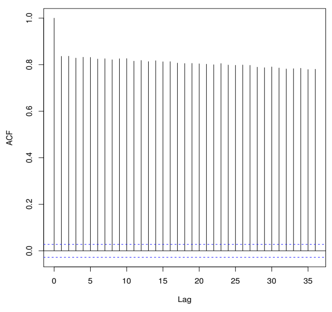
Figure 4 shows the Gelman plot for the simulated values for α_1, β_1 and ϕ in two chains by each algorithm. As can be seen, the number of iterations was sufficient for algorithms A_2 and A_3 to reach the convergence, but not sufficient for algorithm A_1 (Figure 4a,b). The scale reduction factor for each parameter in algorithms A_2 and A_3 are all less than 1.1, while for algorithm A_1 only ϕ presents a scale reduction factor less than 1.1.



(**a**) A_1.

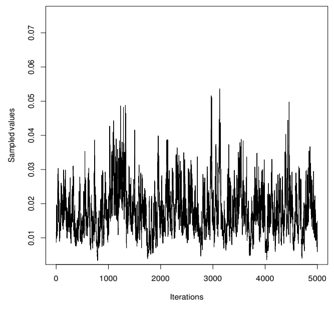


(**b**) A_1.

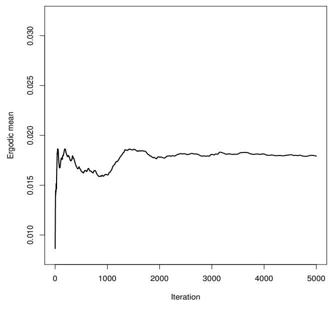


(**c**) A_1.

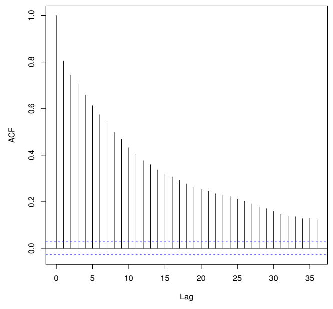
Figure 3. *Cont.*



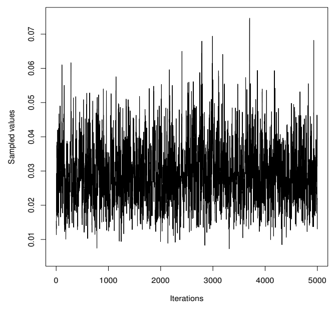
(**d**) A_2.



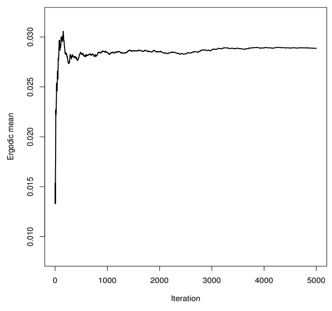
(**e**) A_2.



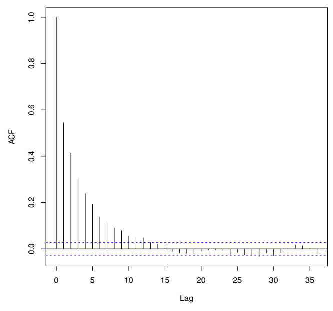
(**f**) A_2.



(**g**) A_3.

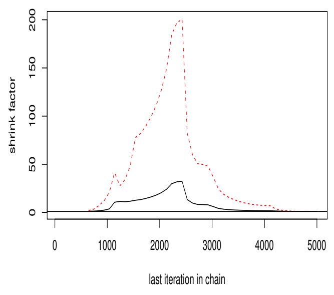


(**h**) A_3.

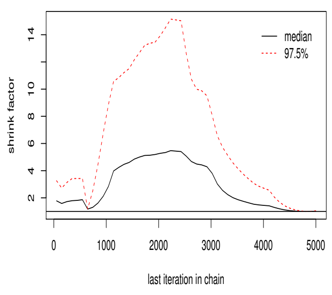


(**i**) A_3.

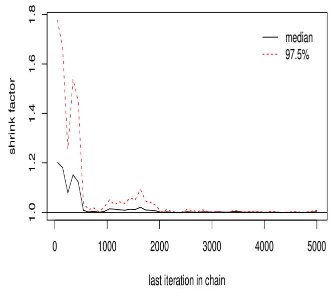
Figure 3. Traceplot, ergodic mean and autocorrelation for sequences produced by algorithms A_1, A_2 and A_3 for β_1.



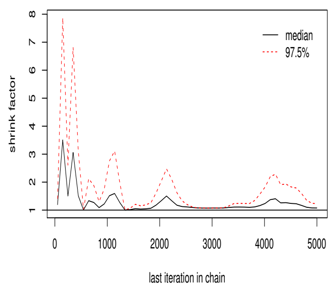
(**a**) α_1 by A_1.



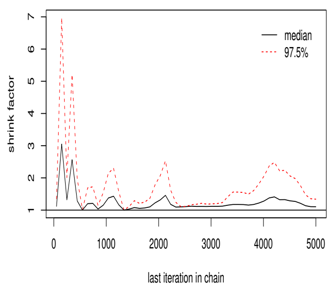
(**b**) β_1 by A_1.



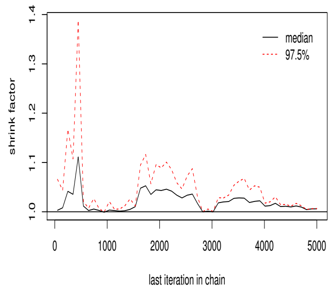
(**c**) ϕ by A_1.



(**d**) α_1 by A_2.

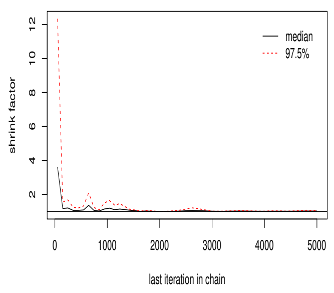


(**e**) β_1 by A_2.

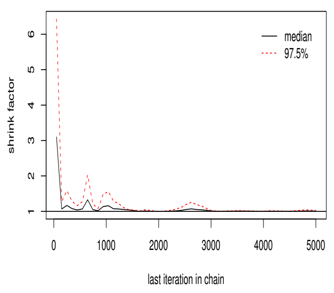


(**f**) ϕ by A_2.

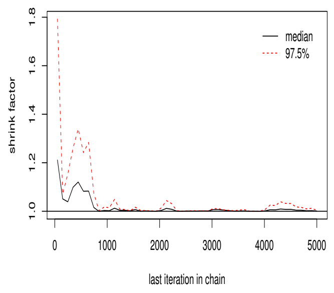
Figure 4. *Cont.*



(g) α_1 by A_3.



(h) β_1 by A_3.



(i) ϕ by A_3.

Figure 4. Gelman plot for two sequences produced by algorithms A_1, A_2 and A_3 for α_1, β_1 and ϕ.

6. Final Remarks

We investigated the performances of three Bayesian computational methods to estimate parameters of a bivariate survival model based on the Ali–Mikhail–Haq copula with marginal Weibull distributions. The performances of the MCMC algorithms were compared using the RMSE criterion. The RMSE values were calculated for different sample sizes and different percentages of censures.

The results obtained from the simulated data sets showed that the RWM and SS algorithms outperformed the IMH algorithm, and that the SS algorithm performed better for lower sample sizes. The results show evidence that MCMC sequences obtained with SS with the same number of iterations L, *burn in* B and thinning value, have better properties (i.e., higher ESS and lower IAT values) than for IMH and RWM, which are standard methods to sample from the joint posterior distribution.

We also illustrate the application of the algorithms using a real data set, available in the literature. The algorithm A_3 (with SS generating the α_j's) presented a better performance when applied to this data set. The criteria used to reach this conclusion were the stability for the ergodic mean, the autocorrelation, the minimum RMSE value, the maximum ESS value, and the minimum IAT value. In addition, the algorithm using SS presented a satisfactory performance in relation to scale factor reduction, and the Gelman plot of the Gelman–Rubin convergence diagnostic.

Our results show that algorithm A_3, which is composed by a mixing of SS for generating α_j, MH for β_j and IMH for ϕ, is an effective algorithm to simulate values from the joint posterior distribution of an AMH copula with Weibull marginal distributions. Moreover, two advantages of SS are that it is easy to implement and it does not need to specify a candidate generating density. A disadvantage in our specific case is that it took longer to perform the simulation when compared with IMH and RWM. The reason for this longer time is that we needed an iterative method to obtain the inverse of the function $\kappa(\alpha_j)$. This was because an analytical solution is not available. All calculations were implemented using the software R and can be obtained from the authors.

An extension of the results obtained here for other Arquimedian copulas as well other marginal distributions and a possible generalization would be a fruitful area for future work.

Author Contributions: The authors E.F.S. and A.K.S. developed the theoretical part of the research. The authors E.F.S., A.K.S. and L.A.M. developed the simulation studies and real data application.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. ESS and IAT Values for Simulated Data Sets

In this section, we present the average of ESS and IAT values for each algorithm by parameter for data set D_1. As discussed in Section 4, Algorithm A_3 presented a better performance than algorithms A_1 and A_2. The results for data sets D_2, D_3 and D_4 are similar.

Table A1. *ESS* by algorithm for data sets D_1.

Sample Size	% of Censures	Algorithm A_1					Algorithm A_2					Algorithm A_3				
		α_1	β_1	α_2	β_2	ϕ	α_1	β_1	α_2	β_2	ϕ	α_1	β_1	α_2	β_2	ϕ
$n = 25$	0%	25.4	1149.9	26.0	1168.4	105.9	1741.7	3493.7	1816.3	3511.8	111.2	4547.7	4110.0	4540.0	4136.9	112.2
	5%	26.4	1360.4	27.4	1311.1	100.6	1758.1	3530.2	1823.3	3563.4	106.8	4569.7	4118.5	4622.4	4125.7	112.0
	10%	27.9	1570.5	28.2	1422.5	97.6	1783.3	3543.0	1827.7	3598.9	99.9	4604.9	4220.7	4672.7	4191.9	105.2
	20%	31.8	2178.7	30.1	1988.6	95.6	1869.0	3943.1	1822.2	3738.9	93.9	4681.8	4275.1	4726.5	4182.3	97.9
	30%	32.9	2293.8	32.7	2146.3	88.5	1931.0	4018.4	1772.0	3885.7	88.1	4782.5	4350.3	4744.4	4329.9	89.6
$n = 50$	0%	19.4	860.7	19.5	1049.2	173.0	1415.2	3259.1	1774.8	3450.9	172.7	4607.70	4132.9	4610.4	4129.5	176.9
	5%	19.6	1061.1	18.7	968.2	167.2	1475.8	3456.2	1796.2	3517.1	167.3	4680.2	4226.3	4698.9	4187.6	169.3
	10%	21.1	1331.7	20.6	1168.2	163.2	1565.6	3662.3	1861.4	3700.1	155.8	4706.1	4237.6	4698.8	4148.0	171.4
	20%	22.5	2134.5	23.1	2005.2	141.6	1668.8	3926.3	1922.5	3804.2	140.0	4825.1	4374.9	4792.8	4299.3	143.6
	30%	24.3	2604.9	24.5	2241.4	127.0	1770.5	4188.2	1989.0	4047.5	132.2	4817.7	4504.1	4819.8	4364.1	133.8
$n = 100$	0%	14.3	817.5	14.8	826.7	316.7	1107.5	3258.6	1518.9	3429.5	323.9	4609.3	4244.3	4668.7	4169.3	325.2
	5%	14.5	899.7	14.5	807.8	304.1	1136.7	3393.6	1549.6	3522.7	290.0	4639.9	4238.7	4689.2	4222.8	311.4
	10%	15.6	1157.9	15.0	938.3	276.9	1199.2	3617.4	1598.7	3698.5	272.9	4729.9	4311.9	4800.5	4295.0	277.3
	20%	16.3	1846.4	16.4	1540.7	260.7	1297.1	3886.4	1706.2	3834.2	265.2	4833.4	4465.1	4827.2	4399.4	271.4
	30%	17.6	3127.3	17.7	2337.1	224.4	1414.1	4292.0	1831.9	4128.8	211.1	4857.6	4475.2	4862.9	4410.8	226.3
$n = 250$	0%	10.3	655.3	10.0	662.7	672.9	712.3	2856.1	1055.4	3236.4	687.8	4588.1	4210.6	4655.5	4275.5	698.8
	5%	10.7	800.5	10.5	816.3	672.3	742.5	3106.1	1083.3	3343.3	640.0	4664.5	4333.8	4734.3	4277.8	693.9
	10%	10.7	1024.2	10.8	951.7	602.3	786.7	3369.7	1128.4	3519.9	607.5	4728.8	4362.8	4757.3	4338.3	620.0
	20%	10.7	1735.2	11.8	1494.5	549.7	863.0	3890.0	1226.9	3845.6	539.6	4741.7	4440.4	4805.1	4451.7	550.0
	30%	12.2	3259.7	12.1	2271.8	466.2	936.6	4279.2	1308.9	4147.7	477.2	4872.7	4625.0	4858.4	4552.6	481.6

Table A2. *IAT* by algorithm for data sets D_1.

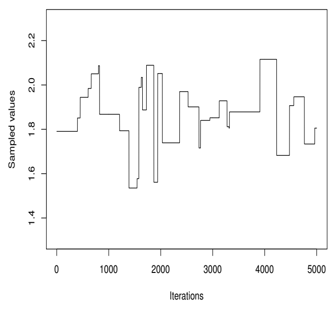
Sample Size	% of Censures	Data Set A_1					Data Set A_2					Data Set A_3				
		α_1	β_1	α_2	β_2	ϕ	α_1	β_1	α_2	β_2	ϕ	α_1	β_1	α_2	β_2	ϕ
$n = 25$	0%	162.7	2.4	162.4	2.3	50.6	3.0	1.5	2.9	1.5	50.2	1.1	1.3	1.1	1.2	50.0
	5%	162.3	2.2	154.0	2.3	52.5	2.9	1.5	2.8	1.5	50.2	1.1	1.2	1.1	1.2	50.0
	10%	152.7	2.0	150.9	2.3	54.1	2.9	1.5	2.8	1.5	54.8	1.1	1.2	1.1	1.2	51.3
	20%	136.8	1.7	136.6	1.9	55.4	2.7	1.3	2.8	1.4	55.8	1.1	1.2	1.1	1.2	54.5
	30%	132.2	1.7	130.4	1.7	59.9	2.6	1.3	3.0	1.4	59.8	1.1	1.2	1.1	1.2	57.6
$n = 50$	0%	208.9	2.3	213.5	2.2	33.2	3.7	1.6	2.9	1.5	32.8	1.1	1.2	1.1	1.2	32.5
	5%	208.7	2.0	233.6	2.2	34.8	3.5	1.5	2.9	1.5	34.5	1.1	1.2	1.1	1.2	34.2
	10%	198.6	1.9	206.5	2.2	35.6	3.3	1.4	2.7	1.4	36.0	1.1	1.2	1.1	1.2	35.2
	20%	183.6	1.6	179.4	1.6	39.5	3.1	1.3	2.7	1.4	39.2	1.1	1.2	1.1	1.2	39.0
	30%	170.5	1.5	170.0	1.6	43.2	2.9	1.2	2.5	1.3	41.9	1.1	1.1	1.1	1.2	40.3
$n = 100$	0%	288.1	2.1	278.2	2.2	17.9	4.6	1.6	3.4	1.5	18.1	1.1	1.2	1.1	1.2	17.2
	5%	284.7	2.2	287.2	2.2	19.7	4.5	1.5	3.3	1.5	20.3	1.1	1.2	1.1	1.2	18.9
	10%	266.8	1.9	271.9	1.9	21.3	4.2	1.4	3.2	1.4	20.5	1.1	1.2	1.1	1.2	20.3
	20%	250.0	1.6	252.8	1.7	22.8	3.9	1.4	3.0	1.4	22.4	1.1	1.1	1.1	1.2	22.3
	30%	233.4	1.3	227.1	1.5	26.5	3.6	1.2	2.8	1.2	27.0	1.1	1.1	1.1	1.2	26.2
$n = 250$	0%	417.9	2.0	418.8	2.0	7.9	7.1	1.8	4.8	1.6	7.9	1.1	1.2	1.1	1.2	7.6
	5%	400.6	1.9	399.7	2.0	8.2	6.8	1.7	4.7	1.6	8.4	1.1	1.2	1.1	1.2	8.1
	10%	391.7	1.8	366.7	1.8	9.1	6.5	1.5	4.5	1.5	9.0	1.1	1.2	1.1	1.2	8.8
	20%	374.6	1.5	355.9	1.6	10.2	5.9	1.3	4.1	1.4	10.3	1.1	1.2	1.1	1.2	10.1
	30%	358.9	1.3	339.2	1.4	11.8	5.5.	1.5	3.9	2.1	11.7	1.1	1.1	1.1	1.1	11.1

Appendix B. Empirical Illustration of the Convergence

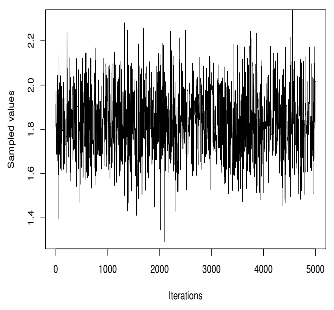
We present here an empirical illustration of the convergence of the simulated sequences for parameters α_1 and β_1. We randomly selected a data set from one of the $M = 200$ generated data sets D_1 with $n = 100$ and $\%cens = 5$ and present the traceplot, graphs showing of the ergodic mean and autocorrelation of the sampled values by algorithm and the Gelman plot.

Figure A1 shows the performance of the algorithms for sampled α_1 values. It can be observed that the IMH (algorithm A_1) does not mix well, it does not have stability for the ergodic mean, and the estimated autocorrelation does not decrease as fast as the other algorithms. The sequences of α_1's generated by RWM and SS are well mixed and present satisfactory stability for the ergodic mean, and the autocorrelation decreases faster, with a clear advantage for algorithm A_3. The Gelman plot indicates that the number of iterations used was sufficient for algorithms A_2 and A_3 to reach the convergence.

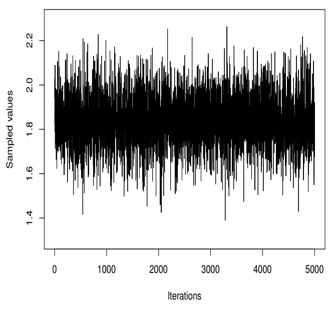
Figure A2 presents the performances of each algorithm for the sequence generated for β_1. As can be observed, the three algorithms present satisfactory properties. The satisfactory performance of the three algorithms is mainly due to the fact that β_1 has a natural candidate generating density with parameters depending on the observed data and values of hyperparameters.



(**a**) A_1.

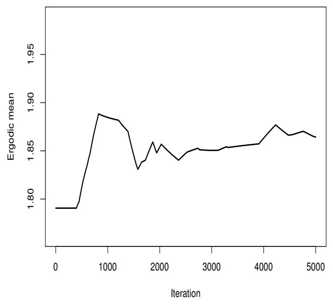


(**b**) A_1.

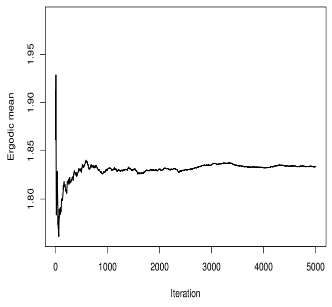


(**c**) A_1.

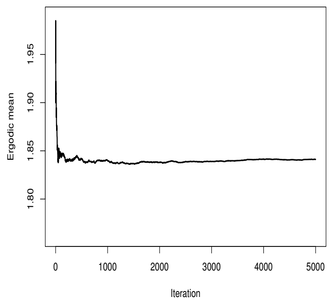
Figure A1. *Cont.*



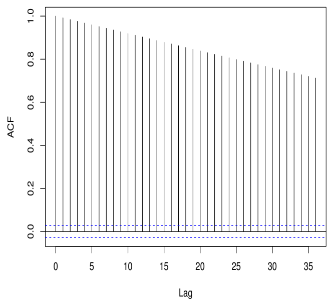
(**d**) A_2.



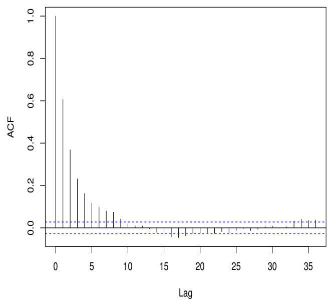
(**e**) A_2.



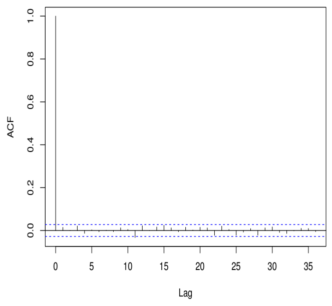
(**f**) A_2.



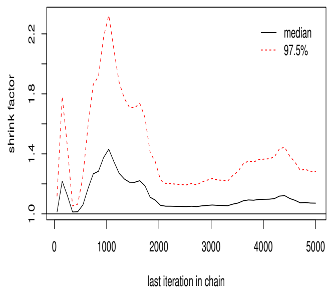
(**g**) A_3.



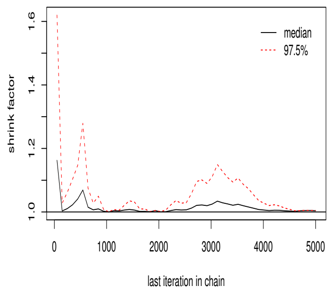
(**h**) A_3.



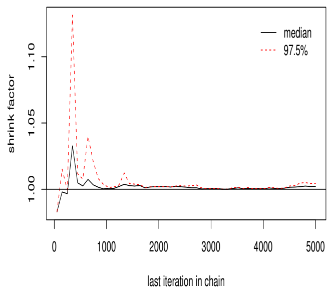
(**i**) A_3.



(**j**) A_3.

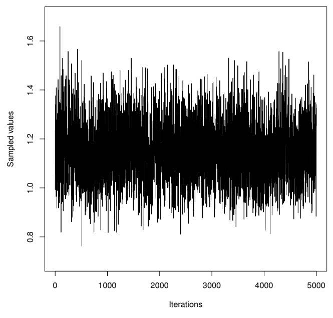


(**k**) A_3.

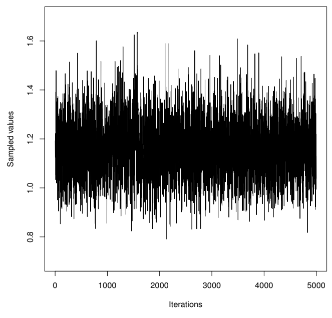


(**l**) A_3.

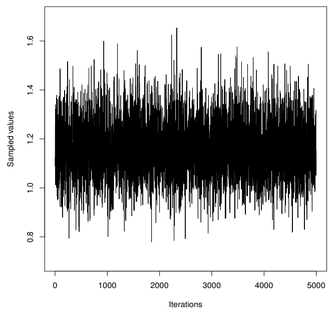
Figure A1. Traceplot, ergodic mean and autocorrelation for sequences produced by algorithms A_1, A_2 and A_3 for α_1.



(**a**) A_1.

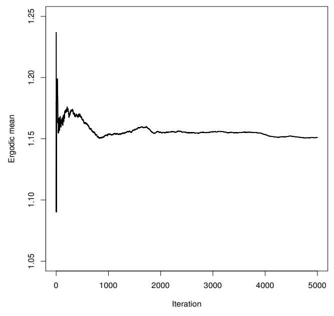


(**b**) A_2.

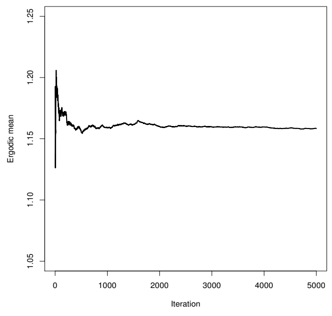


(**c**) A_3.

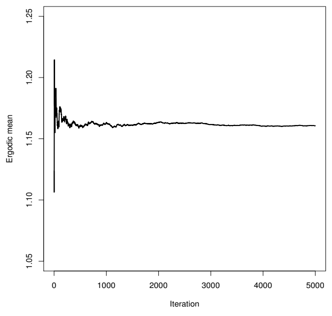
Figure A2. *Cont.*



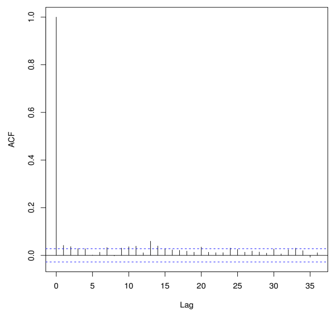
(**d**) A_1.



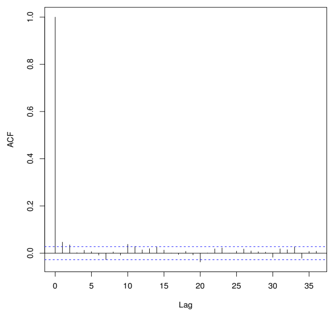
(**e**) A_2.



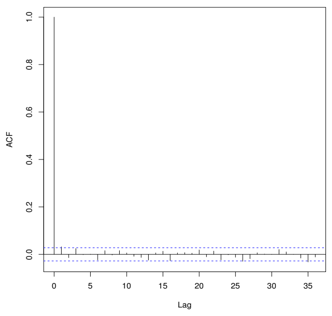
(**f**) A_3.



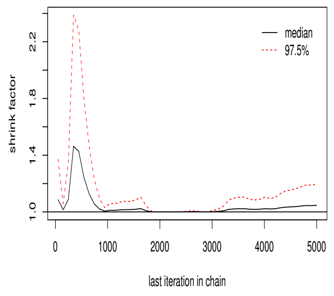
(**g**) A_1.



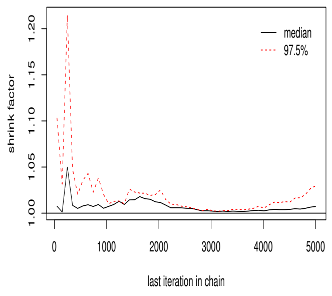
(**h**) A_2.



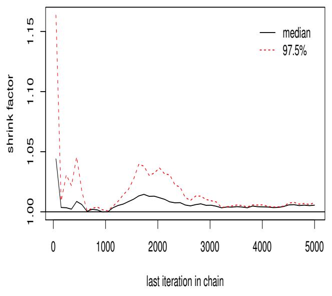
(**i**) A_3.



(**j**) A_1.



(**k**) A_2.



(**l**) A_3.

Figure A2. Traceplot, ergodic mean and autocorrelation for sequences produced by algorithms A_1, A_2 and A_3 for β_1.

References

1. Sahu, S.K.; Dey, D.K. A comparison of frailty and other models for bivariate survival data. *Lifetime Data Anal.* **2000**, *6*, 207–228. [CrossRef] [PubMed]
2. Zhang, S.; Zhang, Y.; Chaloner, K.; Stapleton, J.T. A copula model for bivariate hybrid censored survival data with application to the MACS study. *Lifetime Data Anal.* **2010**, *16*, 231–249. [CrossRef] [PubMed]
3. Shih, J.H.; Louis, T.A. Inferences on the association parameter in copula models for bivariate survival data. *Biometrics* **1995**, *51*, 1384–1399. [CrossRef] [PubMed]
4. Othus, M.; Li, Y. A Gaussian copula model for multivariate survival data. *Stat. Biosci.* **2010**, *2*, 154–179. [CrossRef] [PubMed]
5. Nelsen, R.B. *An Introduction to Copulas*; Springer: New York, NY, USA, 2006.
6. Durante, F.; Sempi, C. *Principles of Copula Theory*; CRC/Chapman and Hall: London, UK, 2015.
7. Romeo, J.S.; Tanaka, N.I.; Pedroso-de-Lima, A.C. Bivariate survival modeling: A Bayesian approach based on copulas. *Lifetime Data Anal.* **2006**, *12*, 205–222. [CrossRef] [PubMed]

8. Da Cruz, J.N.; Ortega, E.M.M.; Cordeiro, G.M.; Suzuki, A.K.; Mialhe, F.L. Bivariate odd-log-logistic-Weibull regression model for oral health-related quality of life. *Commun. Stat. Appl. Methods* **2017**, *24*, 271–290. [CrossRef]
9. Louzada, F.; Suzuki, A.K.; Cancho, V.G. The FGM long-term bivariate survival copula model: Modeling, Bayesian estimation, and case influence diagnostics. *Commun. Stat. Theory Methods* **2013**, *42*, 673–691. [CrossRef]
10. Suzuki, A.K.; Louzada, F.; Cancho, V.G. On estimation and influence diagnostics for a bivariate promotion lifetime model based on the FGM copula: A fully Bayesian computation. *TEMA* **2013**, *14*, 441–461. [CrossRef]
11. Romeo, J.S.; Meyer, R.; Gallardo, D.I. Bayesian bivariate survival analysis using the power variance function copula. *Lifetime Data Anal.* **2018**, *24*, 355–383. [CrossRef] [PubMed]
12. Kumar, P. Probability Distributions and Estimation of Ali–Mikhail–Haq Copula. *Appl. Math. Sci.* **2010**, *14*, 657–666.
13. Neal, R.M. Slice sampling. *Ann. Stat.* **2003**, *31*, 705–767. [CrossRef]
14. Kass, R.E.; Carlin, B.P.; Gelman, A.; Neal, R.M. Markov Chain Monte Carlo in Pratice: A Roundtable Discussion. *Am. Statist.* **1998**, *52*, 93–100.
15. The Diabetic Retinopathy Study Research Group. Preliminary report on the effect of photocoagulation therapy. *Am. J. Ophthalmol.* **1976**, *81*, 383–396. [CrossRef]
16. Therneau, T.M. A Package for Survival Analysis in S, Version 2.38. 2015. Available online: https://CRAN.R-project.org/package=survival (accessed on 4 July 2018).
17. R Development Core Team. *R: A Language and Environment for Statistical Computing*; R Foundation for Statistical Computing: Vienna, Austria, 2012; ISBN 3-900051-07-0.
18. Ali, M.M.; Mikhail, N.N.; Haq, M.S. A class of bivariate distributions including the bivariate logistic. *J. Multivar. Anal.* **1978**, *8*, 405–412. [CrossRef]
19. Lawless, J.F. *Statistical Models and Methods for Life Time Data*; John Wiley and Sons: New York, NY, USA, 1974.
20. Weibull, W. A statistical distribution function of wide applicability. *AMSE J. Appl. Mech.* **1951**, *18*, 292–297.
21. Collett, D. *Modelling Survival Data in Medical Research*, 3rd ed.; Chapman and Hall/CRC: Boca Raton, FL, USA, 2015.
22. Hastings, W.K. Monte Carlo sampling methods using Markov Chains and their applications. *Biometrika* **1970**, *57*, 97–109. [CrossRef]
23. Chib, S.; Greenberg, E. Understanding the Metropolis–Hastings algorithm. *Am. Stat.* **1995**, *49*, 327–335.
24. Gelman, A.; Carlin, J.B.; Stern, H.S.; Rubin, D.B. *Bayesian Data Analysis*; Chapman and Hall: London, UK, 1995.
25. Gilks, W.R.; Richardson, S.; Spiegelhalter, D.J. *Markov Chain Monte Carlo in Practice*; Chapman and Hall: London, UK, 1996.
26. Roberts, G.; Gelman, A.; Gilks, W. Weak convergence and optimal scaling of Random Walk Metropolis algorithms. *Ann. Appl. Probab.* **1997**, *7*, 110–120. [CrossRef]
27. Bedard, M. Weak convergence of Metropolis algorithms for non-i.i.d. target distributions. *Ann. Appl. Probab.* **2007**, *17*, 1222–1244. [CrossRef]
28. Mattingly, J.C.; Pillai, N.S.; Stuart, A.M. Diffusion limits of the random walk Metropolis algorithm in high dimensions. *Ann. Appl. Probab.* **2011**, *22*, 881–930. [CrossRef]
29. Gelman, A., Rubin, D.B. Inference from Iterative Simulation using Multiple Sequences. *Stat. Sci.* **1992**, *7*, 457–511. [CrossRef]