Journal of Statistical Software

MMMMMM YYYY, Volume VV, Issue II. *http://www.jstatsoft.org/*

spatsurv: An R Package for Bayesian Inference with Spatial Survival Models

Benjamin M. Taylor and Barry S. Rowlingson

Lancaster University, UK

Abstract

Survival methods are used for the statistical modelling of time-to-event data. Survival data are characterised by a set of complete records, in which the time of the event is known; and a set of censored records, in which the event was known to have occurred in an interval. When survival data are spatially referenced, the spatial variation in survival times may be of scientific interest. In this article, we introduce a new **R** package, **spatsurv**, for inference with spatially referenced survival data. The specific type of model fitted by this package is a parametric proportional hazards model in which the spatially correlated frailties are modelled by a log-Gaussian stochastic process. The package is extensible in that it allows the user to easily create new models for the baseline hazard function and spatial covariance function. The package implements an advanced adaptive Markov chain Monte Carlo algorithm to deliver Bayesian inference with minimal input from the user.

A particular feature of the new package is the ability to handle large datasets via the use of auxiliary frailties on a regular grid and the technique of circulant embedding for fast matrix computations. We demonstrate the new package on a real-life dataset.

Keywords: Spatial Survival, Correlated Frailties, Parametric Proportional Hazards, Log-Gaussian Frailties.

1. Introduction

Statistical methods for the analysis of survival data are not only applicable in the medical context, but also in many other areas of science and engineering. When survival times are spatially-referenced, some evidence of clustering of high or low times might be apparent on a visual inspection of the data. The question naturally arises as to whether these observed

spatial survival patterns can be explained by incorporating appropriate covariates into the model or whether, in order to obtain reliable inferences for model parameters of interest, it is necessary to explicitly model the unexplained spatial variation. This article introduces an R package (R Core Team 2013) for Bayesian inference with spatially referenced survival data.

Methodological developments in the analysis of spatial survival data have included the following. In Henderson, Shimakura, and Gorst (2002), the authors use a proportional hazards (PH) framework based on Gamma frailties and the Breslow estimate of the hazard function (Breslow 1974) to model the survival times of individuals diagnosed with leukaemia in the north west of England. In Li and Ryan (2002), the authors propose the use of log-Gaussian frailties in a semi-parametric framework. In Diva, Dey, and Banerjee (2008), the authors consider both PH and proportional odds (PO) models for the SEER cancer data (Howlader, Noone, Krapcho, Garshell, Neyman, Altekruse, Kosary, Yu, Ruhl, Tatalovich, Cho, Mariotto, Lewis DR, and EJ 2013); the baseline hazard function of their parametric PH model is based on the Weibull survival model. In each of the above cases, the authors used a Markov chain Monte Carlo (MCMC) algorithm to produce samples from a posterior, enabling them to perform Bayesian inference for each class of models. Li and Ryan (2002) also investigated the Laplace approximation as a means of approximate posterior inference.

To our knowledge, there are few R packages for dealing with spatial survival data. One recent addition to CRAN is the package **spBayesSurv** (Zhou and Hanson 2014), which implements functions for fitting several Bayesian survival models including spatial copula linear dependent Dirichlet process mixture models, anova Dirichlet process mixtures and a marginal spatial proportional hazards model. The R package **INLA** (Rue, Martino, and Chopin 2009; Lindgren, Rue, and Lindström 2011) can also be used to fit survival models. Our package **spatsurv** uses parametric models for the baseline hazard function and correlated log-Gaussian frailties to model spatial dependence. We have chosen to implement Markov chain Monte Carlo (MCMC) inferential algorithms because although they are typically slower than approximate methods, based on the Laplace approximation for example, they in principle deliver unbiased, joint inference for all model parameters and are relatively easily extensible to wider model classes with additional hierarchies.

The software package **BayesX** (Belitz, Brezger, Klein, Kneib, Lang, and Umlauf 2013; Adler, Kneib, Lang, Umlauf, and Zeileis 2013) and associated interface to R, **R2BayesX** (Umlauf, Adler, Kneib, Lang, and Zeileis 2015) also fits spatial survival models. **BayesX** comprises a set of tuned C++ functions for the implementation of structured additive regression models (STAR), including a flexible parametric survival model using penalised splines to model the baseline hazard. STAR models offer a flexible parametrisation to the linear predictor, and a particular advantage of the **BayesX** package in the context of the present article is its capability for handling time-varying coefficients, which is not currently implemented in the **spatsurv** package. The package **BayesX** also has facilities for handling large datasets through reduced rank methodology (Kammann and Wand 2003; Hennerfeind, Brezger, and Fahrmeir 2006). As Park and Liang (2012) state, while methods using a low-dimensional approximation to the latent Gaussian process are indeed effective at reducing the computation time, the dimension of the approximation can still be very high for large datasets, meaning these methods will not work well. There are also other issues with low-rank methods including the question of how to choose an appropriate rank, the specification of priors and the physical interpretation of spatial dependence.

In contrast, the package **spatsurv** implements a new methodology for handling large spatial

datasets that (i) delivers $O(n)$ inference for spatial survival models, where n is the number of observations and (ii) simultaneously solves the spatial prediction problem on a fine regular grid at a cost of $O(m \log m)$, where m is the number of prediction points; both of which would incur cubic complexity using the standard method, see Section 4 for an example and (Taylor 2015a) for further details. The package **spatsurv** implements a combination of advanced adaptive MCMC techniques that require little input from the user and Fourier methods for the fast computation of matrix operations. To illustrate the massive reduction in computation time using the proposed method, we estimate that for the real example in Section 4 (6708 observations and 16384 prediction points), the standard $O(n^3)$ method would take approximately five months to analyse, while our method can be run in little over three hours; moreover, the standard method would incur a massive additional cost in order to predict the spatial process at unsampled locations, whereas with our method, these are a by-product of the sampling scheme, and moreover the user has access to an exact sample from their joint posterior density. Our method is full rank and has the advantage that spatial dependence can be specified through a covariance function, which has interpretational benefits. Lastly, our package is written fully in the R programming language, so from the point of view of extensibility, users without specialist knowledge of another programming language will be able to use and modify **spatsurv** code more easily; examples of this are in Appendices A and B. To the authors' knowledge, this article is the first published walk-through analysis of a spatial survival dataset in the R programming language.

The article is organised as follows. In Section 2, we provide a general introduction to survival models, spatial survival models, the form of the likelihood function and Bayesian statistical inference. In Section 3 we give a tour of **spatsurv** functionality including the preliminaries: formatting data, simulation of spatial survival data, plotting, specifying priors and running the MCMC and also post-processing: MCMC diagnostics, plotting the posterior baseline hazard/cumulative hazard, plotting the posterior covariance function, prediction and the computation of Monte Carlo expectations over the joint posterior. The article concludes with a discussion in Section 6, before detailing some of the more technical aspects in the appendices including user-defined covariance functions in Appendix A and user-defined baseline hazard functions in Appendix B.

2. Survival models

In this section we give a brief overview of the analysis of survival data. We will make frequent reference to medical data, but the reader should note that with the appropriate translation of scenarios and terminology, what is to follow is applicable in duration modelling or reliability analysis etc.

Survival analysis (Cox and Oakes 1984; Klein and Moeschberger 2003; Klein, Ibrahim, Scheike, van Houwelingen, and Van Houwelingen 2013) is concerned with the modelling and prediction of time-to-event data: for each individual, we observe the time of an event. The defining characteristic of survival data relates to the concept of censoring: we do not observe the survival time of *all* individuals, rather we might only know that a certain individual was alive the last time they were seen by a doctor, for example. The observed time might therefore correspond to an event (e.g., death or disease progression), or it might correspond to the last time the patient was recorded as alive. In general, the situation is a little more complex than this example, which is known as *right censoring*: sometimes our data are *left censored*,

meaning the the event happened prior to the patient's first examination by the doctor; or *interval censored*, where we know that the event happened somewhere in an interval of time, between two visits to the doctor for example. In this article, we make the common assumption that the censoring process occurs independently of the process governing the survival times.

Let T be a random variable, the occurrence time of an event of interest. The three (interrelated) quantities of main interest in survival analysis are the *density function*, which gives the probability density function of survival times, denoted $f(t)$; the *survival function*, which gives the probability that an event happened after some time t, $S(t) = \mathbb{P}(T > t)$; and the *hazard function*, which is the instantaneous failure rate at some time t conditional on survival up to that time,

$$h(t) = \lim_{\Delta t \to 0} \left\{ \frac{\mathbb{P}(t \leq T \leq t + \Delta t)}{\Delta t \times S(t)} \right\}.$$

Of additional interest is the *cumulative hazard*, $H(t) = \int_0^t h(s)\mathrm{d}s$ and *lifetime distribution* function, $F(t) = 1 - S(t)$. The following relationships hold,

$$S(t) = 1 - \int_0^t f(x)\mathrm{d}x = \exp\{-H(t)\} \qquad \text{and} \qquad h(t) = \frac{f(t)}{S(t)}. \tag{1}$$

2.1. The spatial parametric proportional hazards model

In this section, we introduce the spatial parametric proportional hazards model for which inference is implemented in the package **spatsurv**.

In this article, we will concentrate on the case where the hazard function takes the following form:

$$h(t_i; \psi, Y_i) = \exp\{X_i\beta + Y_i\} h_0(t_i; \omega), \tag{2}$$

where t_i is the observed time for the ith individual, X_i is a vector of covariates for the ith observation and the parameters of this model, $\psi = (\beta, \omega, \eta)$, correspond respectively to the covariate effects, the parameters of the baseline hazard and the parameters of the covariance function of a spatially continuous stationary latent Gaussian field Y, of which Y_i is the value of the field at the location of observation i. We will parameterise the latent field Y in such a way that $\mathbb{E}[\exp(Y)] = 1$. In the case of a Gaussian field whose marginal variance is σ^2, this is easily achieved by setting the mean of Y to be $-\sigma^2/2$. We do this partly to avoid identifiability issues, but also to give a direct and useful interpretation for $\exp(Y)$, as a multiplicative scaling on the hazard function.

Examples of a suitable spatial covariance function might be the Matérn or Exponential models. The latter specifies the covariance between any two objects of distance d apart to be $\sigma^2 \exp\{-d/\phi\}$, where σ^2 is the marginal variance of the field and ϕ is the 'spatial decay' parameter (the larger ϕ, the longer the range of spatial dependence there is in the field).

We now give two examples of baseline hazard functions derived from parametric survival models. The baseline hazard function derived from the *exponential* survival model has form,

$$\begin{aligned} h_0(t; \theta) &= \theta, \qquad \theta > 0 \\ H_0(t; \theta) &= \theta t. \end{aligned}$$

The parameter $\omega = \theta$ is called the rate parameter. The baseline hazard function derived from the *Weibull* survival model has form,

$$\begin{aligned} h_0(t;\alpha,\lambda) &= \alpha\lambda t^{\alpha-1}, \qquad \alpha,\lambda > 0 \\ H_0(t;\alpha,\lambda) &= \lambda t^{\alpha}; \end{aligned}$$

this is the parametrisation used by the package **spatsurv** i.e., $\omega = (\alpha, \lambda)$. Note that the standard R functions `dweibull`, `rweibull` etc., use a different parametrisation.

Let $\xi_i = (\psi, Y_i) = (\beta, \omega, \eta, Y_i)$. The density function, hazard function, survival function and cumulative hazard function all depend on the parameters ξ, hence we use the notation $f(t;\xi_i)$, $h(t;\xi_i)$, $S(t;\xi_i)$ and $H(t;\xi_i)$ to represent these functions respectively. The density function for a parametric PH spatial survival model can be derived using the right-hand expression in Equation 1 and expanding definitions:

$$\begin{aligned} f(t;\xi_i) &= h(t;\xi_i)S(t;\xi_i), \\ &= h(t;\xi_i)\exp\{-H(t;\xi_i)\}, \\ &= \exp\{X_i\beta + Y_i\}h_0(t;\omega)\exp\left\{-\exp\{X_i\beta+Y_i\}\int_0^t h_0(s;\omega)\mathrm{d}s\right\}, \\ &= \exp\{X_i\beta + Y_i\}h_0(t;\omega)\exp\left\{-\exp\{X_i\beta+Y_i\}H_0(t;\omega)\right\}; \end{aligned} \tag{3}$$

this is the density function for the ith individual. It is straightforward to see that,

$$F(t;\xi_i) = 1 - \exp\left\{-\exp\{X_i\beta+Y_i\}H_0(t;\omega)\right\}. \tag{4}$$

2.2. The likelihood for a survival model

Suppose we have n observations and again let t_i be the data from the ith individual, represented as a vector, $t_i = (t_i^{(1)}, t_i^{(2)})$, in the case of an interval censored observation. We use likelihood-based Bayesian inference to describe what information the data provide about the parameters, $\xi = \{\psi, Y_1, \ldots, Y_n\}$, and functions of these parameters. Conditional on $Y_1, \ldots, Y_n$, we will assume that observations are independent. The likelihood in this case splits into contributing components from left, right and interval censored data as well as uncensored data:

$$\pi(\text{data}|\xi) = \prod_{i \text{ uncensored}} f(t_i;\xi_i) \prod_{i \text{ left censored}} F(t_i;\xi_i) \prod_{i \text{ right censored}} S(t_i;\xi_i) \prod_{i \text{ interval censored}} \left[F(t_i^{(2)};\xi_i) - F(t_i^{(1)};\xi_i)\right]. \tag{5}$$

2.3. Inference

The package **spatsurv** uses a Markov chain Monte Carlo algorithm to perform Bayesian inference for the parametric proportional hazards model. The idea is to use MCMC to draw samples from the posterior:

$$\pi(\xi|\text{data}) \propto \pi(\text{data}|\xi)\pi(\xi),$$

whence we can perform Monte-Carlo inference for $\mathbb{E}_{\pi(\xi|\text{data})}[g(\xi)]$, for functions g for which these expectations exist. MCMC provides an extremely flexible and intuitive basis on which

to perform inference, allowing us to easily generate plots showing quantiles of quantities (such as the hazard, baseline hazard, survival function) of interest.

The package **spatsurv** implements an advanced adaptive MCMC algorithm fully described in Taylor (2015a); an abbreviated description can be found in Appendix C. It remains for the user to specify their prior distributions on the parameters ψ: the MCMC algorithm works with a transformed version of Y, for which there already exists a sensible prior, again see Taylor (2015a) for details.

3. Introducing the package spatsurv

In this section we give a tour of the main functions in the package **spatsurv**.

3.1. Formatting data

In encoding survival data, the package **spatsurv** makes use of the `Surv` class of objects from the **survival** package (Therneau 2014; Terry M. Therneau and Patricia M. Grambsch 2000). Specifically, the `type` of data must be `"right"`, `"left"` or `"interval"`, though negative survival times are not allowed. In a `Surv` object of `type="interval"`, we adopt the convention that times in the first column of the object are the event or censoring times for uncensored, right-censored or left-censored observations: the second column is only used for storing the right endpoints of interval-censored observations. An example is given below.

3.2. Simulating data

We first simulate some data from a spatial parametric PH model using MCMC. Note that simulating spatial survival data using MCMC is much faster than analysing a set of spatial survival data; the simulations here took around a minute on a desktop PC.

We begin by creating a number of objects in which to store the various model parameters.

```
R> library("spatsurv")
R> set.seed(11)
R> n <- 300
R> DIST <- weibullHaz()
R> OMEGA <- c(0.5, 2)
R> COVMODEL <- ExponentialCovFct()
R> COVPARS <- c(0.7, 0.1)
```

In the above, `DIST` is the distribution function on which the baseline hazard is based, in this case a Weibull model with parameters `OMEGA`. The object `COVMODEL` is the spatial covariance function with parameters `COVPARS`.

An alternative way of choosing a spatial covariance function is given through the `covmodel` function, which calls `CovarianceFct` from the package **RandomFields** (Schlather, Malinowski, Oesting, Boecker, Strokorb, Engelke, Martini, Menck, Gross, Burmeister, Manitz, Singleton, Pfaff, and R Core Team 2014), e.g., a Matérn covariance function with roughness parameter 1, can be obtained with `covmodel("matern", pars = 1)` in place of `ExponentialCovFct()`.

Note that for covariance functions defined using the function `covmodel`, any additional parameters (in this case the roughness parameter) are not estimated, rather they are fixed (in this case at 1).

The call `weibullHaz()` returns a list inheriting class `"basehazardspec"` which provides functions to evaluate the baseline hazard and its derivatives. Details of the built-in baseline hazard models are given in Table 1

Code	Name	$h(t)$	$H(t)$	Ranges
`exponentialHaz`	Exponential	θ	θt	$\theta > 0$
`weibullHaz`	Weibull	$\alpha\lambda t^{\alpha-1}$	λt^{α}	$\alpha, \lambda > 0$
`gompertzHaz`	Gompertz	$\alpha \exp\{\beta t\}$	$\frac{\alpha}{\beta}\exp\{\beta t\} - \frac{\alpha}{\beta}$	$\alpha, \beta > 0$
`makehamHaz`	Gompertz-Makeham	$\alpha \exp\{\beta t\} + \mu$	$\frac{\alpha}{\beta}\exp\{\beta t\} - \frac{\alpha}{\beta} + \mu t$	$\alpha, \beta, \mu > 0$
`tpowHaz`	Powers of t, given $\alpha_{1:d}$	$\sum_{i=1}^{d} \lambda_i \alpha_i t^{\alpha_i - 1}$	$\sum_{i=1}^{d} \lambda_i t^{\alpha_i}$	$\lambda_{1:d} > 0$
`BsplineHaz`	B-spline	$\sum_{i=1}^{p} \alpha_i B_i^d(t)$	$\sum_{i=1}^{p} \alpha_i \int_0^t B_i^d(s)\mathrm{d}s$	$\alpha_{1:p} > 0$

Table 1: Showing the available built-in hazard models, their parameters and functional form. In the above, B_i^d is the ith B-spline basis function of degree d.

The package **spatsurv** also provides an extensible base on which to build new models for the baseline hazard and spatial covariance functions, see Appendix A and Appendix B for details on how to do this.

The function `simsurv` is used for simulating spatial parametric proportional hazards data. This function requires the user to supply a design matrix, `X`, together with a column vector of covariate effects, `beta`; in conjunction with the chosen hazard and covariance functions and their parameters as defined above. These inputs enable us to construct the hazard, cumulative hazard, density function etc., and hence we can simulate a set of survival times from the PH model again using MCMC. By default, the spatial location of the observations will be the unit square; this behaviour can be changed by specifying the `coords` argument explicitly. The dataset we generate below corresponds to 300 individuals between the age of 5 and 50; the probability of each individual being male (`X[, "sex"] == 1`) is 0.5 and the probability that the individual has cancer (`X[, "cancer"] == 1`) is 1/5.

```
R> dat <- simsurv(X=cbind(age = runif(n, 5, 50),
```

```
+       sex = rbinom(n, 1, 0.5),
+       cancer = rbinom(n, 1, 0.2)),
+     beta = c(0.0296, 0.0261, 0.035),
+     dist = DIST,
+     omega = OMEGA,
+     cov.parameters = COVPARS,
+     cov.model = COVMODEL,
+     mcmc.control = mcmcpars(nits = 110000,
+       burn = 10000,
+       thin = 100))
```

In the call above, the MCMC chain is run for 110000 iterations with a 10000 iteration burnin, retaining every 100th sample, this is set using the `mcmc.control` argument. The function `simsurv` uses an exponentially-distributed independence proposal kernel to produce samples from the target density: the density function for a spatial parametric PH model. The acceptance probability for this scheme is printed at the end of the run, values close to 1 are optimal. Note that at this stage, there is no censoring. Since the candidate survival times have been generated using MCMC, before proceeding, the user should check that the chains have converged and are mixing sufficiently well. The chains are stored in `dat$T`. We recommend using traceplots and the lag 1 autocorrelation to check for mixing and convergence:

```
R> plot(dat$T[, 178], type = "s")
R> plot(apply(dat$T, 2, function(x){acf(x, plot = FALSE)$acf[2]}),
+    xlab = "Subject Index", ylab = "Lag 1 autocorrelation")
```

The first line gives an example trace plot for the 178th individual and the second produces a plot of the lag-1 autocorrelation for each chain; these plots are shown in Figure 1. Strictly the user should check mixing and convergence in all chains, but for large `n`, a practical solution would be to check 10-20 random traceplots and then use the plot of lag-1 autocorrelations to justify the sample from the posterior being reasonable; in the latter, if most of the points plotted are in the range -0.05 to 0.05, then this is a very good sample.

Having checked for good mixing and convergence, we can proceed to extract the simulated survival times (`data$survtimes`), generate some candidate right-censoring times *independently* of the survival times, `censtimes`, and finally using the function `gencens`, produce an object of class `Surv` that can be used in the analysis:

```
R> survtimes <- dat$survtimes
R> censtimes <- rexp(n, 1 / mean(survtimes))
R> survdat <- gencens(survtimes, censtimes)
```

By default, the function `gencens` simply compares each survival time with the candidate censoring time and declares the observation as censored if the censoring time is less than the simulated survival time.

Lastly, we construct an appropriate object to store the spatial survival data: an **sp** object (Pebesma and Bivand 2005; Bivand, Pebesma, and Gomez-Rubio 2013) of class `SpatialPointsDataFrame`.

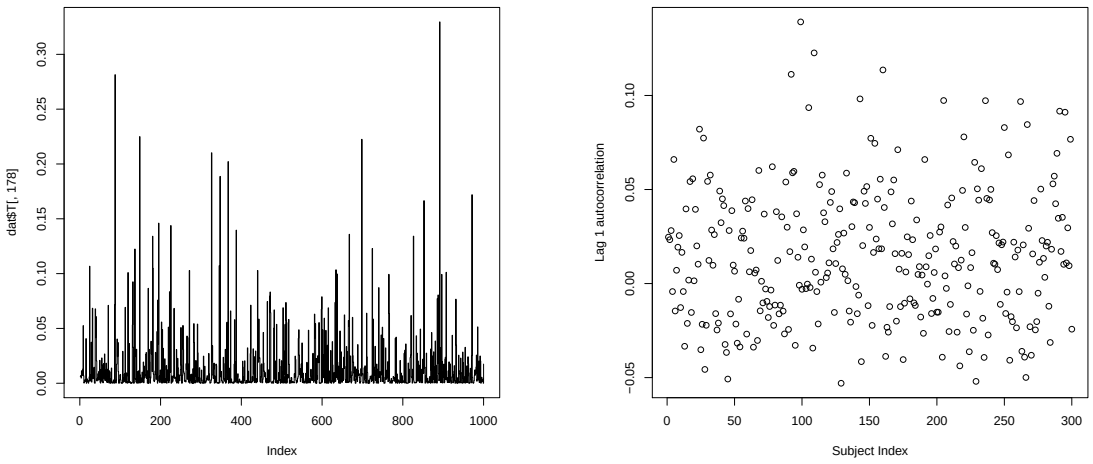

Figure 1: Left: trace plot of simulated survival times for the 178th individual. Right: lag-1 autocorrelation for each of the 300 chains. These plots are used to check mixing and convergence of the Markov chains.

```
R> library("sp")
R> coords <- dat$coords
R> X <- as.data.frame(dat$X)
R> spatdat <- SpatialPointsDataFrame(coords, data = as.data.frame(X))
R> spatdat$ss <- survdat
```

This is the format the main inferential algorithm, `survspat`, will expect data to arrive in.

3.3. Plotting data

The package **spatsurv** includes a basic visualisation tool for spatially referenced survival data.

```
R> plotsurv(spp = spatdat, ss = spatdat$ss, maxcex = 3)
```

The output is shown in Figure 2. The parameter `maxcex` controls how large the plotted points are, acting as a global scale for both the uncensored and censored times, since the size of each dot or plus is proportional to the observed time. The user can control the plotting symbols and colours for the events and the censored observations.

3.4. Specifying priors

Before we can run the main inferential algorithm in the function `survspat`, we must first define prior density functions for the parameters in the model: β, ω and η. As mentioned above, the user does not set a prior for Y; this is because we do not work with Y directly, rather with Γ, where $Y = -\sigma^2/2 + \Sigma_{\sigma,\phi}^{1/2}\Gamma$ and we assume *a priori* that $\Gamma \sim \mathrm{N}(0, \mathbf{1})$, where $\mathbf{1}$ is the identity matrix and $\Sigma_{\sigma,\phi}^{1/2}$ is the Cholesky decomposition of the matrix of the covariance function evaluated at each of the coordinates of the observations.

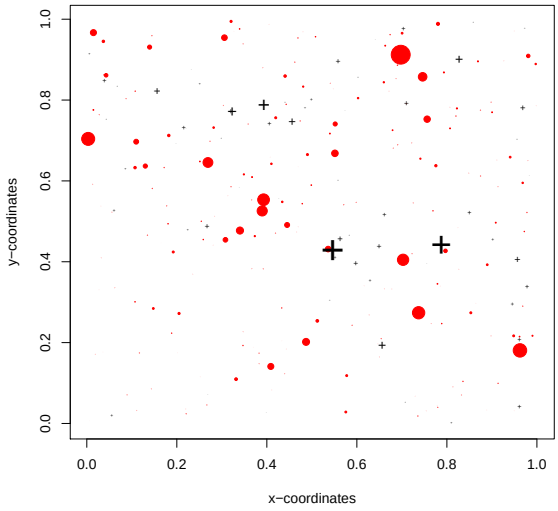

Figure 2: The simulated spatial survival data. Red dots are uncensored events and pluses are censored events. The size of the dot or plus is proportional to the observed time.

We declare priors for β, ω and η as follows:

```
R> betaprior <- betapriorGauss(mean = 0, sd = 10)
R> omegaprior <- omegapriorGauss(mean = 0, sd = 10)
R> etaprior <- etapriorGauss(mean = c(0, -2), sd = 0.5)
```

Each of the functions `betapriorGauss`, `omegapriorGauss` and `etapriorGauss` constructs an independent Gaussian prior for the variable in question (see below). These functions accept vectors or scalars for setting the `mean` and `sd` parameters as they would be processed by the function `dnorm`. In the first line, the code `betapriorGauss(mean = 0, sd = 10)` means that each β parameter will have a mean of zero and standard deviation of 10 independently.

Note that the priors for ω and for η are in fact *priors for the log of these parameters*. This is because σ, ϕ, α and λ are all positive, whereas our MCMC algorithm needs to operate on the whole of the real line. As to which parameters are being sampled on the log (or indeed other) scales, for ω and η, this depends respectively on the choice of baseline hazard distribution and covariance function. The transformation used is specified in the definitions of each of these functions, see Appendix A and Appendix B for further details; in brief all of the parameters in Table 1 are positive and so all of these are in practice sampled on the log-scale.

Note also that although within the MCMC algorithm, ω, η and Y are sampled on a transformed scale, for reporting back to the user at the end of the run, these are back-transformed for ease of interpretation.

Before proceeding to analyse the data, we first wrap the priors in an object of class `mcmcPriors`, constructed as follows,

```
R> priors <- mcmcPriors( betaprior = betaprior,
+    omegaprior = omegaprior,
```

```
+    etaprior = etaprior,
+    call = indepGaussianprior,
+    derivative = derivindepGaussianprior)
```

this can now be passed to the main inferential function, `survspat`.

The function `mcmcPriors` defines independent priors for β, ω and η and the package **spatsurv** only provides functionality for the built-in Gaussian priors discussed above. However, the choice of prior is extensible by the user by creating functions similar to the functions `betapriorGauss`, `omegapriorGauss`, `etapriorGauss`, `indepGaussianprior` and lastly, `derivindepGaussianprior`: the first three of which provide a mechanism for storing and retrieving the parameters of the priors; the fourth, a function for evaluating the log of the prior for a given set of parameter values; and the fifth, a function for evaluating the first and second derivatives of the log of the prior.

3.5. Running the MCMC algorithm

Having chosen a distribution on which the hazard function is to be based and specified our priors, we are in a position to be able to perform Bayesian inference for the model defined by Equation 5. All that remains is to choose the covariates to be included; our model here is `ss ~ age + sex + cancer`, note that unlike `lm` this formula will not include an intercept term. The number of iterations, burnin and thinning parameters are set using the `mcmc.control` argument.

```
R> mod <- survspat(formula = ss ~ age + sex + cancer,
+    data = spatdat,
+    dist = DIST,
+    cov.model = COVMODEL,
+    mcmc.control = mcmcpars(nits = 500000,
+      burn = 10000,
+      thin = 490),
+    priors = priors)
```

This produces an object of class `mcmcspatsurv`, for which there exist various methods, discussed below. Note that this may take several hours to run; the progress bar indicates how far into the burnin or run stage the algorithm is currently at.

3.6. MCMC diagnostics

Before proceeding to interpret any of the model outputs, we must first look for evidence of satisfactory convergence and mixing in the MCMC chain. The chains for β, ω, η and Y are stored in `mod$betasamp`, `mod$omegasamp`, `mod$etasamp` and `mod$Ysamp` respectively

For the parameters β, ω and η, the function `mcmcplot` from the package **mcmcplots** (Curtis 2012) can be used to check trace and autocorrelation plots of the chain:

```
R> require("mcmcplots")
R> mcmcplot(cbind(mod$betasamp, mod$omegasamp, mod$etasamp))
```

This opens up a web browser and displays the diagnostic plots in an easily navigable way, see Figure 3. For parameters sampled on the log scale, it is sometimes preferable to look at plots of the log of that parameter, for example by using `mcmcplot(log(mod$omegasamp))`. Examining the resulting autocorrelation plots shows that the algorithm was producing near independent samples from the posterior with the exception of the parameters σ and ϕ; and these latter two parameters still mixed reasonably well, with the autocorrelation dropping to around zero around the 8th and 10th lag respectively.

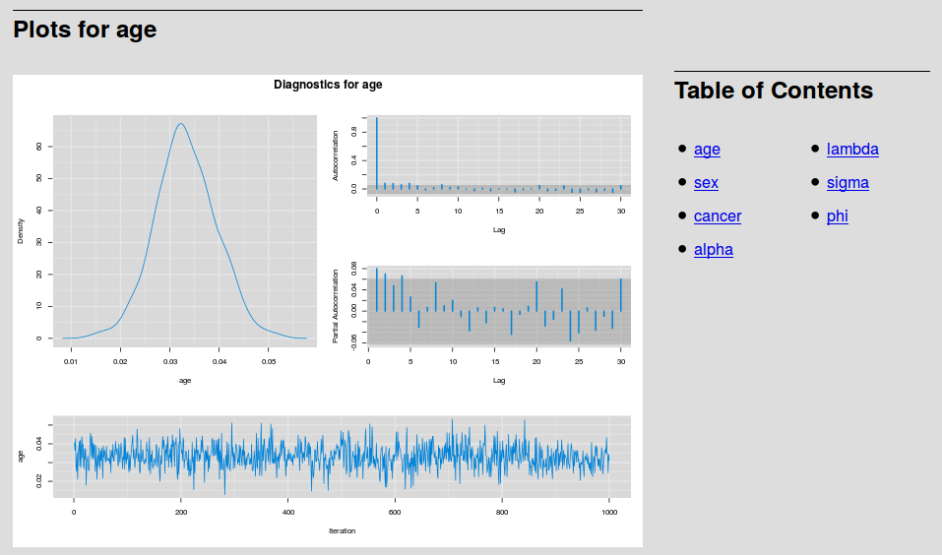

Figure 3: Showing output from `mcmcplot` function from the package **mcmcplots**.

In a similar way to how we conducted diagnostic checks for the simulated data in Section 3.2, we can also look at a plot of the lag-1 autocorrelations for each of the Ys. A further diagnostic check is given by examining a plot of the log-posterior over the saved iterations in the chain, see the second line of code below. Note that this latter plot includes the very first iteration of the chain (so is of length the number of stored iterations plus 1), to provide evidence that the chain has left the transient phase and has reached stationarity.

```
R> frailtylag1(mod)
R> plot(mod$tarrec, ylab = "log-Posterior", type = "s")
```

See Figure 4 for the output of these commands. The left hand plot shows that the lag-1 autocorrelation in the Ys is very low and the right hand plot shows that the MCMC algorithm has found a mode and is exploring it – in this case, the initial value of the log-target appears to be close to the mode of the posterior, more commonly there would be a jump from the initial value, the chain settling at some other value. What is important with this diagnostic plot is that there is no long-term trend evident, which is the case here.

3.7. Post-processing

Now that we have provided evidence that the MCMC chain has satisfactorily converged, we can proceed to producing posterior summaries of the quantities of interest. The package **spatsurv** has a print method for objects of class `mcmcspatsurv`, such as the object `mod`:

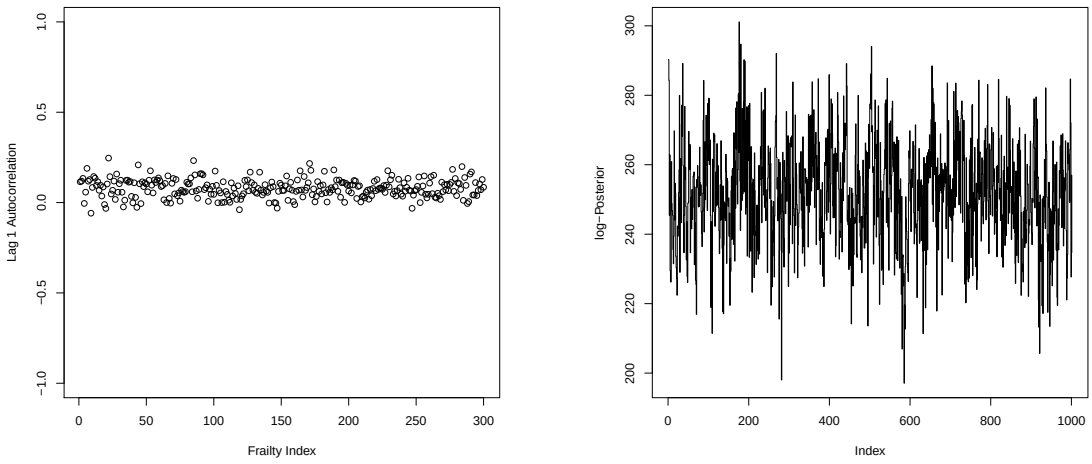

Figure 4: Left: lag-1 autocorrelations in the Y chains. Right: plot of the log-posterior – at this scale, the priors for β and ω appear as flat lines, this is because they uninformative.

```
R> mod

Fixed Effects:
           50%    2.5%  97.5%
age     0.0330  0.0214 0.0456
sex     0.0314 -0.2572 0.3263
cancer -0.0974 -0.4259 0.2470

Baseline Hazard Parameters:
         50%  2.5% 97.5%
alpha  0.532 0.472  0.60
lambda 2.003 1.110  3.76

Spatial Covariance Parameters:
        50%   2.5% 97.5%
sigma 0.661 0.3990 0.992
phi   0.166 0.0734 0.388

Deviance Information Criterion:  -795.6855

MCMC Details:

Number of Iterations 500000
Burnin length         10000
Thining                 490

Running Time:
Time difference of 1.459569 hours
```

there are also methods for the generic functions `summary`, `quantile` and `vcov`. Note that for model selection purposes, the DIC (Spiegelhalter, Best, Carlin, and Van Der Linde 2002) is printed as part of this output and is available as `mod$DIC`.

Plots of the prior and posterior

For inferential purposes, it is of interest to qualify the influence of the choice of prior on the resulting parameter estimates. One way of achieving this is to overlay plots of the prior and posterior densities:

```
R> priorposterior(mod)
```

the result is shown in Figure 5. These plots show the prior density function on its original scale (in this case, on the log-scale for ω and η) as a red line with a histogram of the posterior samples overlaid. For each parameter, we would hope to see a big difference between the prior and posterior, which would indicate that the data are informative about that parameter; in this case, it can be seen that the parameters β and ω and σ are well identified by the data whereas, as is typical in geostatistical inference, the spatial decay parameter, ϕ, is less well identified: see for example Zhang (2004).

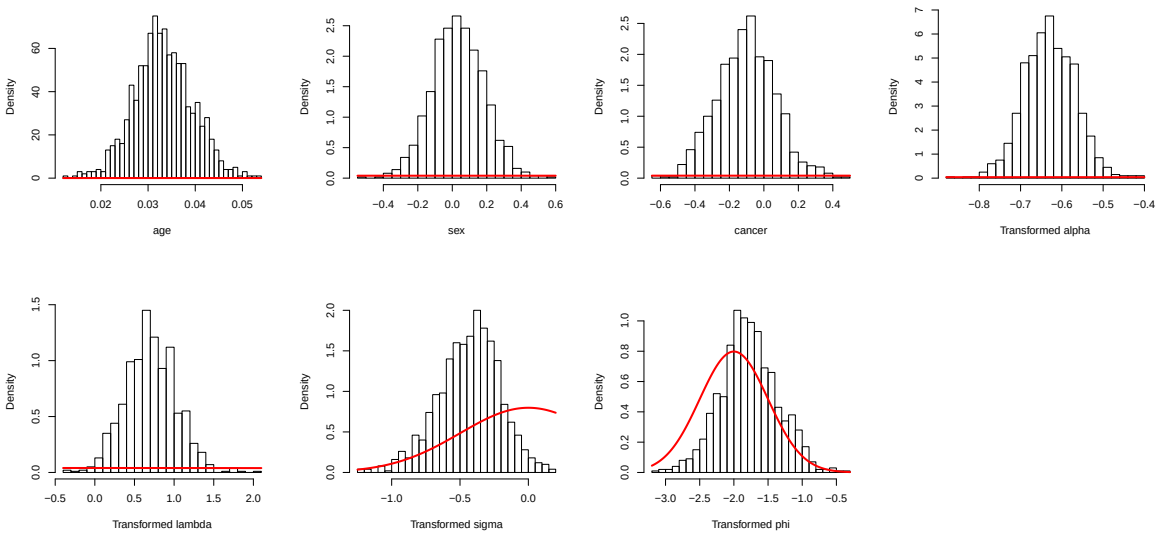

Figure 5: Plot showing the prior (red line) and posterior (histogram) for each parameter.

Posterior baseline hazard and spatial covariance function

We can plot the posterior baseline hazard, h_0, and the posterior spatial covariance function using the following commands:

```
R> baselinehazard(mod)
R> posteriorcov(mod)
```

this produces the output in Figure 6. Note that the posterior cumulative hazard function can be obtained using `baselinehazard(mod,cumulative=TRUE)`. To plot the posterior correlation function, type `posteriorcov(mod,corr=TRUE)`.

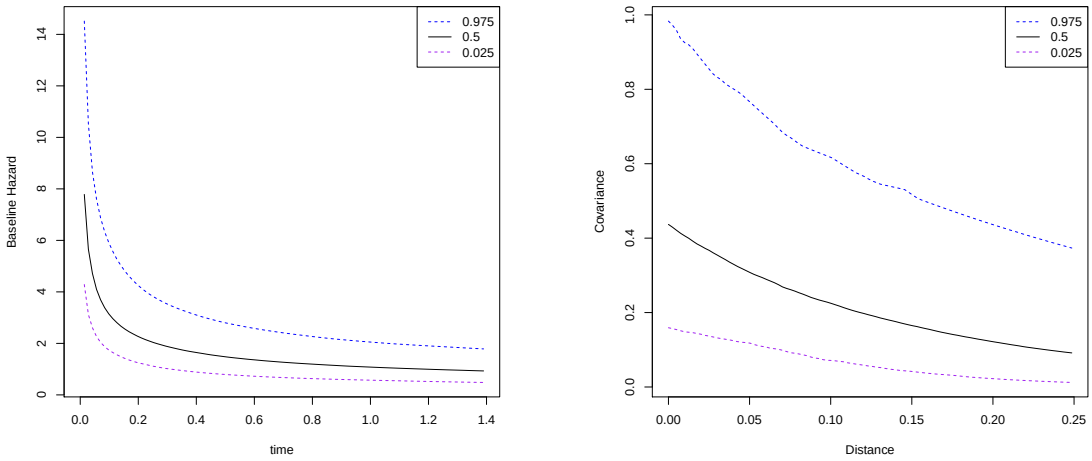

Figure 6: Left: plot showing the posterior median baseline hazard and 95% credible interval. Right: plot showing the posterior median of the spatial covariance function and 95% credible interval.

Prediction

The package **spatsurv** enables the user to extract and examine the hazard, survival and density curves for each individual in the analysis. For example, for individual number 260, we would use the following:

```
R> predict(mod, type = "hazard", indx = 260)
R> predict(mod, type = "survival", indx = 260)
R> predict(mod, type = "density", indx = 260)
```

The result is shown in Figure 7.

The other useful predict `type` options include `"densityquantile"`, which returns the Monte Carlo mean quantiles of the density function for each individual.

```
R> dq <- predict(mod, type = "densityquantile")
```

Note that for some distributions, the quantiles of the density function are not available in an analytically convenient form.

Monte Carlo expectations

Lastly, it is of interest to be able to compute Monte Carlo expectations over the posterior: the package **spatsurv** provides a function `MCE` to facilitate this. Such expectations take the

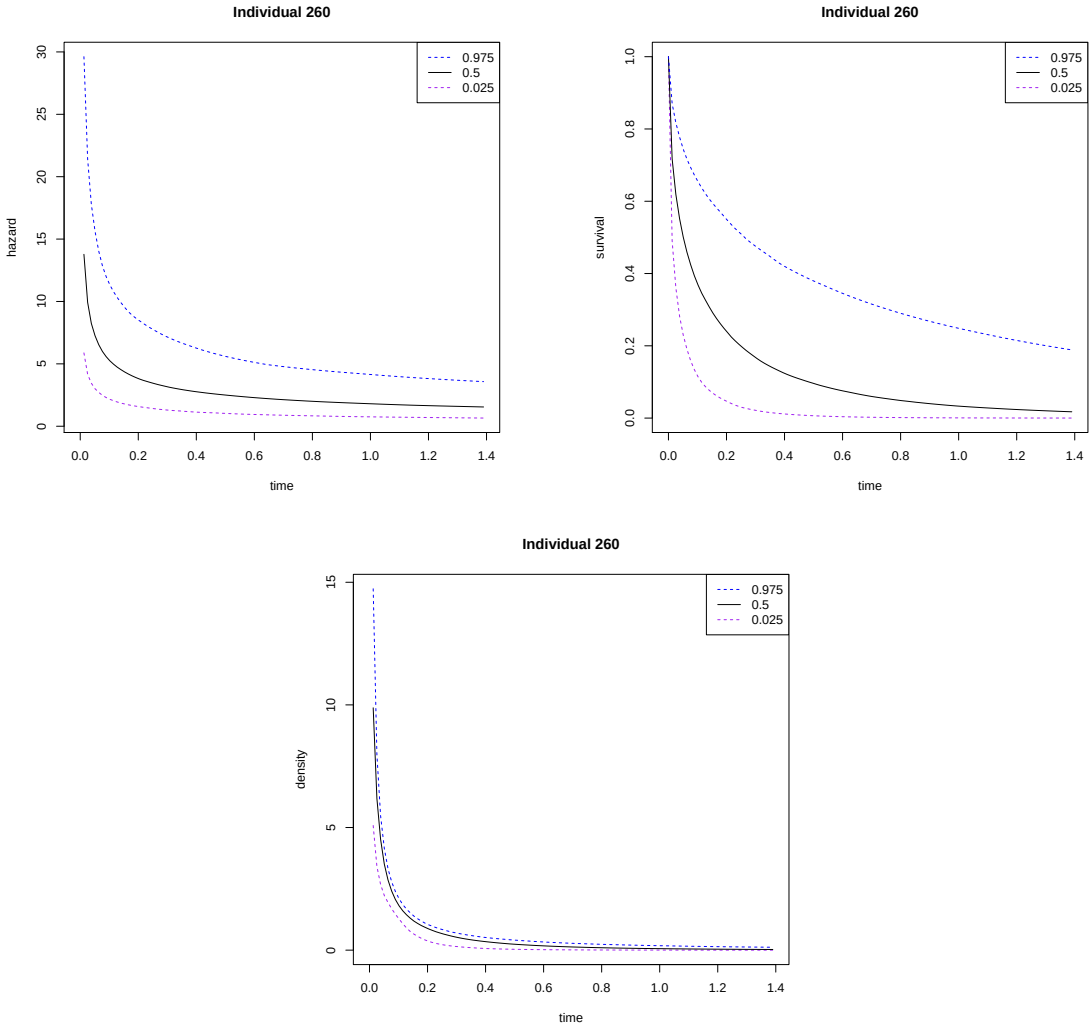

Figure 7: Top left: plot showing the posterior median hazard function and 95% credible interval for individual 260. Top right: plot showing the posterior median survival function and 95% credible interval for individual 260. Bottom: plot showing the posterior median density function and 95% credible interval for individual 260.

general form:

$$\mathbb{E}_{\pi(\beta,\omega,\eta,Y|\text{data})}[g(\beta,\omega,\eta,Y_i)] = \frac{1}{\#\text{ iterations}} \sum_{j=1}^{\#\text{iterations}} g(\beta^{(i)},\omega^{(i)},\eta^{(i)},Y_i^{(j)}),$$

where for example $\beta^{(i)}$ is the ith sample of β drawn from the chain. Note the dependency on Y_i in the function g here. The function `MCE` takes as input the `mcmcspatsurv` object and also a function which has arguments `beta`, `omega`, `eta` and `Y`. An example of such a function is provided by the returned value of the `hazardexceedance` function, detailed below.

Writing the hazard function as,

$$h(t_i;\psi,Y_i) = \exp\{X_i\beta\}\exp\{Y_i\}h_0(t_i;\omega),$$

it can be seen that the frailty for individual i is composed of what we can explain by using covariates, $\exp\{X_i\beta\}$, and what we cannot explain, $\exp\{Y_i\}$, both of which act multiplicatively on the baseline hazard. It is often of scientific interest to quantify the unexplained variation, taking into account the uncertainty in estimation. One way of achieving this is to compute posterior exceedance probabilities, that is $\mathbb{P}[\exp(Y) > c]$ for some threshold c.

```
R> hazardexceedance <- function(threshold, direction = "upper"){
+    fun <- function(beta, omega, eta, Y){
+      EY <- exp(Y)
+      d <- length(Y)
+      len <- length(threshold)
+      A <- matrix(NA, len, d)
+
+      for(i in 1:len){
+        if(direction == "upper"){
+          A[i, ] <- as.numeric(EY > threshold[i])
+        }
+        else{
+          A[i, ] <- as.numeric(EY < threshold[i])
+        }
+      }
+      return(A)
+    }
+    attr(fun, "threshold") <- threshold
+    attr(fun, "direction") <- direction
+    return(fun)
+ }
```

The `hazardexceedance` function, shown above returns an R function taking the appropriate arguments, which can be passed to the `MCE` function, which will compute the Monte Carlo expectation required:

```
R> func <- hazardexceedance(c(1.5, 2, 3))
R> mchaz <- MCE(mod, func)
R> mchaz[, 1:10]
```

```
      [,1]  [,2]  [,3]  [,4]  [,5] [,6]  [,7]  [,8]  [,9] [,10]
[1,] 0.028 0.030 0.016 0.197 0.018    0 0.025 0.363 0.281 0.099
[2,] 0.005 0.004 0.003 0.090 0.002    0 0.007 0.171 0.131 0.033
[3,] 0.000 0.000 0.000 0.018 0.000    0 0.000 0.029 0.042 0.005
```

For example, for the 3rd individual $\mathbb{P}[\exp(Y) > 1.5] = 0.016$, and for the 9th individual $\mathbb{P}[\exp(Y) > 3] = 0.042$. It may be of scientific interest to further investigate individuals for which $\mathbb{P}[\exp(Y) > 1.5] \geq 0.8$, for instance.

4. A large dataset example: fire service response times

The purpose of the previous example was to illustrate basic functionality of the **spatsurv** package. In this section we present an analysis of a large dataset: the London Fire Brigade (LFB) response time data, explored more fully in Taylor (2015b). The data analysed in this section is available from the London Datastore (London Fire and Emergency Planning Authority 2015). Here we present an analysis of the data from 2009. We restrict our attention to fires occurring in dwellings, as most deaths occur in this type of property. Our response variable is the time for the first fire engine to arrive on the scene of a fire at a dwelling and we exclude observations where this response time was not recorded. There is no censoring in the resulting dataset. This is a large dataset containing 6708 observations and a spatial analysis using the model and methods of the previous section would take prohibitively long.

The package **spatsurv** includes methods for computationally efficient inference with large datasets, described in detail in Taylor (2015a). Our model is a slight modification of Equation 2 with respect to the frailties Y_i. The package **spatsurv** implements a shared frailty model on an auxiliary grid of cells on which we wish to predict the response times. The model for the hazard takes the form:

$$h(t_i; \psi, Y) = h_0(t_i; \omega) \exp\{X_i\beta + Y_{\mathcal{G}[i]}\}, \tag{6}$$

where $Y_{\mathcal{G}[i]}$ denotes the value of the process Y in the cell containing observation i. The approximation of the spatially continuous process Y is via a piecewise constant process on a fine grid. The main reason for using this model is computational efficiency: the model in Equation 2 incurs $O(n^3)$ computational cost, where n is the number of observations, whereas the model in Equation 6 is $O(n)$. Moreover since the grid is regular then, provided we assume that Y is a stationary Gaussian process, we can use the technique of circulant embedding to optimise the speed of matrix computations: delivering matrix inversion at a cost of $O(m \log m)$ where m is the number of prediction grid cells. The main advantages of this model are that they allow us to use standard covariance models (such as the exponential or Matérn models) for the spatial dependence between points and spatial prediction of the process Y is a by-product of the Monte Carlo sampling procedure.

We begin by reading in the data, which is stored as a `SpatialPointsDataFrame`.

```
R> library("spatsurv")
R> library("spatstat")
R> library("sp")
R> library("survival")
```

```
R> library("rgeos")
R> library("rgdal")
R> library("leaflet")
R> library("fields")
R> library("raster")
R> library("lubridate")

R> data("fstimes")
```

Our model will include harmonic regression terms to describe the response time as a function of the time of day. Here we set up the required sine and cosine terms in our data frame:

```
R> for(i in 1:4){
R>   fstimes@data[paste("s", i, sep="")] <-
+      sin(i * 2 * pi * fstimes$timenumeric / 24)
R>   fstimes@data[paste("c", i, sep="")] <-
+      cos(i * 2 * pi * fstimes$timenumeric / 24)
R> }
```

We might expect the response time to a call to the fire service to depend on the proximity of the nearest fire station to the location of the fire. However, it is possible the engines from the nearest station are already engaged in responding to another call. In this analysis we therefore use (scaled) fire station intensity as measure of proximity of fire stations to fire locations.

```
R> data("fs")
R> fs <- fs[fs$Description=="Fire Station", ]
R> fscoords <- cbind(fs$Easting,fs$Northing)
R> chull <- convexhull.xy(rbind(coordinates(fstimes), fscoords))
R> win <- expandwinPerfect(chull, TRUE, 1.2)
R> fsppp <- ppp(x=fscoords[, 1], fscoords[, 2], window=win)
R> fsintens <- density.ppp(fsppp)
R> fsintens <- raster(fsintens)
R> proj4string(fsintens) <- CRS("+init=epsg:27700")
R> fstimes$fsintens <- raster::extract(fsintens, fstimes) * 100000000
```

We next set up the model and priors:

```
R> FORM <- S ~ fsintens + s1 + c1 + s2 + c2 + s3 + c3 + s4 + c4
R> betaprior <- betapriorGauss(mean = 0, sd = 100)
R> omegaprior <- omegapriorGauss(mean = 0, sd = 10)
R> etaprior <- etapriorGauss(mean = log(c(1, 1000)), sd = 0.5)
R> priors <- mcmcPriors(   betaprior = betaprior,
+    omegaprior = omegaprior,
+    etaprior = etaprior,
+    call = indepGaussianprior,
+    derivative = derivindepGaussianprior)
```

Lastly, we call the MCMC algorithm to produce samples from the posterior. Here we use the `BsplineHaz` function to model the baseline hazard with a cubic basis spline and the `inference.control` function to specify `gridded=TRUE` which selects the model in Equation 6; the `cellwidth` option is used to control the height and width of the square grid cells. The package **spatsurv** will by default choose the optimum number of computational cells for a given cell width subject to the restriction that the dimension of the grid is a power of 2 in each direction. Smaller choices of the `cellwidth` parameter obviously lead to finer grids and hence a larger computation time, but exactly how much time an MCMC sampler will take to run will depend on the power of the PC in question. We anticipate that to a great extent, the cell width employed in practice will depend on the available computational resource (seeking to make the grid as fine as possible given time constraints). There is an optimum cell width, which minimises the size of the computational grid and hence the computational wastage; the function `getOptCellwidth` computes this optimum for a given initial starting value and the function `showGrid` can be used to visualise the computational grid for a given cell width. The optimal cell width computed with `getOptCellwidth` can be passed to the function `survspat`; if entering manually, this optimal cell width should be rounded upwards.

```
R> mod <- survspat( formula = FORM,
+    data = fstimes,
+    dist = BsplineHaz(fstimes$S[, 1]),
+    cov.model = ExponentialCovFct(),
+    mcmc.control = mcmcpars(nits = 500000,
+      burn = 10000,
+      thin = 490),
+    priors = priors,
+    control = inference.control(gridded = TRUE, cellwidth = 1000))
```

Computation time for this run, performed on a 3.40GHz Intel®Core™i7-4770 desktop computer, was 3.1 hours; the $O(n^3)$ method of the previous section would have taken around five months to run for this dataset and model. Before proceeding to summarise the results, we checked for convergence using traceplots and a plot of the log-posterior. Having established convergence and satisfactory mixing, we can then summarise the results from our model, we discuss here the production of two plots that are of interest in this example.

We first show how to produce a plot of the effect of time of day on the relative risk. Our model includes harmonic regression terms, so we first construct a vector of times of length 100, `tseq`, at which we want to compute the predicted relative risk. The next step is to create a function, `getpred`, to reproduce the harmonic functional at these time points for a given vector of parameters, representing the sampled harmonic coefficients. Finally, applying this function to the appropriate subset of retained β samples yields a matrix, `timetrend`, of dimension 100×1000 from which we can extract quantiles of the functionals evaluated at each time point and plot them. The result is Figure 8. Because a longer 'survival time' in our model is regarded as bad (the engine did not arrive quickly), the intervals of time where the confidence band for the relative risk is below 1 are where the delay in response time is significantly lower than average.

```
R> tseq <- seq(0, 24, length.out = 100)
R> getpred <- function(par){
```

```
+    mat <- rbind( sin(2 * pi * tseq / 24),
+      cos(2 * pi * tseq / 24),
+      sin(2 * pi * tseq * 2 / 24),
+      cos(2 * pi * tseq * 2 / 24),
+      sin(2 * pi * tseq * 3 / 24),
+      cos(2 * pi * tseq * 3 / 24),
+      sin(2 * pi * tseq * 4 / 24),
+      cos(2 * pi * tseq * 4 / 24))
+    return(exp(colSums(par*mat)))
R> }
R> timetrend <- apply(mod$betasamp[, -1], 1, getpred)
R> qts <- t(apply(timetrend, 1, quantile, probs=c(0.025, 0.5, 0.975)))
R> matplot(tseq, qts, type="l", col = c("purple", "black", "blue"),
+    lty = c("dashed", "solid", "dashed"), xlab = "Time of Day",
+    ylab = "Relative Risk")
R> legend("topleft", lty = c("dashed", "solid", "dashed"),
+    col = rev(c("purple", "black", "blue")),
+    legend = rev(c(0.025, 0.5, 0.975)))
```

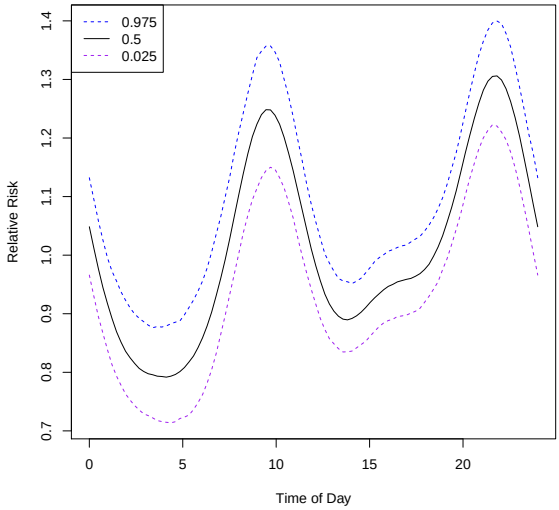

Figure 8: Plot illustrating the functional form of the effect of time of day on the relative risk. Intervals of time where the confidence interval is less than 1 are the times of where engines take significantly longer to respond to calls.

The second plot of interest is a plot of the relative risk over space, computed using the function `hazardexceedance` with option `direction = "lower"` to compute the probability that the relative risk is below the thresholds specified:

```
R> func <- hazardexceedance(c(0.9, 0.8, 0.7), direction = "lower")
R> mchaz <- MCE(mod, func)
```

We can plot these relative risks on a map by creating a `SpatialPolygonsDataFrame` object corresponding to the inferential grid with the exceedance probabilities stored as an entry in the data frame, `exceed`. Alternatively, the results can be stored in a **raster** `brick` or a `SpatialPixelsDataFrame` object, depending on what type of grid is returned by the function `getgrid`. Finally, we can plot the exceedance probabilities using the wrapper function, `spplot1`, to features from the **leaflet** package (Cheng and Xie 2015), see Figure 9. Areas of high probability in this plot are where the relative risk of an engine arriving on the scene of a fire, adjusted for time of day and proximity of fire stations, is such that there may be cause for concern about the provision of services.

```
R> gr <- getGrid(mod)
R> gr$exceed <- mchaz[1, ]
R> m <- spplot1(gr, "exceed", useLeaflet = TRUE,
+    palette=rev(brewer.pal(5,"RdBu")))
R> m
```

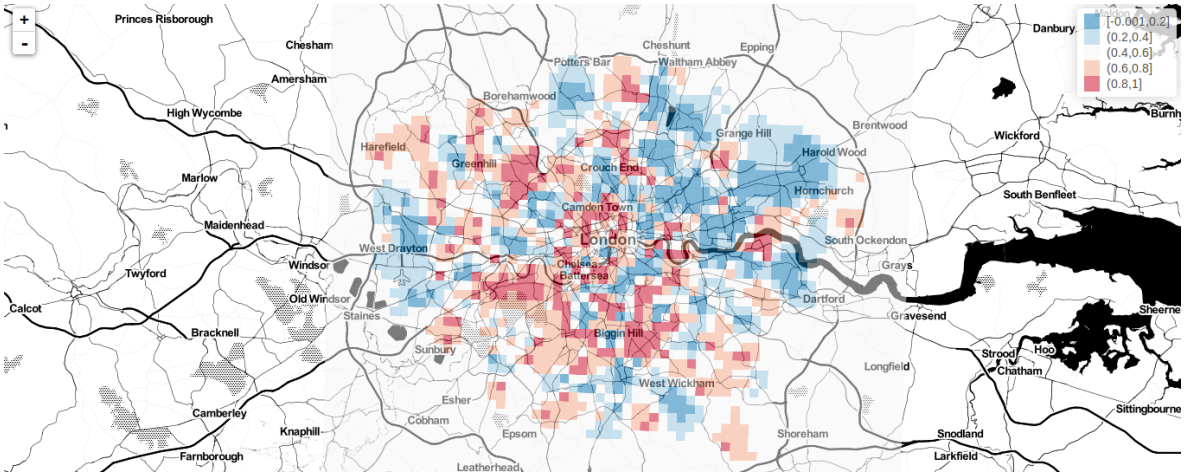

Figure 9: Zoomable leaflet plot of $\mathbb{P}[\exp(Y) < 0.9]$ i.e., the posterior probability that the covariate-adjusted relative risk of an engine arriving on the scene of a fire is less than 0.9.

5. Polygonal data

The package **spatsurv** also handles survival data where the locations of individuals are only known to the regional level, such as the SEER cancer data (Howlader *et al.* 2013). Note that it is not possible to redistribute the SEER data, so we have not included an example analysis in the present article, though this would largely proceed as in the examples above but with the call to `survspat` as detailed below. In this chunk of code, `FORM` is a model formula; `cancerdata` is a `data.frame` containing the covariate and survival data; `DIST` is the assumed distribution for the baseline hazard function; `COVMODEL` is the spatial covariance model; `priors` are the proirs, as set up in the first example above.

```
R> mod <- survspat(formula = FORM,
+    data = cancerdata,
+    dist = DIST,
+    cov.model = COVMODEL,
+    mcmc.control = mcmcpars(nits = 500000,
+      burn = 10000,
+      thin = 490),
+    priors = priors,
+    shape = stateshp,
+    ids = list(shpid = "FIPS", dataid = "STCOUNTY"))
```

The penultimate argument `shape` is assigned an object `stateshp`; which can be either a `spatialPolygonsDataFrame` or a `spatialPointsDataFrame`. In either case, as part of the MCMC algorithm, the command `coordinates(shape)` will be called in order to compute inter-region (or more technically, inter-point) distances. These inter-region distances are later passed to the covariance function, so that in the case of a polygonal `shape`, the covariance between regions is a function of the distance between the centroids of those regions. The reason we have the option of accepting a `SpatialPointsDataFrame` object is so that it is possible to use weighted centroids of the polygons to define the distance between two regions; an example would be population weighted centroids. The package **spatsurv** does not include any functionality for computing weighted centroids, but if they are available they may thus be passed to the inferential algorithm, `survspat`.

The argument `ids` tells the program how to match the case data, `data`, to the `shape` argument; it is assumed that there is a variable in each of these containing the names (or regional identifiers) of the polygons of interest. This argument is a list object with two entries: `shpid` is the name of the variable in the `shape` object with the polygon (or point) names; and `dataid` is the name of the variable in the object `cancerdata` containing the polygon (or point) names at the individual level. In the case of the SEER cancer data (not presented here), `"FIPS"` would be the Federal Information Processing Standard code assigned to polygonal regions labelling counties within states and `"STCOUNTY"` is the corresponding variable in the SEER cancer data at the individual level.

6. Discussion

In this article, we have introduced a new R package, **spatsurv**, for the analysis of spatially referenced survival data using parametric proportional hazards models. Our model for the spatially-correlated frailties has assumed log-Gaussian form and the package is extensible in the sense that it allows the user to construct their own baseline hazard and spatial covariance functions. The package functions are able to cope with right, left or interval censored survival data. For reference purposes, we have provided a statistical introduction to spatial survival models and have demonstrated how to simulate and analyse data using advanced Markov chain Monte Carlo methods. Our package provides simple functions for assessing model convergence and post-processing tools for visualising key quantities of interest. We have also introduced a model and computational method for handling large spatial survival datasets, and demonstrate the performance of this method on a real-life example.

Developing reliable and efficient MCMC code for Bayesian inference with spatial survival

models is a challenging task. We hope that the **spatsurv** package will prove useful to those in the statistical/epidemiological and other communities who seek an off-the-shelf MCMC algorithm they can straightforwardly apply to their data.

Acknowledgements

Map tiles by Stamen Design, under CC BY 3.0. Data by OpenStreetMap, under ODbL.

The London Fire Brigade data in this article are subject to the Open Government License: http://www.nationalarchives.gov.uk/doc/open-government-licence/.

References

Adler D, Kneib T, Lang S, Umlauf N, Zeileis A (2013). "**BayesXsrc**: R Package Distribution of the **BayesX** C++ Sources." R package version 2.1-2, http://CRAN.R-project.org/package=BayesXsrc.

Andrieu C, Thoms J (2008). "A tutorial on adaptive MCMC." *Statistics and Computing*, **18**(4), 343–373.

Belitz C, Brezger A, Klein N, Kneib T, Lang S, Umlauf N (2013). "**BayesX**: Software for Bayesian Inference in Structured Additive Regression Models." Version 2.1, http://www.BayesX.org/.

Bivand RS, Pebesma E, Gomez-Rubio V (2013). *Applied Spatial Data Analysis with R, Second Edition*. Springer-Verlag, NY. URL http://www.asdar-book.org/.

Breslow N (1974). "Covariance Analysis of Censored Survival Data." *Biometrics*, **30**(1), pp. 89–99.

Cheng J, Xie Y (2015). ***leaflet**: Create Interactive Web Maps with the JavaScript **LeafLet** Library*. R package version 0.0.16, URL https://github.com/rstudio/leaflet.

Cox D, Oakes D (1984). *Analysis of Survival Data*. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. Taylor & Francis. ISBN 9780412244902.

Curtis SM (2012). ***mcmcplots**: Create Plots from MCMC Output*. R package version 0.4.1, URL http://CRAN.R-project.org/package=mcmcplots.

Diva U, Dey DK, Banerjee S (2008). "Parametric Models for Spatially Correlated Survival Data for Individuals with Multiple Cancers." *Statistics in Medicine*, **27**(12), 2127–44.

Henderson R, Shimakura S, Gorst D (2002). "Modeling Spatial Variation in Leukemia Survival Data." *Journal of the American Statistical Association*, **97**, 965–972.

Hennerfeind A, Brezger A, Fahrmeir L (2006). "Geoadditive Survival Models." *Journal of the American Statistical Association*, **101**(475), 1065–1075.

Howlader N, Noone AM, Krapcho M, Garshell J, Neyman N, Altekruse SF, Kosary CL, Yu M, Ruhl J, Tatalovich Z, Cho H, Mariotto A, Lewis DR CHS, EJ F (2013). "SEER Cancer Statistics Review 1975-2010. http://seer.cancer.gov/csr/1975_2010/."

Kammann EE, Wand MP (2003). "Geoadditive models." *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, **52**(1), 1–18. ISSN 1467-9876. doi:10.1111/1467-9876.00385. URL http://dx.doi.org/10.1111/1467-9876.00385.

Klein J, Ibrahim J, Scheike T, van Houwelingen J, Van Houwelingen H (2013). *Handbook of Survival Analysis*. Chapman and Hall/CRC Handbooks of Modern Statistical Methods Series. Taylor & Francis Group. ISBN 9781466555662.

Klein J, Moeschberger M (2003). *Survival Analysis: Techniques for Censored and Truncated Data*. Statistics for Biology and Health. Springer-Verlag. ISBN 9780387953991.

Li Y, Ryan L (2002). "Modeling Spatial Survival Data Using Semiparametric Frailty Models." *Biometrics*, **58**(2).

Lindgren F, Rue H, Lindström J (2011). "An Explicit Link Between Gaussian Fields and Gaussian Markov Random Fields: The Stochastic Partial Differential Equation Approach." *Journal of the Royal Statistical Society B*, **73**(4), 423–498.

London Fire and Emergency Planning Authority (2015). "London Fire Brigade Incident Records." http://data.london.gov.uk/dataset/london-fire-brigade-incident-records.

Park J, Liang F (2012). "Bayesian Analysis of Geostatistical Models With an Auxiliary Lattice." *Journal of Computational and Graphical Statistics*, **21**(2), 453–475.

Pebesma EJ, Bivand RS (2005). "Classes and Methods for Spatial Data in R." *R News*, **5**(2), 9–13. URL http://CRAN.R-project.org/doc/Rnews/.

R Core Team (2013). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.

Roberts G, Rosenthal J (2001). "Optimal Scaling for Various Metropolis-Hastings Algorithms." *Statistical Science*, **16**(4), 351–367.

Rue H, Martino S, Chopin N (2009). "Approximate Bayesian Inference for Latent Gaussian Models by Using Integrated Nested Laplace Approximations." *Journal of the Royal Statistical Society B*, **71**(2), 319–392.

Schlather M, Malinowski A, Oesting M, Boecker D, Strokorb K, Engelke S, Martini J, Menck P, Gross S, Burmeister K, Manitz J, Singleton R, Pfaff B, R Core Team (2014). ***RandomFields**: Simulation and Analysis of Random Fields*. R package version 3.0.10, URL http://CRAN.R-project.org/package=RandomFields.

Spiegelhalter DJ, Best NG, Carlin BP, Van Der Linde A (2002). "Bayesian Measures of Model Complexity and Fit." *Journal of the Royal Statistical Society B*, **64**(4), 583–639.

Taylor BM (2015a). "Auxiliary Variable Markov Chain Monte Carlo for Spatial Survival and Geostatistical Models." http://arxiv.org/abs/1501.01665.

Taylor BM (2015b). "Spatial Modelling of Emergency Service Response Times." Under review, preprint available from http://arxiv.org/abs/1503.07709.

Taylor BM, Davies TM, Rowlingson BS, Diggle PJ (2013). "**lgcp**: An R Package for Inference with Spatial and Spatio-Temporal Log-Gaussian Cox Processes." *Journal of Statistical Software*, **52**(4), 1–40. URL http://www.jstatsoft.org/v52/i04/.

Terry M Therneau, Patricia M Grambsch (2000). *Modeling Survival Data: Extending the Cox Model*. Springer-Verlag, New York. ISBN 0-387-98784-3.

Therneau TM (2014). *A Package for Survival Analysis in S*. R package version 2.37-7, URL http://CRAN.R-project.org/package=survival.

Umlauf N, Adler D, Kneib T, Lang S, Zeileis A (2015). "Structured Additive Regression Models: An R Interface to **BayesX**." *Journal of Statistical Software*, **63**(21).

Zhang H (2004). "Inconsistent Estimation and Asymptotically Equal Interpolations in Model-Based Geostatistics." *Journal of the American Statistical Association*, **99**(465), 250–261.

Zhou H, Hanson T (2014). ***spBayesSurv**: Bayesian Modeling and Analysis of Spatially Correlated Survival Data*. R package version 1.0.3, URL http://CRAN.R-project.org/package=spBayesSurv.

A. User-defined covariance functions

In order to define a new covariance function for use with **spatsurv**, the user must provide a list inheriting class `c("covmodel","fromUserFunction")`. An example of how to create such a list is shown in the function `ExponentialCovFct` below. The essential elements of this list are: `npar`, an integer specifying the number of parameters for the model (note typically this should be a small number e.g., 2 or 3); `parnames`, names for the parameters – note that at least one of the parameters must be named `sigma` and have the interpretation that σ^2 is the marginal variance of the latent field; `itrans`, a transformation applied to the working parameters returning a vector the same length as the parameters; `trans`, the inverse of the `$itrans` function, this is the scale on which the MCMC algorithm performs sampling (in this case, the log-scale) and again this function must return a vector the same length as the parameters; and `eval`, a function accepting a vector input of distances, `u`, and parameters, `pars`, which evaluates the covariance function for each element of `u`.

```
R> ExponentialCovFct <- function(){
+    ans <- list()
+    ans$npar <- 2
+    ans$parnames <- c("sigma", "phi")
+    ans$itrans <- exp
+    ans$trans <- log
+    ans$eval <- function(u, pars){
+      ans<- pars[1] ^ 2 * exp(-u / pars[2])
+      return(ans)
+    }
```

```
+    class(ans) <- c("covmodel","fromUserFunction")
+    return(ans)
+ }
```

B. User-defined hazard functions

In this section, we give details on how to define new baseline hazard functions to be used with the package **spatsurv**. This involves defining a function that returns a list whose elements are themselves functions; the returned list must inherit the class `"basehazardspec"`. The package **spatsurv** calls these functions at various points during an MCMC run, or while post-processing. The example code below is from the `weibullHaz` function, this particular function has no arguments. We now discuss each of these functions individually.

The `distinfo` function is used to provide basic distribution specific information to other **spatsurv** functions. The user is required to provide the following information in the returned list: `npars`, the number of parameters in this distribution; `parnames`, the names of the parameters; `trans`, the transformation scale on which the priors will be provided; `itrans`, the inverse transformation function that will be applied to the parameters before the hazard, and other functions are evaluated; `jacobian`, the derivative of the inverse transformation function with respect to each of the parameters; and `hessian`, the second derivatives of the inverse transformation function with respect to each of the parameters – note that currently the package **spatsurv** only allows the use of functions where the parameters are transformed independently.

The `basehazard` function is used to evaluate the baseline hazard function for the distribution of interest. It returns a function that accepts as input a vector of times, `t` and returns a vector.

The `gradbasehazard` function is used to evaluate the gradient of the baseline hazard function with respect to the parameters, this typically returns a vector. It returns a function that accepts as input a vector of times, `t`, and returns a matrix.

The `hessbasehazard` function is used to evaluate the Hessian of the baseline hazard function. It returns a function that accepts as input a vector of times, `t` and returns a list of hessian matrices corresponding to each `t`.

The `cumbasehazard` function is used to evaluate the cumulative baseline hazard function for the distribution of interest. It returns a function that accepts as input a vector of times, `t` and returns a vector.

The `gradcumbasehazard` function is used to evaluate the gradient of the cumulative baseline hazard function with respect to the parameters, this typically returns a vector. It returns a function that accepts as input a vector of times, `t`, and returns a matrix.

The `hesscumbasehazard` function is used to evaluate the Hessian of the cumulative baseline hazard function. It returns a function that accepts as input a vector of times, `t` and returns a list of hessian matrices corresponding to each `t`.

The `densityquantile` function is used to return quantiles of the density function. This is *not required for running the MCMC*, merely for use in post-processing with the `predict` function where `type="densityquantile"`. In the case of the Weibull model for the baseline hazard,

it can be shown that the qth quantile is:

$$-\frac{\log(1-q)}{(\lambda \exp\{X_i\beta + Y_i\})^{1/\alpha}}.$$

Note that quantiles of the density function are not always analytically tractable, in which case the `densityquantile` function should not return anything, instead issuing an error, for example, `stop("Quantiles of the density function are not available")`, or something similar.

```
R> weibullHaz <- function(){
+
+    flist <- list()
+
+    flist$distinfo <- function(){
+      retlist <- list()
+      retlist$npars <- 2
+      retlist$parnames <- c("alpha", "lambda")
+      retlist$trans <- log
+      retlist$itrans <- exp
+      retlist$jacobian <- exp
+      retlist$hessian <- list(exp, exp)
+      return(retlist)
+    }
+
+    flist$basehazard <- function(pars){
+      fun <- function(t){
+        return(pars[2] * pars[1] * t ^ (pars[1] - 1))
+      }
+      return(fun)
+    }
+
+    flist$gradbasehazard <- function(pars){
+      fun <- function(t){
+        return(t ^ (pars[1] - 1) *
+          cbind(pars[2] * (1 + pars[1] * log(t)), pars[1]))
+      }
+      return(fun)
+    }
+
+    flist$hessbasehazard <- function(pars){
+      funfun <- function(t, pars){
+        m <- matrix(0, 2, 2)
+        m[1, 2] <- m[2, 1] <- t ^ (pars[1] - 1) *
+          (1 + pars[1] * log(t))
+        m[1, 1] <- pars[2] * t ^ (pars[1] - 1) *
+          log(t) * (2 + pars[1] * log(t))
+        return(m)
```

```
+      }
+
+      fun <- function(t){
+        return(lapply(t, funfun, pars = pars))
+      }
+      return(fun)
+    }
+
+    flist$cumbasehazard <- function(pars){
+      fun <- function(t){
+        return(pars[2] * t ^ (pars[1]))
+      }
+      return(fun)
+    }
+
+    flist$gradcumbasehazard <- function(pars){
+      fun <- function(t){
+        return(t ^ (pars[1]) * cbind(pars[2] * log(t), 1))
+      }
+      return(fun)
+    }
+
+    flist$hesscumbasehazard <- function(pars){
+      funfun <- function(t, pars){
+        m <- matrix(0, 2, 2)
+        other <- log(t) * t ^ pars[1]
+        m[1, 2] <- m[2, 1] <- other
+        m[1, 1] <- pars[2] * other * log(t)
+        return(m)
+      }
+
+      fun <- function(t){
+        return(lapply(t, funfun, pars = pars))
+      }
+      return(fun)
+    }
+
+    flist$densityquantile <- function(pars, other){
+      fun <- function(probs){
+        return((-log(1 - probs) /
+          (pars[2] * other$expXbetaplusY)) ^ (1 / pars[1]))
+      }
+      return(fun)
+    }
+
+    class(flist) <- c("basehazardspec", "list")
+    return(flist)
```

```
+ }
```

C. Details of the MCMC algorithm

In this section we discuss the MCMC algorithm employed in the package **spatsurv**. We do not in usually sample ω and η directly, but because these parameters often only have support on the positive real line or a subset of it, we perform MCMC on a transformed scale, an appropriate transforms might be $\tilde{\eta} = \log \eta$, for instance. The posterior is as follows:

$$\pi(\beta, \tilde{\omega}, \tilde{\eta}, \Gamma|\text{data}) = \frac{\pi(\text{data}|\beta, \tilde{\omega}, \tilde{\eta}, \Gamma)\pi(\beta, \tilde{\omega}, \tilde{\eta}, \Gamma)}{\pi(\text{data})} \propto \pi(\text{data}|\beta, \tilde{\omega}, \tilde{\eta}, \Gamma)\pi(\beta, \tilde{\omega}, \tilde{\eta}, \Gamma);$$

where $\pi(\text{data})$ is the marginal likelihood. The conditional independence properties of this model imply that $\pi(\text{data}|\beta, \tilde{\omega}, \tilde{\eta}, \Gamma) = \pi(\text{data}|\beta, \tilde{\omega}, \Gamma)$.

The MCMC method we use in the package **spatsurv** is a Metropolis-Hastings sampling scheme. Having initialised the chain at, $\{\beta^{(0)}, \tilde{\omega}^{(0)}, \tilde{\eta}^{(0)}, \Gamma^{(0)}\}$, the ith step consists of drawing a candidate $\{\beta^*, \tilde{\omega}^*, \tilde{\eta}^*, \Gamma^*\}$ from a proposal density, q, and accepting it, that is setting $\{\beta^{(i)}, \tilde{\omega}^{(i)}, \tilde{\eta}^{(i)}, Y^{(i)}\} = \{\beta^*, \tilde{\omega}^*, \tilde{\eta}^*, \Gamma^*\}$, with probability

$$\min\left\{1, \frac{\pi(\beta^*, \tilde{\omega}^*, \tilde{\eta}^*, \Gamma^*|\text{data})}{\pi(\beta^{(i-1)}, \tilde{\omega}^{(i-1)}, \tilde{\eta}^{(i-1)}, \Gamma^{(i-1)}|\text{data})} \frac{q(\beta^{(i-1)}, \tilde{\omega}^{(i-1)}, \tilde{\eta}^{(i-1)}, \Gamma^{(i-1)}|\beta^*, \tilde{\omega}^*, \tilde{\eta}^*, \Gamma^*)}{q(\beta^*, \tilde{\omega}^*, \tilde{\eta}^*, \Gamma^*|\beta^{(i-1)}, \tilde{\omega}^{(i-1)}, \tilde{\eta}^{(i-1)}, \Gamma^{(i-1)})}\right\}.$$

Let $\zeta = (\beta, \tilde{\omega}, \tilde{\eta}, \Gamma)$. In the package **spatsurv**, we use the following proposal:

$$q(\zeta^{(i^*)}|\zeta^{(i-1)}) = \text{N}\left[\zeta^{(i^*)}; \mu_{\zeta^{(i-1)}}, h^2\Sigma\right]. \tag{7}$$

where

$$\mu_{\zeta^{(i-1)}} = \begin{pmatrix} (\beta, \tilde{\omega})^{(i-1)} + \frac{h^2 h^2_{\beta,\tilde{\omega}}}{2}\Sigma_{\beta,\tilde{\omega}}\frac{\partial \log\{\pi(\zeta^{(i-1)}|Y)\}}{\partial(\beta,\tilde{\omega})} \\ \tilde{\eta}^{(i-1)} \\ \Gamma^{(i-1)} + \frac{h^2 h^2_{\Gamma}}{2}\Sigma_{\Gamma}\frac{\partial \log\{\pi(\zeta^{(i-1)}|Y)\}}{\partial\Gamma} \end{pmatrix} \text{ and } \Sigma = \begin{pmatrix} h^2_{\beta,\tilde{\omega}}\Sigma_{\beta,\tilde{\omega}} & 0 & 0 \\ 0 & ch^2_{\tilde{\eta}}\Sigma_{\tilde{\eta}} & \\ 0 & 0 & h^2_{\Gamma}\Sigma_{\Gamma} \end{pmatrix} \tag{8}$$

In Equation (8), $\Sigma_{\beta,\tilde{\omega}}$ is an approximation to the negative inverse of the observed information matrix, $\{-\mathbb{E}[\mathcal{I}(\hat{\beta}, \hat{\tilde{\omega}})]\}^{-1}$ conditional on an estimated Γ and $\tilde{\eta}$, and similarly for $\Sigma_{\tilde{\eta}}$ and Σ_{Γ}. Due to the size of Σ_{Γ}, we use a diagonal matrix rather than the full covariance matrix. The numeric constants $h^2_{\beta,\tilde{\omega}}$, $h^2_{\tilde{\eta}}$ and h^2_{γ} are approximately optimal scalings for Gaussian targets explored by Gaussian random walk or MALA proposals (Roberts and Rosenthal 2001) and we set these as $h^2_{\beta,\tilde{\omega}} = 1.65^2/[\dim(\beta)+\dim(\tilde{\omega})]^{1/3}$, $h^2_{\tilde{\eta}} = 2.38^2/\dim(\tilde{\eta})$ and $h^2_{\Gamma} = 1.65^2/\dim(\Gamma)^{1/3}$, where dim is the dimension.

The constant c is present because we adapt the value of h in our chain using a method described in Andrieu and Thoms (2008) (see Taylor, Davies, Rowlingson, and Diggle (2013) for details) to achieve an approximately optimal acceptance rate of 0.574 to suit the Langevin kernels. This acceptance rate is too high for the random walk, so we set $c = 0.4$ which is roughly the ratio of the approximate optimal acceptance rates for random walk and Langevin kernels (0.234/0.574).

In the package **spatsurv**, we use a combination of maximum likelihood estimation and *ad hoc* methods to obtain initial estimates of β, $\tilde{\omega}$, $\tilde{\eta}$ and Γ; conditional on these, the matrices $\Sigma_{\beta,\tilde{\omega}}$ and Σ_{Γ} are obtained from the second derivative of the posterior with respect to those parameters (Taylor 2015a). We initialise the MCMC run from these estimates apart from Γ, which is initialised at $\Gamma = 0$.

D. Simulation study

We performed a simulation study to check that our code was able to estimate model parameters correctly. This study consisted of 16 simulated datasets (indexed by the ID column in Table 2), and we ran the `survspat` function on each dataset three times. In the first 8 datasets the data were simulated assuming an exponential baseline hazard and in the last 8 scenarios we used a Weibull baseline hazard. When it came to analysing the data, we assumed the functional form of the baseline hazard was known (i.e., we assumed an exponential model for datasets 1–8 and a Weibull model for datasets 9–16). We simulated data using an exponential covariance function for the spatial random effects and again assumed this was known when analysing the data. We assumed that survival time depended on two covariates: age and sex the former being treated as a continuous variable and the latter a binary.

The true and estimated parameters from each of the three runs for each of the 16 datasets are shown in Table 2. We ran each MCMC sampler for 1,000,000 iterations with a burnin of 10,000 iterations and retaining every 990th sample. For each dataset, we assumed $\mathrm{N}(0, 10^2)$ priors for β and $\log\omega$; $\mathrm{N}(0, 0.5)$ priors for $\log\sigma$; and $\mathrm{N}(-3, 0.4)$ for $\log\phi$. The results show that with the exception of dataset 12 we obtain very good estimates of parameters; the columns with title 'IN' have a checkmark where the credible interval contains the true parameter value, though note that due to the presence of the spatial random effects in the model, we do not necessarily expect to obtain the true parameters back when we analyse these datasets. For dataset 12, the discrepancy between true and simulated values in some of the estimated parameters is likely due to either this combination of true parameters being 'difficult' to estimate, or the spatial random effects masking the true effects.

			Run 1				Run 2				Run 3			
Parameter	ID	True	2.5%	50%	97.5%	IN	2.5%	50%	97.5%	IN	2.5%	50%	97.5%	IN
age	1	0.03	0.01	0.02	0.03	✓	0.01	0.02	0.03	✓	0.01	0.02	0.04	✓
sex	1	0.03	-0.18	0.09	0.38	✓	-0.2	0.1	0.38	✓	-0.18	0.09	0.38	✓
θ	1	0.25	0.21	0.31	0.49	✓	0.21	0.31	0.47	✓	0.21	0.32	0.49	✓
σ	1	0.5	0.41	0.7	0.98	✓	0.42	0.68	0.97	✓	0.45	0.71	0.98	✓
ϕ	1	0.02	0.02	0.03	0.06	✓	0.02	0.03	0.06	✓	0.02	0.03	0.06	✓
age	2	0.06	0.04	0.05	0.07	✓	0.04	0.05	0.07	✓	0.04	0.05	0.07	✓
sex	2	0.03	-0.33	-0.04	0.26	✓	-0.29	-0.04	0.24	✓	-0.3	-0.03	0.23	✓
θ	2	0.5	0.33	0.48	0.7	✓	0.34	0.48	0.71	✓	0.34	0.48	0.7	✓
σ	2	1	0.52	0.72	0.93		0.51	0.71	0.92		0.52	0.72	0.92	
ϕ	2	0.02	0.02	0.03	0.05	✓	0.02	0.03	0.05	✓	0.02	0.03	0.05	✓
age	3	0.09	0.07	0.09	0.1	✓	0.07	0.09	0.1	✓	0.07	0.09	0.1	✓
sex	3	0.03	-0.14	0.18	0.48	✓	-0.12	0.18	0.48	✓	-0.1	0.2	0.5	✓
θ	3	0.75	0.34	0.56	0.97	✓	0.33	0.56	1.03	✓	0.33	0.56	0.95	✓
σ	3	1.5	1.19	1.39	1.62	✓	1.21	1.39	1.62	✓	1.2	1.4	1.61	✓
ϕ	3	0.02	0.02	0.03	0.04	✓	0.02	0.03	0.04	✓	0.02	0.03	0.04	✓
age	4	0.12	0.1	0.12	0.13	✓	0.11	0.12	0.13	✓	0.11	0.12	0.13	✓
sex	4	0.03	-0.32	0.01	0.36	✓	-0.33	0.01	0.37	✓	-0.31	0.02	0.36	✓
θ	4	1	0.32	0.56	1	✓	0.31	0.55	1.03	✓	0.31	0.56	1.08	✓
σ	4	2	1.56	1.76	1.99		1.56	1.77	1.99		1.59	1.77	2	✓
ϕ	4	0.02	0.02	0.02	0.03	✓	0.02	0.02	0.03	✓	0.02	0.02	0.04	✓
age	5	0.03	0.02	0.03	0.05	✓	0.02	0.03	0.05	✓	0.02	0.03	0.05	✓
sex	5	0.09	-0.37	-0.09	0.19	✓	-0.37	-0.09	0.2	✓	-0.37	-0.1	0.21	✓
θ	5	1.25	0.96	1.47	2.33	✓	0.97	1.48	2.2	✓	1	1.47	2.31	✓
σ	5	0.5	0.48	0.7	0.97	✓	0.45	0.69	0.93	✓	0.43	0.68	0.94	✓
ϕ	5	0.05	0.02	0.04	0.07	✓	0.02	0.04	0.07	✓	0.02	0.04	0.08	✓
age	6	0.06	0.05	0.06	0.07	✓	0.05	0.06	0.07	✓	0.05	0.06	0.07	✓
sex	6	0.09	0.2	0.48	0.75		0.21	0.46	0.78		0.18	0.47	0.77	
θ	6	1.5	0.72	1.11	1.78	✓	0.7	1.14	1.8	✓	0.73	1.14	1.83	✓
σ	6	1	0.76	0.93	1.13	✓	0.76	0.93	1.15	✓	0.74	0.94	1.14	✓
ϕ	6	0.05	0.03	0.04	0.07	✓	0.03	0.04	0.07	✓	0.03	0.04	0.07	✓
age	7	0.09	0.07	0.09	0.1	✓	0.07	0.09	0.1	✓	0.07	0.08	0.1	✓
sex	7	0.09	-0.14	0.17	0.44	✓	-0.14	0.16	0.45	✓	-0.11	0.17	0.47	✓
θ	7	1.75	1.29	2.28	4.19	✓	1.25	2.24	4.2	✓	1.33	2.21	4.11	✓
σ	7	1.5	1.17	1.38	1.63	✓	1.18	1.37	1.61	✓	1.18	1.37	1.58	✓
ϕ	7	0.05	0.03	0.05	0.07	✓	0.03	0.05	0.07	✓	0.03	0.05	0.07	✓
age	8	0.12	0.12	0.13	0.15	✓	0.12	0.13	0.15	✓	0.12	0.13	0.15	✓
sex	8	0.09	-0.36	-0.03	0.32	✓	-0.35	-0.04	0.3	✓	-0.36	-0.03	0.3	✓
θ	8	2	0.67	1.43	3.54	✓	0.69	1.42	3.31	✓	0.64	1.33	3.28	✓
σ	8	2	1.71	1.95	2.23	✓	1.73	1.95	2.24	✓	1.69	1.93	2.19	✓
ϕ	8	0.05	0.04	0.05	0.07	✓	0.04	0.05	0.08	✓	0.04	0.05	0.08	✓
age	9	0.03	0.03	0.04	0.04	✓	0.03	0.04	0.04	✓	0.03	0.04	0.04	✓
sex	9	0.15	-0.1	0.09	0.29	✓	-0.1	0.09	0.28	✓	-0.11	0.09	0.29	✓
α	9	0.25	0.23	0.25	0.27	✓	0.23	0.25	0.27	✓	0.23	0.25	0.27	✓
λ	9	1	0.68	0.89	1.21	✓	0.67	0.91	1.24	✓	0.66	0.9	1.23	✓
σ	9	0.5	0.3	0.47	0.65	✓	0.31	0.48	0.67	✓	0.31	0.47	0.67	✓
ϕ	9	0.1	0.04	0.07	0.12	✓	0.03	0.07	0.12	✓	0.04	0.07	0.13	✓
age	10	0.06	0.05	0.06	0.07	✓	0.05	0.06	0.07	✓	0.05	0.06	0.07	✓
sex	10	0.15	-0.01	0.21	0.44	✓	-0.02	0.21	0.43	✓	-0.01	0.21	0.43	✓
α	10	0.5	0.41	0.45	0.5	✓	0.41	0.45	0.5	✓	0.41	0.45	0.51	✓
λ	10	1	0.54	0.8	1.23	✓	0.55	0.79	1.21	✓	0.54	0.8	1.19	✓
σ	10	1	0.5	0.69	0.94		0.5	0.69	0.95		0.48	0.69	0.93	
ϕ	10	0.1	0.04	0.07	0.12	✓	0.04	0.07	0.12	✓	0.04	0.07	0.12	✓
age	11	0.09	0.07	0.08	0.1	✓	0.07	0.08	0.1	✓	0.07	0.08	0.1	✓
sex	11	0.15	0.09	0.36	0.62	✓	0.11	0.36	0.62	✓	0.09	0.35	0.63	✓
α	11	0.75	0.6	0.68	0.77	✓	0.61	0.68	0.77	✓	0.61	0.68	0.77	✓
λ	11	1	0.49	0.83	1.6	✓	0.48	0.83	1.59	✓	0.48	0.82	1.48	✓
σ	11	1.5	0.88	1.14	1.47		0.9	1.14	1.46		0.89	1.14	1.44	
ϕ	11	0.1	0.04	0.06	0.1	✓	0.04	0.06	0.1	✓	0.04	0.07	0.1	✓
age	12	0.12	0.09	0.11	0.12	✓	0.09	0.11	0.12	✓	0.09	0.11	0.12	✓
sex	12	0.15	0.19	0.44	0.72		0.17	0.46	0.72		0.18	0.45	0.73	
α	12	1	0.78	0.87	0.98		0.78	0.87	0.98		0.77	0.87	0.98	
λ	12	1	0.11	0.24	0.55		0.12	0.24	0.54		0.13	0.24	0.57	
σ	12	2	1.14	1.41	1.76		1.11	1.42	1.79		1.11	1.43	1.78	
ϕ	12	0.1	0.07	0.1	0.14	✓	0.07	0.1	0.15	✓	0.07	0.1	0.15	✓
age	13	0.03	0.02	0.03	0.04	✓	0.02	0.03	0.04	✓	0.02	0.03	0.04	✓
sex	13	0.21	-0.03	0.17	0.38	✓	-0.05	0.17	0.39	✓	-0.05	0.17	0.39	✓
α	13	0.25	0.22	0.25	0.27	✓	0.22	0.24	0.27	✓	0.22	0.25	0.27	✓
λ	13	2	1.43	2.01	3.05	✓	1.38	1.99	3	✓	1.38	1.99	3.26	✓
σ	13	0.5	0.35	0.54	0.75	✓	0.35	0.53	0.73	✓	0.34	0.53	0.75	✓
ϕ	13	0.15	0.04	0.07	0.13		0.04	0.07	0.13		0.04	0.07	0.12	
age	14	0.06	0.04	0.06	0.07	✓	0.05	0.06	0.07	✓	0.05	0.06	0.07	✓
sex	14	0.21	0.01	0.25	0.49	✓	0.01	0.25	0.48	✓	-0.01	0.24	0.5	✓
α	14	0.5	0.45	0.5	0.56	✓	0.45	0.5	0.56	✓	0.45	0.5	0.55	✓
λ	14	2	2.19	3.76	7.2		2.15	3.67	7.3		2.22	3.77	6.62	
σ	14	1	0.8	1.03	1.33	✓	0.78	1.02	1.32	✓	0.79	1.03	1.29	✓
ϕ	14	0.15	0.05	0.08	0.13		0.05	0.08	0.13		0.05	0.07	0.12	
age	15	0.09	0.07	0.08	0.1	✓	0.07	0.08	0.1	✓	0.07	0.08	0.1	✓
sex	15	0.21	-0.16	0.09	0.32	✓	-0.15	0.09	0.33	✓	-0.16	0.08	0.35	✓
α	15	0.75	0.65	0.72	0.8	✓	0.65	0.72	0.8	✓	0.65	0.72	0.8	✓
λ	15	2	0.59	1.08	2.21	✓	0.6	1.08	2.16	✓	0.58	1.08	2.17	✓
σ	15	1.5	0.88	1.12	1.43		0.87	1.12	1.42		0.86	1.12	1.41	
ϕ	15	0.15	0.07	0.1	0.16	✓	0.07	0.1	0.15	✓	0.07	0.1	0.16	✓
age	16	0.12	0.1	0.11	0.13	✓	0.1	0.11	0.13	✓	0.1	0.11	0.13	✓
sex	16	0.21	-0.07	0.2	0.47	✓	-0.07	0.2	0.49	✓	-0.06	0.2	0.46	✓
α	16	1	0.8	0.89	1	✓	0.8	0.89	1	✓	0.8	0.89	0.99	
λ	16	2	0.53	1.35	5.56	✓	0.55	1.42	4.31	✓	0.55	1.31	3.76	✓
σ	16	2	1.34	1.68	2.07	✓	1.33	1.67	2.07	✓	1.35	1.66	2.04	✓
ϕ	16	0.15	0.09	0.13	0.2	✓	0.09	0.13	0.19	✓	0.09	0.13	0.19	✓

Table 2: Results of the simulation study, see text for details.

Affiliation:

Benjamin M. Taylor
Department of Medicine,
Lancaster University,
Lancaster,
LA1 4YF,
UK
E-mail: b.taylor1@lancaster.ac.uk
URL: http://www.lancs.ac.uk/staff/taylorb1/

Barry S. Rowlingson
Department of Medicine,
Lancaster University,
Lancaster,
LA1 4YF,
UK
E-mail: b.rowlingson@lancaster.ac.uk
URL: http://www.research.lancs.ac.uk/portal/en/people/barry-rowlingson

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume VV, Issue II *Submitted:* yyyy-mm-dd
MMMMMM YYYY *Accepted:* yyyy-mm-dd