Journal of Statistical Software

January 2015, Volume 63, Issue 7. *http://www.jstatsoft.org/*

Bayesian Inference and Data Augmentation Schemes for Spatial, Spatiotemporal and Multivariate Log-Gaussian Cox Processes in R

Benjamin M. Taylor
Lancaster University

Tilman M. Davies
University of Otago

Barry S. Rowlingson
Lancaster University

Peter J. Diggle
Lancaster University

Abstract

Log-Gaussian Cox processes are an important class of models for spatial and spatiotemporal point-pattern data. Delivering robust Bayesian inference for this class of models presents a substantial challenge, since Markov chain Monte Carlo (MCMC) algorithms require careful tuning in order to work well. To address this issue, we describe recent advances in MCMC methods for these models and their implementation in the R package **lgcp**. Our suite of R functions provides an extensible framework for inferring covariate effects as well as the parameters of the latent field.

We also present methods for Bayesian inference in two further classes of model based on the log-Gaussian Cox process. The first of these concerns the case where we wish to fit a point process model to data consisting of event-counts aggregated to a set of spatial regions: we demonstrate how this can be achieved using data-augmentation. The second concerns Bayesian inference for a class of marked-point processes specified via a multivariate log-Gaussian Cox process model. For both of these extensions, we give details of their implementation in R.

Keywords: Cox process, R, spatiotemporal point process, multivariate spatial process, Bayesian Inference, MCMC.

1. Introduction

A major goal of epidemiological research is to investigate the effects of environmental exposures on health outcomes. Our underlying premise is that cases of a health outcome arise in

a spatiotemporal continuum through the presence of a population at risk and a combination of environmental and individual characteristics that affect the risk of disease at each location in space and time. It is therefore natural to model both population density and risk as continuous phenomena in time and space whilst recognising, firstly that the available data will be spatially incomplete and/or aggregated as well as susceptible to measurement error, and secondly that even after modelling the effects of all candidate variables, there will often be a residual component of spatiotemporal variation in risk that can only be captured by including in the model one or more latent, spatiotemporal stochastic processes.

In the present article, our focus is on Bayesian inference for a particular class of statistical models that in our opinion offer a flexible and intuitive framework for delivering answers to many scientific questions arising in spatial and spatiotemporal epidemiology: the log-Gaussian Cox process (LGCP). The spatial LGCP was introduced by Møller, Syversveen, and Waagepetersen (1998). Brix and Diggle (2001) and Diggle, Rowlingson, and Su (2005a) extended this class to include spatiotemporal processes. Diggle, Moraga, Rowlingson, and Taylor (2013) discuss extensions of the LGCP methodology encompassing aggregated count data and multivariate data. Open source software delivering the advanced Markov chain Monte Carlo (MCMC) methods required for inference has been a recent development in the form of the package **lgcp** (Taylor, Davies, Rowlingson, and Diggle 2013), but these methods until now have been restricted to spatial and spatiotemporal LGCPs with *known model parameters*.

When the goal of the analysis is spatial prediction, parameter uncertainty typically makes only a small contribution to the overall prediction error, and *ad hoc* methods of parameter estimation may suffice (Brix and Diggle 2001). However, in general it is more satisfactory to account for parameter uncertainty within a single analysis, and important to do so when parameter values are of scientific interest in themselves, for example when estimating the effects of putative environmental exposures on the risk of disease. The Bayesian inferential framework provides an elegant and transparent means of encapsulating this uncertainty and also delivering joint inference on all model parameters including the latent field, the parameters of the latent field and covariate effects. The aim of this article is to present novel open-source software routines for Bayesian analysis of spatial, spatiotemporal, aggregated and multivariate LGCPs, delivering joint inference on all model parameters. For each of these classes, we provide a brief introduction to the statistical model and a walk-through analysis of a relevant dataset. The new methods are implemented in version 1.3 of the package **lgcp**.

As in previous releases, the package **lgcp** makes extensive use of functions developed in the R (R Core Team 2014) community. Specifically, many of the data structures are built around pre-existing structures in the packages **sp** (Pebesma and Bivand 2005; Bivand, Pebesma, and Gómez-Rubio 2013); **spatstat** (Baddeley and Turner 2005); **RandomFields** (Schlather, Malinowski, Menck, Oesting, and Strokorb 2015); and **ncdf** (Pierce 2014). New and significant dependencies include the packages **rgeos** (Bivand and Rundel 2013) and **raster** (Hijmans and van Etten 2013).

To our knowledge, the package **lgcp** is unique as an R package specifically designed for MCMC-based Bayesian inference for log-Gaussian Cox Processes. Approximate Bayesian inference for log-Gaussian Cox processes may also be performed using the popular **INLA** package (Rue, Martino, and Chopin 2009; Lindgren, Rue, and Lindström 2011). The package **INLA** can be used for the modelling of general latent Gaussian processes and the integrated nested Laplace approximation (INLA) methodology has been used for inference with log-Gaussian

Cox processes, see for example Simpson, Illian, Lindgren, Sorbye, and Rue (2011), Illian, Sorbye, and Rue (2012a) and Illian, Sorbye, Rue, and Hendrichsen (2012b). In Brown (2015), the author introduces R packages **geostatsp** and **geostatsinla** to make the interface to functions from the **INLA** package more user-friendly and furthermore provides an example in which a spatial log-Gaussian Cox process is used to model incidences of murder in Toronto.

The main advantage of using integrated nested Laplace approximations for inference with log-Gaussian Cox processes is computational cost, although this issue is not completely clear-cut, see for example Taylor and Diggle (2014). When using the package **INLA**, invoking `strategy = simplified.laplace` in the `control.inla` argument list, delivers the fastest, but least accurate inference. One advantage of the MCMC-based implementation provided in the package **lgcp** over **INLA**-based counterparts is that once stationarity has been reached, MCMC produces unbiased samples from the joint posterior of all model parameters. The MCMC methods in the package **lgcp** furthermore permit the use of any valid covariance function, rather than being restricted to a subset of the Matérn class.

As pointed out in Taylor and Diggle (2014), we believe that both INLA and MCMC are important inferential tools in the analysis of spatial and spatiotemporal data. INLA is especially convenient for model selection, but once this has been reduced to a single model, a long run of a carefully designed MCMC algorithm may be a safer option for performing inference. Lastly, the two approaches are not mutually exclusive: in Haran and Tierney (2012), the authors use a heavy tailed approximation similar to INLA to construct efficient MCMC proposal schemes.

The remainder of this article is structured as follows. In Section 2, we introduce the log-Gaussian Cox process. In Section 3, we provide a brief introduction to Bayesian inference and MCMC for LGCPs. In Section 4, we introduce four LGCP models and for each example, provide a practical R session to be used as guidance on the implemention of our MCMC routines, diagnostics and post-processing. Specifically, in Section 4.1 we discuss the Bayesian analysis of spatial point process data; in Section 4.2 we analyse a dataset consisting of case counts aggregated to regions; in Section 4.3 we perform a Bayesian analysis of a spatiotemporal LGCP; and in Section 4.4 we analyse a multivariate LGCP. The article concludes with a discussion in Section 5.

2. The log-Gaussian Cox process

In this section, we introduce the log-Gaussian Cox process; for simplicity, we focus the discussion on spatial LGCPs. Throughout this article, we let X be the observed data, Z be measured covariates, β be the effect size and Y be the residual process; also let π denote a generic probability density function.

We begin with some definitions. An *intensity process*, $R : S \to [0, \infty)$, is a non-negative valued stochastic process: a function from a non-null measurable set, in this case $S \subset \mathbb{R}^2$, to the non-negative real line, $[0, \infty)$. A locally finite random set $X \subset S$ is known as a *Cox process* directed by an intensity process, R, if the conditional distribution of X given R is a Poisson process with intensity R. This means that for any bounded Borel set B,

$$\text{card}(X \cap S) \sim \text{Poisson}\left(\int_B R(s)\text{d}s\right).$$

If $\{Y(s) : s \in \mathbb{R}^2\}$ is a spatial stochastic process with the property that for any finite collection, $\{s_i \in S, i = 1, \ldots, n\}$, the joint density $\pi[Y(s_1), \ldots, Y(s_n)]$ is multivariate Gaussian, we say

that Y is a *Gaussian process*. A *log-Gaussian Cox process* is a Cox process whose log-intensity is a Gaussian process.

In the remainder of the article, we will assume that computation takes place on a regular fine grid. We *think* of s as belonging to $\mathbb{R}^2$, but use the notation $X(s)$ to denote the number of events in the computational grid cell *containing* s. In this article, we therefore notate the spatial log-Gaussian Cox process as follows:

$$\begin{aligned} X(s) &\sim \text{Poisson}\{R(s)\} \\ R(s) &= C_A \lambda(s) \exp\{Z(s)\beta + Y(s)\}, \end{aligned}$$

where C_A is cell area, $\lambda(s)$ is a known population offset, $Z(s)$ is a vector of covariate values, with associated effects β, Y is a Gaussian process; the notational extension to spatiotemporal and multivariate processes will be obvious. In practice, we aim to make the computational grid as fine as possible, ideally sufficiently fine that each cell count is either zero or one with high probability.

Our model for the covariance matrix of the process Y on the computational grid will come from a parametric family. Typically, these parameters include a variance parameter, σ^2, and a scale parameter, ϕ. We set $\mathsf{E}[Y] = -\sigma^2/2$ which, using the properties of a log-Normal random variable, gives $\mathsf{E}[\exp(Y)] = 1$; this parametrisation also has an advantage in that $\exp(Y)$ can be interpreted as covariate-adjusted relative risk. We will use η to denote the parameters of the covariance function transformed onto an appropriate scale for the MCMC algorithm e.g., in this article we use $\eta = \{\log(\sigma), \log(\phi)\}$.

3. Inference for log-Gaussian Cox processes

In this section, we provide a brief review of two inferential techniques for LGCPs. In Section 3.1, we explain a computationally simple approach to parameter estimation for the parameters of the process Y, the minimum contrast estimator. Then in Section 3.2, we explain how Bayesian inference can be used to deliver inference for all model parameters: β, η and Y.

3.1. Minimum contrast parameter estimation

It is useful to have fast methods that provide provisional estimates of the parameters of the latent Gaussian process, Y. Such an estimate can be used to decide how fine the computational grid must be in order to capture spatial dependence in Y. Møller *et al.* (1998) use the method of *minimum contrast estimation* (also referred to as the *least-squares* approach). Minimum contrast methods involve finding the parameter values which minimise the squared discrepancy between the assumed parametric form of the second-order characteristic of interest (in our context this will represent either the spatial or temporal covariance), and a corresponding nonparametric estimate thereof. For a comprehensive overview of minimum contrast estimation for spatial and spatiotemporal LGCPs, including a suite of simulation studies gauging proximity of minimum contrast parameter estimates to their true values, see Davies and Hazelton (2013).

The parametric functions typically used for spatial minimum contrast are the pair correlation function (PCF) g and Ripley's K function (Ripley 1977). These are convenient because they are theoretically tractable within the LGCP framework, and they also have accessible

nonparametric estimators. The functional forms of both of g and K depend upon the choice of spatial correlation function. Let $J_{\sigma^2,\phi}$ represent either g or K, and let $\hat{J}$ represent the corresponding nonparametric estimate based on the observed data. The general form of the minimum contrast criterion with respect to spatial lags u is

$$\begin{aligned}\mathcal{M}_J(\sigma^2,\phi) &= \int_{u_0}^{u_{\max}} w(u)\big[v\{\hat{J}(u)\} - v\{J_{\sigma^2,\phi}(u)\}\big]^2\,\mathrm{d}u \\ &\approx u_{\text{diff}}^{-1}\sum_{u\in U} w(u)\big[v\{\hat{J}(u)\} - v\{J_{\sigma^2,\phi}(u)\}\big]^2, \end{aligned} \tag{1}$$

where u_0 is the smallest spatial lag to be considered (typically zero, though this must technically be > 0 for evaluation of the nonparametric PCF), $u_{\max}$ is an upper bound on the distances to be considered (typically chosen as some fraction the size of the spatial observation window), $w(u)$ represents an optional set of lag-dependent weights and $v\{\cdot\}$ is an optional transformation to be applied to the quantities of interest. The integral is approximated in practice by summing over a fine, evenly spaced sequence of values $U = \{u_0, u_1, \ldots, u_{\max}\}$ such that u_{diff} is the difference between any two consecutive terms in U. It is worth noting that dependent upon the design of the model under scrutiny, $\hat{J}$ may represent either the homogeneous or inhomogeneous version of the nonparametric estimator, with the fixed heterogeneous intensity in the latter case specified by some external means.

A similar construction is used in Brix and Diggle (2001) and Diggle *et al.* (2005a) for estimation of the scale of temporal dependence (parameter θ). In that setting, the first step is to estimate the spatial parameters by Equation 1 using 'time-averaged' versions of K or g. Then, estimation of θ proceeds using the temporal autocorrelation function of the frequency of observations over time. The spatial parameter estimates are plugged-in to the theoretical formulation of the temporal correlation, expressions for which can be found in Brix and Diggle (2001) (see also the corrections made in Brix and Diggle 2003; Taylor and Diggle 2013).

Minimum contrast methods suffer from the somewhat arbitrary nature in which one must 'calibrate' the criterion via e.g., $u_{\max}$, w, and v, not to mention whether use of g is 'better' than K or *vice versa* in Equation 1. Use of g also requires selection of a smoothing bandwidth for its nonparametric estimation. There have been some efforts in the literature to aid in these decisions, involving both theoretical e.g., (Guan and Sherman 2007) and numerical e.g., (Diggle and Ribeiro 2003) endeavours. Concerns over subjectiveness aside, Davies and Taylor (2014) indicate minimum contrast methods perform well against approximate likelihood methods in terms of practical performance.

3.2. Bayesian inference and the role of MCMC

In this section we introduce Bayesian inference and, using the example of a spatial log-Gaussian Cox process as an illustration, explain the details of our new methods for the practical fitting of models from this class.

In a 'Classical' or 'Frequentist' analysis, statistical inference is usually concerned with making statements about the asymptotic distribution of the maximum likelihood estimates. When this maximisation problem is in some sense intractable, either because the likelihood is not analytic or because the optimisation problem is too hard, an alternative is to use Bayesian methods (Bernardo and Smith 2008).

The two main ingredients for a Bayesian statistical analysis are 1) a statistical model (or

likelihood), which determines the density $\pi(X|\beta,\eta,Y)$; and 2) a set of prior beliefs about the distribution of the parameters, expressed through the probability density $\pi(\beta,\eta,Y)$. By Bayes' Theorem, the product of the prior and likelihood is proportional to the posterior:

$$\pi(\beta,\eta,Y|X) = \frac{\pi(X|\beta,\eta,Y)\pi(\beta,\eta,Y)}{\pi(X)} \propto \pi(X|\beta,\eta,Y)\pi(\beta,\eta,Y);$$

the quantity $\pi(X)$ is called the marginal likelihood. Note that the conditional independence properties of this model imply that $\pi(X|\beta,\eta,Y) = \pi(X|\beta,Y)$.

Bayesian statistical inference is concerned with making probabilistic statements about the posterior, $\pi(\beta,\eta,Y|X)$, however, in almost all non-trivial applications it is not possible to make analytic probability statements about this density function. In order to proceed, we therefore must resort to either 1) making statistical inference from an approximation of the posterior; or 2) sample-based Monte Carlo inference, based on a sample, $\{\beta^{(j)},\eta^{(j)},Y^{(j)}\}_{j=1}^N$, drawn from $\pi(\beta,\eta,Y|X)$. In this article, we refer to the former as 'approximate methods' because inference is not based on the true posterior; and we refer to the latter as 'exact' methods because as $N \to \infty$, any sample-based estimate of a posterior expectation of interest, $\frac{1}{N}\sum_{i=1}^N f(\beta^{(j)},\eta^{(j)},Y^{(j)})$, is an unbiased estimator of the exact quantity, $\mathsf{E}_{\pi(\beta,\eta,Y|X)}[f(\beta,\eta,Y)]$, for any function f where this expectation exists.

Some advantages of the Bayesian approach are: that it provides a transparent framework for inference; secondly, it is flexible and often provides an elegant and practical solution to inference arising from very complex statistical models. Note, in particular, that Bayesian methods make no formal distinction between estimation of β and η and prediction of Y, and in this way naturally incorporate parameter uncertainty into predictive inference. Against this, obtaining the sample $\{\beta^{(j)},\eta^{(j)},Y^{(j)}\}_{j=1}^N \sim \pi(\beta,\eta,Y|X)$ can itself be a major challenge.

Along with Gibbs sampling, the most commonly employed method for generating the sample $\{\beta^{(j)},\eta^{(j)},Y^{(j)}\}_{j=1}^N$ is the Metropolis-Hastings algorithm (Metropolis, Rosenbluth, Rosenbluth, Teller, and Teller 1953; Hastings 1970). The idea is to simulate from a Markov chain whose stationary distribution is the target of interest, namely $\pi(\beta,\eta,Y|X)$. Having initialised the chain at time 0, $\{\beta^{(0)},\eta^{(0)},Y^{(0)}\}$, the ith step of the algorithm involves drawing a candidate $\{\beta^*,\eta^*,Y^*\}$ from a proposal density, $q(\beta^*,\eta^*,Y^*|\beta^{(i-1)},\eta^{(i-1)},Y^{(i-1)})$ and accepting it, i.e., setting $\{\beta^{(i)},\eta^{(i)},Y^{(i)}\} = \{\beta^*,\eta^*,Y^*\}$, with probability

$$\min\left\{1, \frac{\pi(\beta^*,\eta^*,Y^*|X)}{\pi(\beta^{(i-1)},\eta^{(i-1)},Y^{(i-1)}|X)} \frac{q(\beta^{(i-1)},\eta^{(i-1)},Y^{(i-1)}|\beta^*,\eta^*,Y^*)}{q(\beta^*,\eta^*,Y^*|\beta^{(i-1)},\eta^{(i-1)},Y^{(i-1)})}\right\}.$$

The design of q is critical and forms a major stem of academic research in this field, see Gilks, Richardson, and Spiegelhalter (1995) and Gamerman and Lopes (2006) for reviews. The design of q in the package **lgcp**, discussed below, is a mix of random walk and Langevin proposal kernels. Whilst in some sense the random walk is a blind proposal mechanism, the main idea of using a Langevin kernel, known as the Metropolis-adjusted Langevin algorithm (MALA), is to exploit gradient information on the target to help propose moves towards areas of higher posterior probability. An alternative to the MALA for log-Gaussian Cox process is Hamiltonian Monte Carlo (Girolami and Calderhead 2011). Hamiltonian methods have better theoretical mixing properties compared with a MALA, but are more difficult to tune, requiring pilot runs of the algorithm (Neal 2011). It is for this reason that we have chosen to implement a MALA in the package **lgcp**: in practice we can adaptively choose a

tuning parameter to achieve an approximately optimal acceptance rate of 0.574 (Roberts and Rosenthal 2001).

As in previous implementations of a MALA for the LGCP, we work with a transform of Y, namely Γ, where $Y = \Sigma_\eta^{1/2}\Gamma + \mu_\eta$ and a subscript η denotes dependence on the parameters of the latent process, η (Møller *et al.* 1998; Brix and Diggle 2001; Diggle *et al.* 2013). Let $\zeta = \{\beta, \eta, \Gamma\}$. A full Metropolis-adjusted Langevin algorithm for our target would use the following proposal kernel:

$$q(\zeta^{(i^*)}|\zeta^{(i-1)}) = \mathrm{N}\left[\zeta^{(i^*)}; \zeta^{(i-1)} + \frac{h^2}{2}\Sigma_{\mathrm{opt}}\nabla \log\{\pi(\zeta^{(i-1)}|X)\}, h^2\Sigma_{\mathrm{opt}}\right], \tag{2}$$

where h is a scaling constant. The ideal choice for Σ_{opt} would be the inverse of the Fisher information matrix evaluated at the maximum likelihood estimate of ζ (Girolami and Calderhead 2011). Unfortunately, due to the high dimensionality of ζ, it is infeasible to work with this matrix. However, we are at liberty to use an approximation of the optimal choice and it transpires that in doing this we can still obtain an efficient algorithm.

As mentioned above, we use a proposal kernel that is a mix of random walk and MALA components. The proposal variance matrix is block diagonal, where each block is an approximation of the inverse of the Fisher information matrix. We mix random walk and MALA proposals because the gradient of the target with respect to η is in general difficult to compute and also computationally costly. We suggest the following overall proposal:

$$q(\zeta^{(i^*)}|\zeta^{(i-1)}) = \mathrm{N}\left[\zeta^{(i^*)}; \mu_{\zeta^{(i-1)}}, h^2\Sigma\right]. \tag{3}$$

where

$$\mu_{\zeta^{(i-1)}} = \begin{pmatrix} \Gamma^{(i-1)} + \frac{h^2 h_\Gamma^2}{2}\Sigma_\Gamma \frac{\partial \log\{\pi(\zeta^{(i-1)}|Y)\}}{\partial \Gamma} \\ \beta^{(i-1)} + \frac{h^2 h_\beta^2}{2}\Sigma_\beta \frac{\partial \log\{\pi(\zeta^{(i-1)}|Y)\}}{\partial \beta} \\ \eta^{(i-1)} \end{pmatrix} \text{ and } \Sigma = \begin{pmatrix} h_\Gamma^2\Sigma_\Gamma & 0 & 0 \\ 0 & h_\beta^2\Sigma_\beta & 0 \\ 0 & 0 & ch_\eta^2\Sigma_\eta \end{pmatrix} \tag{4}$$

In Equation 4, Σ_Γ is an approximation to the negative inverse of the Fisher information matrix, $\{-\mathsf{E}[\mathcal{I}(\hat{\Gamma})]\}^{-1}$, and similarly for Σ_β and Σ_η. The constants h_Γ^2, h_β^2 and h_η^2 are approximately optimal scalings for Gaussian targets explored by Gaussian random walk or MALA proposals, see Roberts and Rosenthal (2001). We set $h_\Gamma^2 = 1.65^2/\dim(\Gamma)^{1/3}$, $h_\beta^2 = 1.65^2/\dim(\beta)^{1/3}$ and $h_\eta^2 = 2.38^2/\dim(\eta)$, where dim is the dimension. In the package **lgcp**, we construct Σ_Γ, Σ_β and Σ_η based on initial *ad hoc* guesses at Γ, β and η followed by a quadratic approximation to the target. We tune h so that the MCMC algorithm has an average acceptance rate of 0.574, which is approximately optimal for the MALA components. However, in order to have the random walk block running at approximate optimality, we introduce the scaling constant $c = 0.4$ to temper the proposals in the η component; we have observed that this choice works well across a variety of scenarios.

The package **lgcp** allows the user to specify a prior of the form $\pi(\beta, \eta, \Gamma) = \pi(\beta)\pi(\eta)\pi(\Gamma)$. Following Møller *et al.* (1998) and Brix and Diggle (2001), we set the prior for Γ to $\pi(\Gamma) = \mathrm{N}(0, I)$, where I is the identity matrix. This is a sensible prior for Γ, given the relationship with Y, so we do not allow the user to modify this choice. The user *is* however able to specify priors for β and η, as will be demonstrated below. Although a prior of the form $\pi(\Gamma)\pi(\beta, \eta)$

would give the user greater choice, in practical contexts it is difficult to justify an *a priori* covariance structure between β and η.

4. Worked examples

In this section, we provide four worked examples. Each of these examples assumes a different statistical model for the data: in Section 4.1, we use a spatial LGCP to model cases of primary biliary cirrhosis in Newcastle-Upon-Tyne; in Section 4.2, we re-visit the cirrhosis data, but pretend the case counts had been aggregated to regions; in Section 4.3, we use a spatiotemporal LGCP to model cases of gastrointestinal infection in Hampshire; lastly, in Section 4.4, we use a multivariate LGCP to investigate the spatial segregation of four genotypes of bovine tuberculosis in Cornwall.

Due to the computationally demanding nature of some aspects of the model fitting process, we recommend the user follows the procedure detailed in Algorithm 1. Although pilot runs are not strictly necessary for implementing the adaptive MCMC algorithm, it is nevertheless wise to perform some short runs to make sure everything appears to be in order before committing a large chunk of computation time to a long MCMC run for the final analysis. We have found that printing the current value of h to the console is invaluable as an online check in step 5 of Algorithm 1; this is done in all our implementations below. The scaling parameter h tends to converge very quickly, so using this in pilot runs of 1,000–5,000 iterations as a guide to help choose the number of iterations for the final run is well worthwhile.

In our experience of using this MCMC algorithm, we have found that values of h between 0.5 and 1 usually indicate that the chain is mixing very well and 500,000–1,000,000 iterations is usually sufficient to achieve convergence; values between 0.2 and 0.5 will likely need to be run for slightly longer to achieve convergence e.g., 1,000,000–3,000,000 iterations; and values around 0.02–0.05 will likely need a considerable number of iterations to achieve stationarity e.g., 20,000,000 iterations.

4.1. PBC in Newcastle-Upon-Tyne, a spatial point process model

Introduction

In the next two sections we re-visit a point process dataset originally analysed in Prince, Chetwynd, Diggle, Jarner, Metcalf, and James (2001). These data consist of geo-referenced cases of definite or probable primary biliary cirrhosis (PBC) alive between 1987 and 1994. In this section, we treat these data as they were collected: as a spatial point process dataset. Our statistical model is given in Equation 5:

$$\begin{aligned} X(s) &= \text{Poisson}[R(s)], \\ R(s) &= C_A \lambda(s) \exp\{Z(s)\beta + Y(s)\}. \end{aligned} \tag{5}$$

Here $X(s)$ is the number of events in the cell of the computational grid containing s, $R(s)$ is the Poisson rate, C_A is the cell area, $\lambda(s)$ is a known offset, $Z(s)$ is a vector of measured covariates and $Y(s)$ is the latent Gaussian process on the computational grid. The other parameters in the model are β, the covariate effects; and $\eta = \{\log(\sigma), \log(\phi)\}$, the parameters of the process Y on an appropriately transformed (in this case log) scale.

Algorithm 1 Recommended procedure for Bayesian modelling of log-Gaussian Cox processes in **lgcp**

1: We recommend computing approximate values of the parameters, η, of the process Y using *ad hoc* methods. These approximate values are used for two main reasons: (1) to help inform the size of the computational grid, since we will need to use a cell width that enables us to capture the dependence properties of Y and (2) to help inform the proposal kernel for the MCMC algorithm.
2: We recommend that the user choose an appropriate grid on which to perform inference; this will partly be determined by the results of the first stage and partly by the available computational resources available to perform inference.
3: If environmental covariates are used in the analysis, we suggest that the user next constructs an overlay of these data (in the form of `SpatialPixelsDataFrame` or `SpatialPolygonsDataFrame` objects) onto the computational grid that will be used in the subsequent analysis. This can be an expensive step, as **lgcp** employs polygon/polygon overlays to infer covariate values on the grid. We further recommend that the user saves this object after it has been constructed, and in future reference to the data reloads this object, rather than having to re-compute it.
4: Decide on which covariates are to play a part in the analysis and use the **lgcp** functions to interpolate these onto the computational grid. Note that having saved the results from step three, this is a relatively quick operation, and allows the user quickly to construct different design matrices, Z, from different candidate models for the data.
5: Set up the population offset and specify the priors and, if desired, the initial values for the MCMC.
5: Run the MCMC algorithm and save the output to disk. We recommend dumping information to disk because it offers much greater flexibility in terms of MCMC diagnosis and post-processing.
6: Perform post-processing analyses including MCMC diagnostic checks and produce summaries of the posterior expectations we require for presentation.

Analysis of a point-process dataset

We start by loading the **lgcp** package and data for this example.

```
R> library("lgcp")
R> load("sd_liver.RData")
```

The point process data are contained in an object `sd` of **spatstat** class `ppp`; these are a subset of the original data consisting of 415 cases located in the Newcastle-Upon-Tyne area, illustrated in Figure 1; the background of this figure was obtained using the **OpenStreetMap** package (Fellows 2013).

Along with the case data, our covariate information consists of population counts and socio-demographic information in a `SpatialPolygonsDataFrame` object.

```
R> load("popshape_liver.RData")
```

This loads an object called `popshape` consisting of population counts (total and male/female counts) and the 7 individual domains of the Index of Multiple Deprivation (IMD) measured

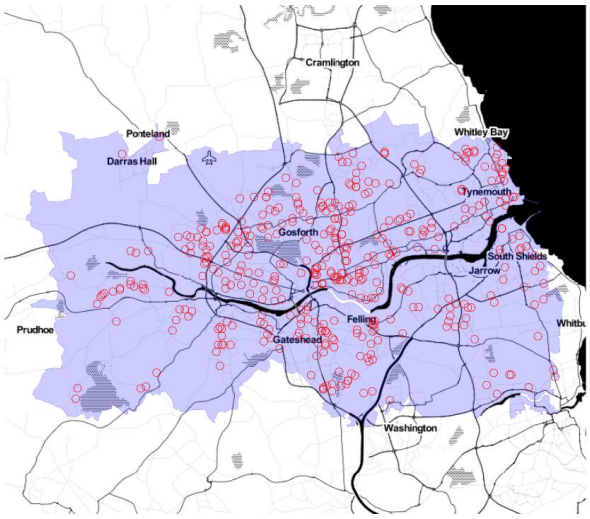

Figure 1: Plot of cases of primary biliary cirrhosis in Newcastle-Upon-Tyne.

at the Lower Super Output Area (LSOA) level. Since this is only an illustrative example of our methodology and R code, we used population data from the 2001 census and IMD data from the 2007 report. We constructed a variable `propmale` equal to the proportion of males in each of the LSOAs.

We now have all the raw information required to fit the model in Equation 5 and begin, according to Algorithm 1, by obtaining approximate estimates of the parameters.

```
R> minimum.contrast(sd, model = "exponential", method = "g",
+    intens = density(sd), transform = log)

$estimates
        scale variance
[1,] 275.6771 1.560087

$discrepancy
     Squared discrepancy
[1,]            299.1335
```

As the approximate spatial scale of dependence, 275 metres, is quite small, this tells us that quite a fine grid *might* be necessary to capture the dependence structure in the process Y. We use the command `chooseCellwidth` interactively to choose an appropriate grid size:

```
R> chooseCellwidth(sd, cwinit = 300)
```

Running this command several times with different values of `cwinit` produces plots of the computational grid overlaid on top of the observation window; the size of the output grid appears in the subtitle. Figure 2 shows two such plots; note that computation grids of size $2^m \times 2^n$ for some positive integers m and n are the most efficient sizes for the Fast Fourier Transform (Wood and Chan 1994; Taylor and Diggle 2014). Both of the plots in this figure show output grids of size 128×64, but the right-hand plot, using a cellwidth of 300, is the

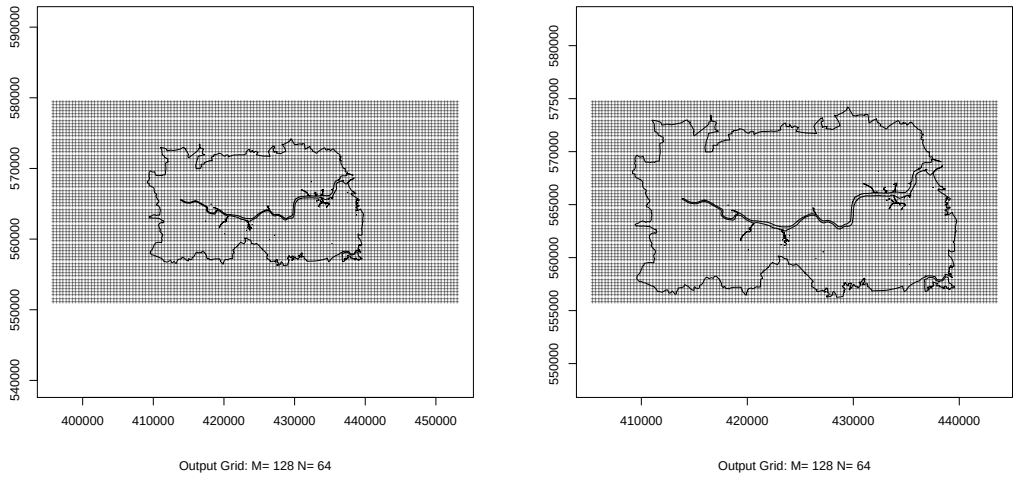

Figure 2: Left: Output of `chooseCellwidth(sd, cwinit = 450)`. Right: Output of `chooseCellwidth(sd , cwinit = 300)`. The right hand plot shows a more efficient use of the computational resources, since there are more cells inside the observation window but the output grid, and hence the computational cost, is the same.

more efficient, since more of the cells in the computational grid fit inside the observation window. We chose a cellwidth of 300m because it just sufficient to capture the dependence properties of Y (if they are assumed *a posteriori* to be the estimated approximate value of 275 metres) and also leads to an efficient computational grid size.

As we will be using environmental covariates in the analysis, the next step in Algorithm 1 is to perform and save the polygon overlay operations. The result of this step is a polygon/polygon overlay of the computational grid onto the `SpatialPolygonsDataFrame` containing the covariate information. In the section of code below, we define objects `CELLWIDTH`, the chosen cell width; and `EXT`, the amount by which the computational grid will be extended in the x and y directions in order to obtain a block circulant matrix see Wood and Chan (1994), Taylor *et al.* (2013) and Davies and Bryant (2013). Typically the parameter `EXT` will be set to 2, unless there is suspected long-range dependence in the process Y (see Appendix E).

```
R> CELLWIDTH <- 300
R> EXT <- 2
R> polyolay <- getpolyol(data = sd, regionalcovariates = popshape,
+    cellwidth = CELLWIDTH, ext = EXT)
```

It would be ideal to use a finely sampled pixel image of population as a Poisson offset. However, since we only have access to population in LSOA, we instead enter the variable `pop` as a covariate (technically we require log-population, see below for further details): in rural areas population counts in LSOA do not give an accurate representation of small-scale variation in the underlying population. In Appendix B, we give an example of how to specify a Poisson offset.

Other covariates in this model include `propmale`, the proportion of males in each LSOA; `Income`, income deprivation; `Employment`, employment deprivation; `Barriers`, deprivation in access to housing and services; `Crime` deprivation due to crime; and `Environment`, living

environment deprivation. Each of these variables is defined in the documentation available from the UK Government archives: http://webarchive.nationalarchives.gov.uk/, /http://www.communities.gov.uk/communities/neighbourhoodrenewal/deprivation/deprivation07/.

The next step in Algorithm 1 is to define our model,

```
R> FORM <- X ~ pop + propmale + Income + Employment + Education +
+    Barriers + Crime + Environment
```

and interpolate the independent variables in this model onto the computational grid. Further details on the interpolation methods are available in Appendix A. Note that the Bayesian MCMC functions in **lgcp** expect at least one spatial covariate (e.g., `X ~ 1` for an intercept, or `X ~ pop - 1` for a population covariate without intercept).

In the case where our model takes the form `X ~ 1`, i.e., the intercept-only case, the latest version of **lgcp** can be used to perform Bayesian inference for the models discussed in Taylor *et al.* (2013). The only minor difference is that using the Bayesian methods discussed here, the exponential of the posterior mean of the intercept would then be proportional to the expected number of cases over the observation window, which in Taylor *et al.* (2013) was estimated and fixed at the observed number of cases. Diggle *et al.* (2013) discuss another use of intercept-only models: as a model-based alternative to kernel smoothing of point-patterns.

Returning to the example at hand, we first guess at the type of interpolation for each variable:

```
R> popshape@data <- guessinterp(popshape@data)

LSOA04CD interpolation via Majority
LSOA04NM interpolation via Majority
pop interpolation via Majority
males interpolation via Majority
females interpolation via Majority
propmale interpolation via ArealWeightedMean
IMD interpolation via ArealWeightedMean
Income interpolation via ArealWeightedMean
Employment interpolation via ArealWeightedMean
Health interpolation via ArealWeightedMean
Education interpolation via ArealWeightedMean
Barriers interpolation via ArealWeightedMean
Crime interpolation via ArealWeightedMean
Environment interpolation via ArealWeightedMean
```

The function `guessinterp` assigns interpolation by area-weighted mean for any numeric variable and otherwise assigns interpolation by majority. It can be see that the population variables, `pop`, `males` and `females`, have by default been assigned interpolation by majority: this is not correct, as we wish population to represent the number of people in each computational grid square. We replace this with an area-weighted sum instead:

```
R> popshape@data <- assigninterp(df = popshape@data,
+    vars = c("pop", "males", "females"), value = "ArealWeightedSum")
```

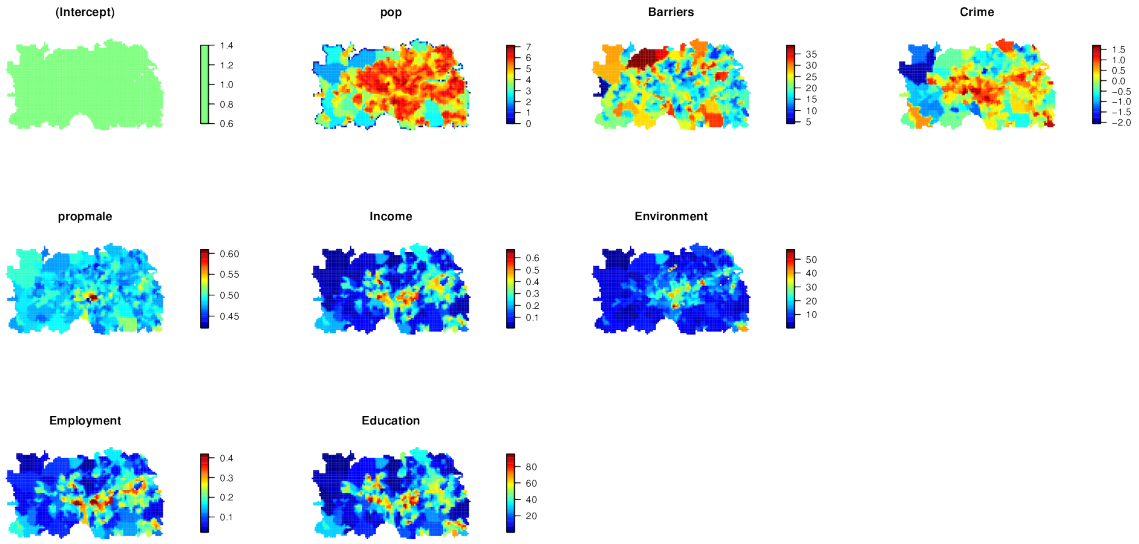

Figure 3: Plots of the interpolated covariates for the liver example.

so that,

```
R> class(popshape@data$pop)

[1] "ArealWeightedSum" "integer"
```

Next, we interpolate the covariate data onto the computational grid:

```
R> Zmat <- getZmat(formula = FORM, data = sd, regionalcovariates = popshape,
+    cellwidth = CELLWIDTH, ext = EXT, overl = polyolay)
```

As mentioned above, since we are using population as an explanatory variable, before we proceed to analyse the data we need to replace this covariate with the logarithm of population. This is because under a Poisson model, we expect the number of cases to be proportional to the population at risk (and not the exponential of population). Having interpolated the raw population counts above, we can now construct the logarithm as follows:

```
R> Zmat[, "pop"] <- log(Zmat[, "pop"])
R> Zmat[, "pop"][is.infinite(Zmat[, "pop"])] <- min(
+    Zmat[, "pop"][!is.infinite(Zmat[, "pop"])])
```

In the second line, we replace any zero population cell counts with the minimum value over the observation window; this avoids numerical problems handling negative infinite values. To see what the covariate data look like, we use

```
R> plot(Zmat)
```

which brings up a sequence of plots shown in Figure 3.

According to Algorithm 1, the next step is to define the population offset and priors. Since we are not using an offset in this example, we move onto defining the priors; note that more information on Poisson offsets together with an example are given in Appendix B.

The current version of **lgcp** allows two types of prior densities: a multivariate Gaussian prior for β and a multivariate Gaussian prior on the log-scale for the positive parameters σ and ϕ (and also θ in the spatiotemporal version) i.e.,

$$\beta \sim \mathrm{N}(\mu_\beta, \Sigma_\beta) \qquad \text{and} \qquad \eta = \{\log\sigma, \log\phi\} \sim \mathrm{N}(\mu_\eta, \Sigma_\eta).$$

We define these priors in **lgcp** as follows:

```
R> priors <- lgcpPrior(etaprior = PriorSpec(
+      LogGaussianPrior(mean = log(c(1, 500)), variance = diag(0.15, 2))),
+    betaprior = PriorSpec(
+      GaussianPrior(mean = rep(0, 9), variance = diag(10^6, 9))))
```

Note that the priors for η are always given in the order $\{\log\sigma, \log\phi\}$. Lastly, we specify the initial values and choice of covariance function for Y:

```
R> INITS <- lgcpInits(etainit = log(c(sqrt(1.5), 275)), betainit = NULL)
R> cf <- CovFunction(exponentialCovFct)
```

It is not necessary to specify an initial value for η or β as in the first line of code: by default, **lgcp** will initialise the MCMC using the prior mean for η, and for β it will initialise using the estimate obtained from an overdispersed Poisson `glm` fit of the cell counts against the covariates, and offset if appropriate. In the second line of code, we specify that the spatial dependence properties of Y should follow an exponential covariance function. For details on how to specify other sorts of covariance function, see Appendix C.

We are now in a position to run the MCMC algorithm. The following code runs the MALA chain for 1,000,000 iterations, with an initial burn-in of 100,000 iterations, followed by a further 900,000 iterations, of which every 900th sample is saved to disk. Note that the call to this function is not dissimilar to previous versions of the code, and we refer the reader to Taylor *et al.* (2013), for an explanation of the options not discussed below.

```
R> BASEDR <- getwd()
R> lg <- lgcpPredictSpatialPlusPars(formula = FORM, sd = sd, Zmat = Zmat,
+    model.priors = priors, model.inits = INITS, spatial.covmodel = cf,
+    cellwidth = CELLWIDTH, poisson.offset = NULL, mcmc.control = mcmcpars(
+      mala.length = 1000000, burnin = 100000, retain = 900,
+      adaptivescheme = andrieuthomsh(inith = 1, alpha = 0.5, C = 1,
+        targetacceptance = 0.574)),
+    output.control = setoutput(gridfunction = dump2dir(
+      dirname = paste(BASEDR, "/liver/", sep = ""), forceSave = TRUE)),
+    ext = EXT)
R> save(list = ls(), file = file.path(BASEDR, "liver", "liver.RData"))
```

The last step in Algorithm 1 is to perform diagnostic checks and then summarise the results: the package **lgcp** provides functions to aid in this process.

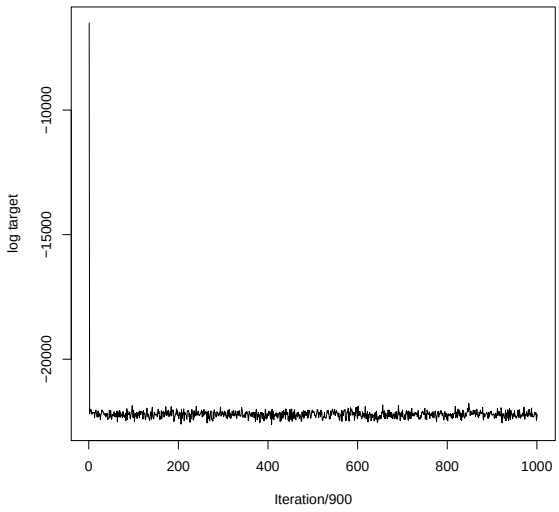

Figure 4: Diagnosing convergence to a posterior mode: a plot of the log target.

Diagnostic checks include (1) checking that the Markov chain is mixing well and (2) checking convergence of the Markov chain. Establishing convergence for models of this class is difficult due to the fact that the target is high-dimensional: there are 32,779 parameters to estimate in this case. A simple, but effective method to check that the chain has converged to a posterior mode is to examine a plot of the log-target, $\log\{\pi(\beta, \eta, Y|X)\} + c$ up to an additive constant c:

```
R> plot(ltar(lg), type = "s", xlab = "Iteration/900", ylab = "log target")
```

the results of which appear in Figure 4.

The plot shows that initially the chain was far away from a mode with the log-target having a value of around $-6{,}000$, but it quickly appears to have settled around values at about $-22{,}000$; note that these values have been thinned by the same amount as the original chain. If this plot does not appear to have converged, then this indicates that the Markov chain has not converged, and needs to be run for a longer period of time.

We next check the mixing of the latent field Y:

```
R> lagch <- c(1, 5, 15)
R> Sacf <- autocorr(lg, lagch, inWindow = NULL)
R> for(i in 1:3) {
+    image.plot(xvals(lg), yvals(lg), Sacf[, , i], zlim = c(-1, 1),
+      axes = FALSE, xlab = "", ylab = "", asp = 1,
+      sub = paste("Lag:", lagch[i]))
+    plot(sd$window, add = TRUE)
+    scalebar(5000, label = "5 km")
+  }
```

which produces the plots in Figure 5. These plots show from left-to-right the lag 1, 5 and 15 cellwise autocorrelation in the Y chain. Note that producing such plots is only possible if the chain has been dumped to disk (set using the `dump2dir` option in the call to `lgcpPredictSpatialPlusPars`).

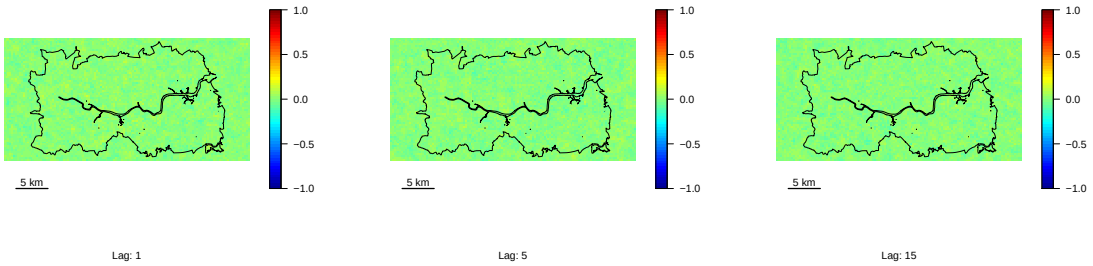

Figure 5: Left to right: lag 1, 5, 15 autocorrelation in the field $Y^{(j)}$.

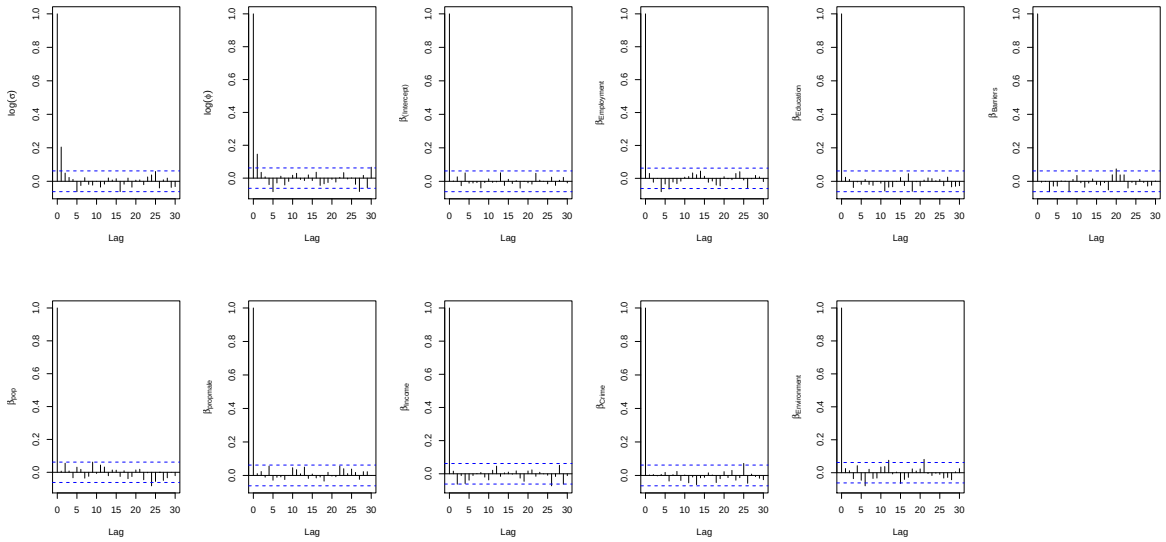

Figure 6: Autocorrelation plots of the parameters β and η from the spatial point process model for PBC.

These plots show that there is very little autocorrelation in the sampled $Y^{(j)}$. Similarly, we can produce autocorrelation plots for the parameters β and η using `parautocorr(lg)`, shown in Figure 6; and trace plots using the command `traceplots(lg)`, shown in Figure 7. For manual extraction of the β and η chains, use `betavals(lg)` and `etavals(lg)` respectively.

Having established satisfactory convergence of the chain, we can now proceed to making inferences from the model. We first produce a table of parameter estimates:

```
R> parsum <- parsummary(lg)
```

which can then be printed to the console by typing `parsum`. Alternatively, using

```
R> parsum <- parsummary(lg, LaTeX = TRUE)
R> library("miscFuncs")
R> latextable(parsum, rownames = rownames(parsum),
+    colnames = c("Parameter", colnames(parsum)), digits = 4)
```

converts the output to a LaTeX table, which can be copied and pasted into a LaTeX document and later edited by the user; the results are shown in Table 1.

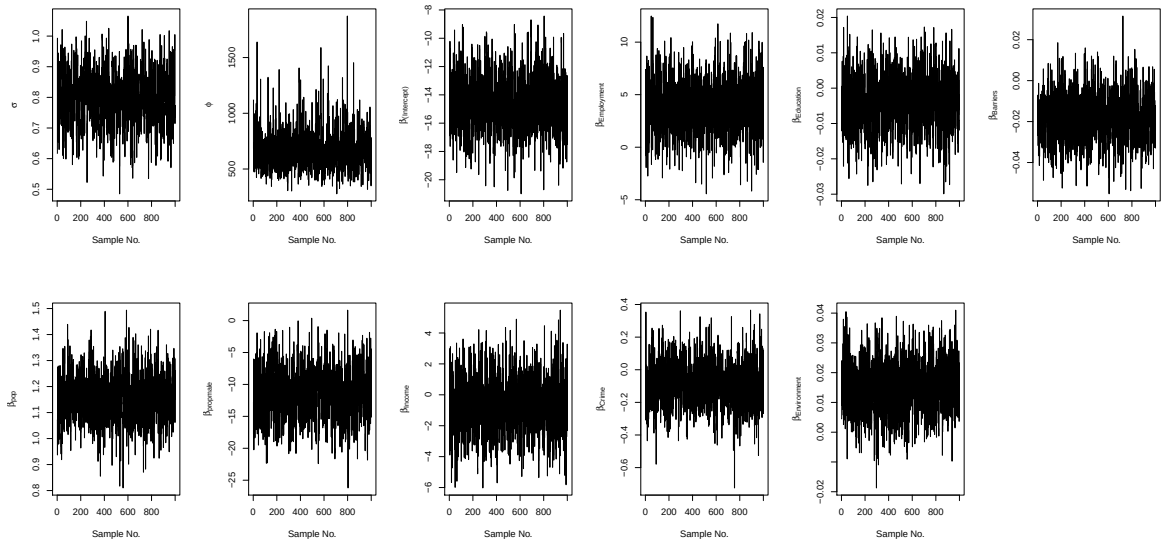

Figure 7: Trace plots of the parameters β and η from the spatial point process model for PBC.

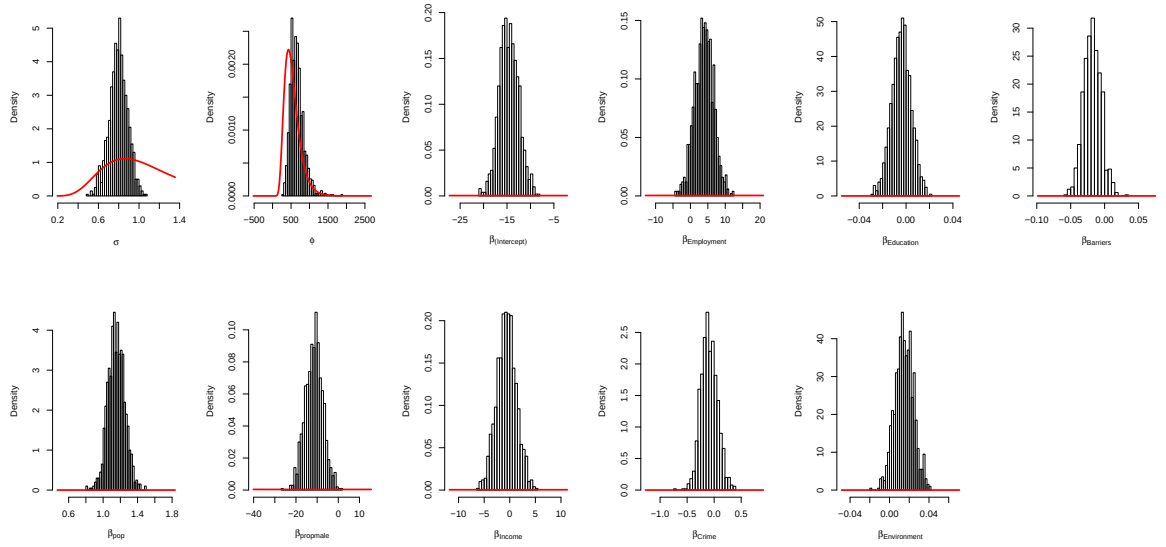

Figure 8: Plots of the prior and posterior values of each parameter.

A LaTeX formatted verbal summary of the table can also be produced:

```
R> textsummary(lg, digits = 4)
```

The output can be copied and pasted into a LaTeX document and later edited by the user, the result is shown in `quote` style below. In the user's report, it remains to edit the text as desired and add details of the variables and units, where appropriate.

Parameter	Median	Lower 95% CRI	Upper 95% CRI
σ	0.7999	0.6033	0.9695
ϕ	637.1	389.3	1098
$\exp(\beta_{(Intercept)})$	4.111×10^{-7}	8.735×10^{-9}	3.019×10^{-5}
$\exp(\beta_{pop})$	3.162	2.633	3.84
$\exp(\beta_{propmale})$	1.328×10^{-5}	3.937×10^{-9}	4.62×10^{-2}
$\exp(\beta_{Income})$	0.5449	1.425×10^{-2}	23.05
$\exp(\beta_{Employment})$	52.73	0.2343	9981
$\exp(\beta_{Education})$	0.9961	0.9797	1.012
$\exp(\beta_{Barriers})$	0.982	0.9594	1.007
$\exp(\beta_{Crime})$	0.9034	0.6956	1.223
$\exp(\beta_{Environment})$	1.015	0.9967	1.035

Table 1: Parameter estimates for the LGCP point pattern model for the PBC data.

A summary of the parameters of the latent field is as follows. The parameter σ had median 8×10^{-1} (95% CRI 0.603 to 0.97) and the parameter ϕ had median 637 (95% CRI 389 to 1098).

The following effects were found to be significant: each unit increase in `propmale` led to a reduction in relative risk with median 1.33×10^{-5} (95% CRI 3.94×10^{-9} to 4.62×10^{-2}); each unit increase in `pop` led to a increase in relative risk with median 3.16 (95% CRI 2.63 to 3.84).

The remainder of the main effects were not found to be significant: each unit increase in `Income` led to a reduction in relative risk with median 0.545 (95% CRI 1.42×10^{-2} to 23); each unit increase in `Education` led to a reduction in relative risk with median 0.996 (95% CRI 0.98 to 1.01); each unit increase in `Barriers` led to a reduction in relative risk with median 0.982 (95% CRI 0.959 to 1.01); each unit increase in `Crime` led to a reduction in relative risk with median 0.903 (95% CRI 0.696 to 1.22); each unit increase in `Employment` led to a increase in relative risk with median 52.7 (95% CRI 0.234 to 9981); each unit increase in `Environment` led to a increase in relative risk with median 1.01 (95% CRI 0.997 to 1.03).

It is also of interest to examine plots of the prior and posterior distributions of the parameters. This is particularly important to help us interpret of the posterior density of the spatial scale parameter, ϕ, which tends not to be well identified by the data, see Zhang (2004) for an example in the classical geostatistical context. Typing

```
R> priorpost(lg)
```

produces the plots in Figure 8. These plots confirm that whilst the covariate effects, β, are well identified by the data, the parameters of the process Y are not so well identified. The parameter σ shows a greater departure from the prior compared with the parameter ϕ; one must therefore exercise caution in making strongly probabilistic inferential statements about these parameters, since they appear to be influenced by the prior.

Next we produce a plot of the posterior covariance function using:

```
R> postcov(lg)
```

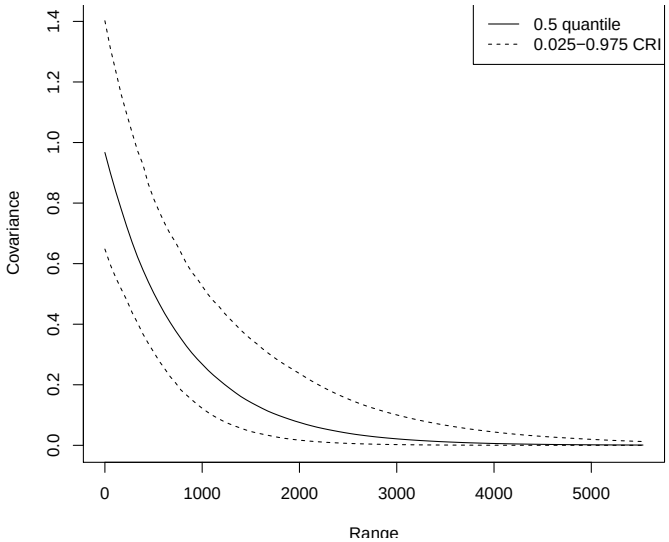

Figure 9: Plot of the posterior estimated covariance function of Y.

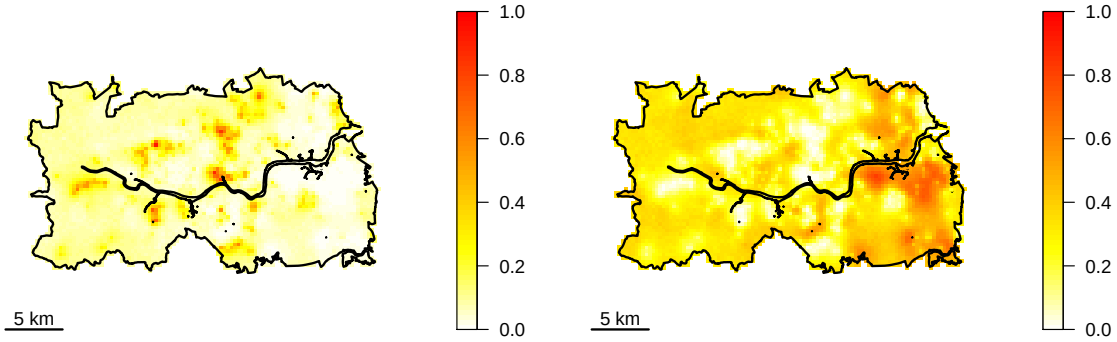

Figure 10: Left: Plot of the posterior probability that the relative risk exceeds 2. Right: Plot of the posterior probability that the relative risk is below 0.5. Note that by setting `sub = NULL`, or omitting `sub` from the call to `plotExceed` above, the threshold will be printed as a subtitle in each of the plots (i.e., printing the threshold as a subtitle is the default behaviour for `plotExceed`).

the results are shown in Figure 9.

Finally, it is also of interest in epidemiology to understand if there are some spatial areas of (covariate-adjusted) particularly high or low incidence. These exceedance (or respectively lower-tail exceedance) probabilities, are $\mathsf{P}\{\exp(Y) > k|X\}$ or $\mathsf{P}\{\exp(Y) < k|X\}$ for a pre-specified threshold k. These quantities can be expressed as a posterior expectation,

$$\mathsf{P}[\exp(Y) > k|X] = \mathsf{E}_{\pi(\beta,\eta,Y|X)}\{\mathbb{I}[\exp(Y) > k]\} = \frac{1}{n}\sum_{i=1}^{n}\mathbb{I}[\exp(Y^{(i)}) > k],$$

where $\mathbb{I}$ is the indicator function. We use the `exceedProbs` function to set up these probabilities and `lgcp:::expectation.lgcpPredict` to compute the Monte Carlo expectation; note that an explicit call to `lgcp:::expectation.lgcpPredict` *is* necessary here as in the latest

version of **lgcp** there is a new method `expectation` for functions of β, η *and* Y for objects generated by the function `lgcpPredictSpatialPlusPars`. See Appendix D for examples of more complex expectations. The code below generates the plots in Figure 10.

```
R> ep <- exceedProbs(c(1.5, 2, 5, 10))
R> sp <- exceedProbs(c(2/3, 1/2, 1/5, 1/10), direction = "lower")
R> ex <- lgcp:::expectation.lgcpPredict(lg, ep)
R> su <- lgcp:::expectation.lgcpPredict(lg, sp)
R> plotExceed(ex[[1]], "ep", lg, zlim = c(0,1), asp = 1,
+    axes = FALSE, xlab = "", ylab = "", sub = "")
R> scalebar(5000, label = "5 km")
R> plotExceed(su[[1]], "sp", lg, zlim = c(0, 1), asp = 1,
+    axes = FALSE, xlab = "", ylab = "", sub = "")
R> scalebar(5000, label = "5 km")
```

4.2. PBC in Newcastle-Upon-Tyne, an aggregated count model

Introduction to continuous models for areal count data

Now suppose that instead of observing the exact location of events we instead observe T_i, the total number of cases in region A_i, where $A_i \cap A_j = \emptyset$ for $i \neq j$, $\bigcup_{i=1}^{m} A_i = W$ and W is the observation window; see Figure 11.

Aggregated exposure or outcome data are often used because individual-level information is not available for economic or confidentiality reasons (Beale, Abellan, Hodgson, and Jarup 2008; Diggle, Guan, Hart, Paize, and Stanton 2010). Area-level analyses are prone to so-called 'ecological bias' (Wakefield and Lyons 2010; Wakefield, Haneuse, Dobra, and Teeple 2011); the name refers to differences in effect sizes that result from modelling association

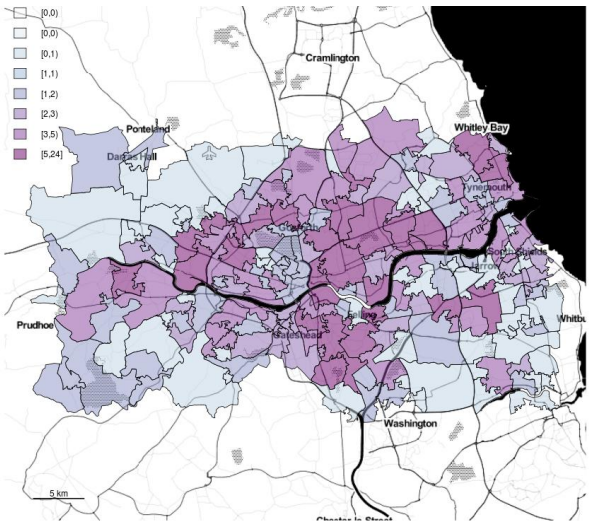

Figure 11: Plot of cases of primary biliary cirrhosis in Newcastle-Upon-Tyne, case counts aggregated to regions.

between exposure and outcome at different spatial resolutions. Spatial models for aggregated data include the popular Besag, York, and Mollié (1991) model, as well as conditional and simultaneous autoregressive models. Whilst aggregated count models such as these are computationally quick to fit, there is an argument against using them when the regions are quite varied in shape and size, as the definition of what it means to be a neighbour is then somewhat contrived and can lead to undesirable properties in parameter estimates, see Wall (2004).

Other authors e.g., Møller *et al.* (1998), Brix and Diggle (2001) and Diggle *et al.* (2005a), take the intuitively more natural approach of modelling variation in risk as a spatially continuous process, but their methods do not directly handle areal data. Kelsall and Wakefield (2002) model area-level counts as a product of the expected number of counts (based on population demography) and a relative risk term, which they model as a spatially continuous log-Gaussian process, from which they were able to compute covariances between regions accounting for each region's size and shape.

Our aim in the present article is to fit a model of the form given in Equation 5 to areal count data: the number of events T_i in each region A_i. In this case, it is not only Y, η and β that are unknown, but also the event counts in each computational grid cell. In order to proceed, we use the technique of data augmentation, see van Dyk and Meng (2001) for a review. We augment the list of parameters, $\{\beta, \eta, Y\}$, with an additional variable N, the cell counts and sample from,

$$\pi(\beta, \eta, Y, N|T_1, \ldots, T_m).$$

This can be achieved using a Gibbs scheme, alternately sampling from $\pi(\beta, \eta, Y|N, T_{1:m})$ and $\pi(N|\beta, \eta, Y, T_{1:m})$, see Li, Brown, Gesink, and Rue (2012) and Diggle *et al.* (2013). Note that the random variable N is akin to the observed data X in Equation 5. Conditional independence properties imply that

$$\pi(\beta, \eta, Y|N, T_{1:m}) = \pi(\beta, \eta, Y|N).$$

We sample from this density using exactly the same MALA algorithm as for the spatial point process. The density $\pi(N|\beta, \eta, Y, T_{1:m})$ turns out to be multinomial, and so is straightforward to sample from. Of critical importance to the data augmentation is the way in which **lgcp** handles the polygon/polygon overlay operations: each non-trivial intersection between the computational grid cells and the `SpatialPolygonsDataFrame` is computed, allowing accurate sampling from $\pi(N|, \beta, \eta, Y, T_{1:m})$. Although straightforward in principle, sampling from $\pi(N|\beta, \eta, Y, T_{1:m})$ nevertheless incurs a computational cost. Rather than doing this every iteration, the MCMC function for aggregated data, `lgcpPredictAggregateSpatialPlusPars`, has an argument `Nfreq`, which allow the user to set this frequency; by default, this is set to draw a new $N \sim \pi(N|\beta, \eta, Y, T_{1:m})$ every 101 iterations.

Areal count data in ***lgcp***

We now discuss how to fit a spatially continuous log-Gaussian Cox process model to areal count data. These data were contained in a `SpatialPolygonsDataFrame` object `spdf`,

```
R> load("liver_spdf.RData")
R> spdf

class       : SpatialPolygonsDataFrame
nfeatures   : 177
```

```
extent      : 409144.1, 439753.8, 556265.4, 574219.4 (xmin, xmax, ymin, ymax)
coord. ref. : NA
nvariables  : 1
names       :  X
min values  :  0
max values  : 24
```

a plot of these data is in Figure 11. The object `spdf` has a column X containing the event counts in each polygon.

We initially used the same `CELLWIDTH` and `EXT` parameters as for the point process version of our model. However, in pilot runs it became apparent that for this model, we needed to use a larger value of `EXT`, see Appendix E for details on this matter. Otherwise, we used the same formula, `FORM`; priors, `priors`; and covariance function, `cf`, as in the point process version of this model discussed in the previous section. This allows us to compare inferences where the point locations are known and where the point locations are unknown.

For this example, we did not use minimum contrast methods to obtain initial estimates of the parameters. Had the collection of polygons in `spdf` been on a sufficiently fine scale to capture spatial variation in population reasonably well, then the function `spSample` could have been used to impute cases into the observation window, from which minimum contrast estimates could have been obtained using `minimum.contrast` as before. However, in this instance the set of polygons *do not* capture this fine scale variation, so we instead opt to initialise the MCMC algorithm using the prior mean.

Running the following commands,

```
R> CELLWIDTH <- 300
R> EXT <- 3
R> polyolay <- getpolyol(data = spdf, cellwidth = CELLWIDTH,
+    regionalcovariates = popshape, ext = EXT)
R> Zmat <- getZmat(formula = FORM, data = spdf, cellwidth = CELLWIDTH,
+    regionalcovariates = popshape, ext = EXT, overl = polyolay)
R> Zmat[, "pop"] <- log(Zmat[, "pop"])
R> Zmat[, "pop"][is.infinite(Zmat[, "pop"])] <- min(
+    Zmat[, "pop"][!is.infinite(Zmat[, "pop"])])
```

sets up the polygon/polygon overlay and performs interpolation onto the computational grid and replaces population with log-population as in the point process model.

The function call to run the MCMC routine is:

```
R> BASEDR <- getwd()
R> lg <- lgcpPredictAggregateSpatialPlusPars(formula = FORM, spdf = spdf,
+    Zmat = Zmat, overlayInZmat = FALSE, model.priors = priors,
+    spatial.covmodel = cf, cellwidth = CELLWIDTH, mcmc.control = mcmcpars(
+      mala.length = 3100000, burnin = 100000, retain = 3000,
+      adaptivescheme = andrieuthomsh(inith = 1, alpha = 0.5, C = 1,
+        targetacceptance = 0.574)),
+    output.control = setoutput(gridfunction = dump2dir(
```

```
+      dirname = paste(BASEDR, "/liveraggregated/", sep = ""),
+      forceSave = TRUE)),
+    ext = EXT)
R> save(list = ls(), file = file.path(
+    BASEDR, "liveraggregated", "liveraggregated.RData"))
```

In this call, the option `overlayInZmat` is used to declare when the `regionalcovariates` object from the calls to `getpolyol` and `getZmat` is the same as the object `spdf`: if these objects are *not* the same, then a new polygon/polygon overlay of the computational grid onto `spdf` is computed. The MALA chain was run with a initial burn-in of 100,000 iterations and followed by a further 3,000,000 iterations, of which every 3,000th sample was retained.

As before, we can produce convergence diagnostics including a plot of the log-target with `ltar(lg)`; plots of the autocorrelation in the latent field, using `autocorr(lg, c(1, 5, 15))` for example; trace plots, using `traceplots(lg)`; and autocorrelation plots for the other model parameters using `parautocorr(lg)`.

Having established convergence and ascertained satisfactory mixing of the MCMC, we can proceed to produce plots of the prior and posterior for η and β using `priorpost(lg)`, not shown; producing plots of the posterior covariance function using `postcov(lg)`, shown in Figure 12; summarising parameter estimates in a table using `parsummary`, shown in Table 2; and computing exceedance probabilities, see Figure 13.

The main differences between the results from the point process model compared with the aggregated model can be seen on comparing Figure 10 with Figure 13: the latter shows a much more attenuated relative-risk surface; this is expected due to the uncertainty in the precise location of cases.

The point estimates of σ from the two models were similar: 0.80 (CRI 0.60, 0.97) from the point process version and 0.84 (CRI 0.56, 1.22) from the aggregated version, though the confidence interval from the latter was wider. The point estimate of ϕ from the aggregated model: 595 metres (CRI 283, 1269 metres) was comparable with the 637 metres (CRI 389, 1098) from the point process version; again there was greater uncertainty in the estimates from the aggregated model.

4.3. Spatiotemporal point process data

Introduction

In this section, we show how to fit a spatiotemporal log-Gaussian Cox process. Our model for the spatiotemporal data is given in Equation 6:

$$\begin{aligned} X(s,t) &= \text{Poisson}[R(s,t)] \\ R(s,t) &= C_A \lambda(s,t) \exp\{Z(s,t)\beta + Y(s,t)\} \end{aligned} \tag{6}$$

In this model, $X(s,t)$ is the number of events in the cell of the computational grid containing the point s at time t, $R(s,t)$ is the Poisson rate, C_A is the cell area, $\lambda(s,t)$ is a known offset, $Z(s,t)$ is a vector of measured covariates and $Y(s,t)$ is the latent Gaussian process on the computational grid. The parameters in the model are β, the covariate effects; and $\eta = \{\log(\sigma), \log(\phi), \log(\theta)\}$, the parameters of the spatiotemporal Gaussian process Y. We

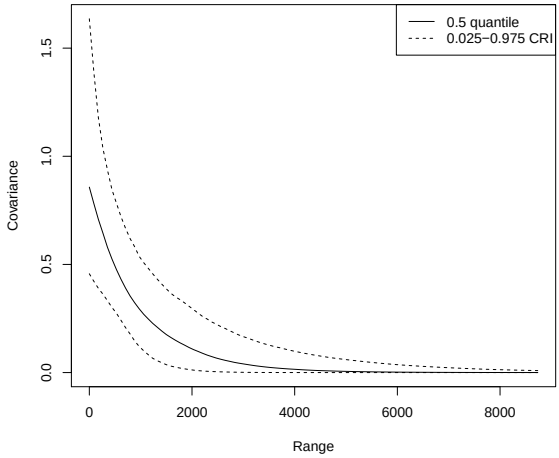

Figure 12: Plot of the posterior estimated covariance function of Y for the aggregated cirrhosis example.

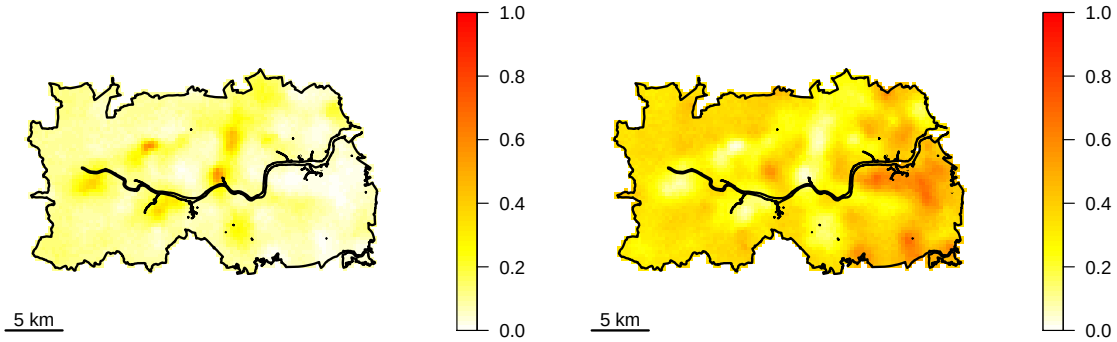

Figure 13: Exceedance probabilities from the aggregated count model for the liver data. Left: Plot of the posterior probability that the relative risk exceeds 2. Right: Plot of the posterior probability that the relative risk is below 0.5.

Parameter	Median	Lower 95% CRI	Upper 95% CRI
σ	0.8423	0.5642	1.217
ϕ	594.9	283	1269
$\exp(\beta_{(Intercept)})$	7.86×10^{-9}	1.702×10^{-12}	3.817×10^{-5}
$\exp(\beta_{pop})$	3.546	2.693	4.729
$\exp(\beta_{propmale})$	2.625×10^{-2}	7.451×10^{-10}	1.246×10^{6}
$\exp(\beta_{Income})$	7.808	1.547×10^{-2}	4049
$\exp(\beta_{Employment})$	1.645	5.116×10^{-5}	54304
$\exp(\beta_{Education})$	0.9882	0.959	1.014
$\exp(\beta_{Barriers})$	0.986	0.9468	1.023
$\exp(\beta_{Crime})$	0.9067	0.6043	1.412
$\exp(\beta_{Environment})$	1.001	0.9631	1.034

Table 2: Parameter estimates from the aggregated cont model for the liver data.

assume a separable spatiotemporal covariance function for Y:

$$\text{cov}[Y(s_1,t_1),Y(s_2,t_2)] = \sigma^2 \exp\{-||s_2 - s_1||/\phi - |t_2 - t_1|/\theta\}.$$

The parameter θ determines the temporal correlation in the process Y.

This model is an extension of the one proposed in Brix and Diggle (2001); Diggle *et al.* (2005a) and implemented in Taylor *et al.* (2013), in which,

$$R(s,t) = C_A \mu_0(t) \lambda_0(s) \exp\{Y(s,t)\},$$

where μ_0 and λ_0 were respectively known temporal and spatial offsets, and the parameters of Y were assumed known.

Our new model in Equation 6 not only allows the user to specify an offset of the form $\lambda(s,t)$, but also includes estimation of covariate effects and model parameters via Bayesian inference. In this section we will illustrate how to fit this model using as an example a short section of observed data from the AEGISS project (Ascertainment and Enhancement of Gastroenteric Infection Surveillance Statistics, see `http://www.maths.lancs.ac.uk/~diggle/Aegiss/day.html%3Fyear=2002&month=2&day=11&exceed=2`) for an example of the output that was updated on a daily basis, see Diggle, Knorr-Held, Rowlingson, Su, Hawtin, and Bryant (2003) for a full description of the project.

Suppose we have data up to the present, time T, then we should ideally like to obtain samples from $\pi(\beta,\eta,Y|X_0,\ldots,X_T)$, the posterior of the parameters given the complete history of observations at each time, $X_0,\ldots,X_T$. Brix and Diggle (2001) argue that since (1) the complete posterior grows with time, making the inferential problem increasingly challenging and (2) the alternative of solving the filtering recursions for this target is not tractable, a pragmatic approach is to ignore data from far in the past, and sample from $\pi(\beta,\eta,Y|X_{T-l},\ldots,X_T)$ instead, where l is a lag parameter to be set by the user. The methods we present in this section adopt the same strategy.

A spatiotemporal analysis

The points in this dataset are the geo-locations of callers to the NHS Direct telephone service who reported symptoms of diarrhoea or vomiting. The main idea of the AEGISS project was to perform daily spatiotemporal surveillance of the rate of such calls, which could be a used as a proxy for the incidence of gastrointestinal illness.

The data for our example are shown in Figure 14. These are case counts observed over a period of two weeks ending on 18th August, 2002.

We begin by loading the data: an object `xyt` of class `stppp`. Following Algorithm 1, we begin by computing approximate parameter estimates to help inform the computational grid size and initial values for the MCMC routine.

```
R> load("aegiss.RData")
R> minc <- minimum.contrast.spatiotemporal(xyt, model = "exponential",
+    method = "g", transform = log, spatial.dens = density.ppp(xyt),
+    temporal.intens = muEst(xyt))
R> minc
```

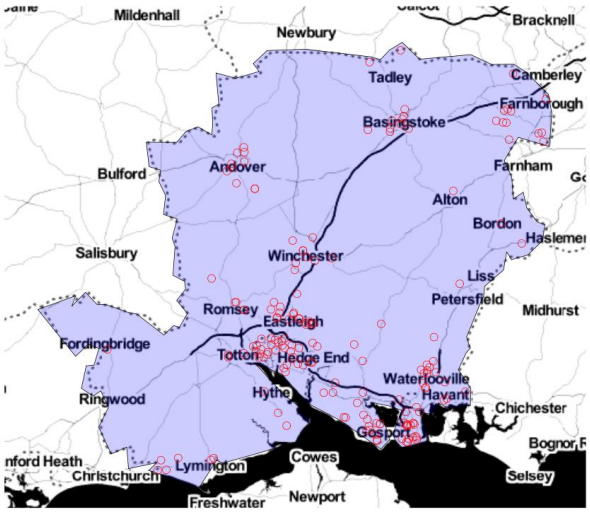

Figure 14: Plot of locations of calls to NHS Direct reporting suspected gastrointestinal infection in Hampshire.

```
$estimates
     scale (spatial) variance (spatial) scale (temporal)
[1,]        1286.983             2.86001        0.5336735

$discrepancy
     Squared discrepancy (spatial)
[1,]                       5455.624
```

In the above, we use the kernel-smoothed estimates `density.ppp(xyt)` and `muEst(xyt)` to account for the inhomogeneity in the spatial and temporal intensity of events, respectively. We chose a cell width of 1,400 metres by running the command `chooseCellwidth(xyt, cwinit = 1400)` for various values of `cwinit`; this value is just sufficient to capture the spatial correlation. Looking at values of $\exp(-(0:7)*0.53)$ shows that, under the assumption that the estimated temporal correlation parameter is approximately 0.53, we expect temporal correlation in the Y process to drop to 0.02 after 7 days. We therefore initially used an observation window width of 7 days. However, on running the MCMC algorithm, it was noted that the quadratic approximation to the posterior with respect to the latent parameters of Y was poor, leading to a proposal variance matrix that was not symmetric and positive definite. In cases such as these, the MCMC routine will produce a warning:

```
Warning: Cannot find good approximation of posterior
      variance w.r.to eta: using the following variance instead:
```

and the default is instead to use a Σ_η (see Equation 4) that is diagonal with entries equal to 1/100 times the prior variance for these parameters. Occasionally, this choice will lead to an algorithm that mixes reasonably well, but in our case, we wanted a better choice of proposal variance. We therefore decided to use 14 days of data, which enabled a sensible proposal matrix to be computed via the quadratic approximation.

Our next task is to set up the computational grid and overlay onto the covariate data; we set the extension parameter equal to 2:

```
R> CELLWIDTH <- 1400
R> EXT <- 2
R> polyolay <- getpolyol(data = xyt, regionalcovariates = popshape,
+    cellwidth = CELLWIDTH, ext = EXT)
```

We use three covariates in this analysis: population (`pop`); IMD (`IMD`); and day-of-the-week (`dotw`) as an indicator. Our population variable comes from the 2001 census and is therefore only available in aggregated form, as before. As in the liver example, we use IMD data from the 2007 survey as a proxy for deprivation in 2002; we emphasise that this is for illustrative purposes and is not ideal.

In the following, we load the `SpatialPolygonsDataFrame` containing the spatial covariate data and specify the interpolation method for each of the variables

```
R> load("popshape_aegiss.RData")
R> popshape@data <- guessinterp(popshape@data)
R> popshape@data <- assigninterp(df = popshape@data,
+    vars = c("pop", "males", "females"), value = "ArealWeightedSum")
```

We next construct a list of design matrices Z to feed into the prediction algorithm. Since we use data from 14 time points in the analysis, we are required to create a list of 14 objects, each element having been constructed in one of four ways:

1. where $Z(s,t)$ cannot be decomposed, i.e., Z are true spatiotemporal covariates. In this case, each element of the list must be constructed separately using the `getZmat` command 14 times on the covariates for each time point.

2. $Z(s,t)\beta = Z_1(s)\beta_1 + Z_2(t)\beta_2$: the spatial and temporal effects are separable; in this case, the package **lgcp** provides a function, `addTemporalCovariates`, to aid in the construction of the list, as illustrated below.

3. $Z(s,t)\beta \equiv Z(s)\beta$, in which case the user only needs to perform the interpolation using `getZmat` once: each of the fourteen elements of the list will be identical.

4. $Z(s,t)\beta \equiv Z(t)\beta$, in which case we follow the procedure for the separable case in item 2 above. For example, if we did not have a spatial population covariate, but a temporal day-of-the-week covariate, then we would follow the example below, but with `FORM <- X ~ dotw`, `FORM.spatial <- X ~ 1`, followed by `TEMPORAL.FORMULA <- t ~ dotw - 1`. This would result in a design matrix incorporating an intercept and indicator variables for the six other levels of the day-of-the-week effects.

In this example, we treat day-of-the-week and population/IMD as separable effects. We define three formulae in this case:

```
R> FORM <- X ~ pop + IMD + dotw
R> FORM.spatial <- X ~ pop + IMD
R> TEMPORAL.FORMULA <- t ~ dotw - 1
```

The first object, FORM, is the overall model for these data and will eventually be passed on to the MCMC algorithm lgcpPredictSpatioTemporalPlusPars; the second, FORM.spatial, will be used to construct a design matrix for the spatial element of the model; the last, TEMPORAL.FORMULA will be used to 'bolt on' the temporal covariate information. Note that in the latter, there is no intercept term in the formula: this is important, as the intercept is already included in the formula for the spatial covariates.

We begin by interpolating the spatial covariates and taking the logarithm of population in a similar way to our first two analyses in the preceding sections:

```
R> Zmat <- getZmat(formula = FORM.spatial, data = xyt, cellwidth = CELLWIDTH,
+    regionalcovariates = popshape, ext = EXT, overl = polyolay)
R> Zmat[, "pop"] <- log(Zmat[, "pop"])
R> Zmat[, "pop"][is.infinite(Zmat[, "pop"])] <- min(
+    Zmat[, "pop"][!is.infinite(Zmat[, "pop"])])
```

In this example, the inferential observation window in the object xyt had been artificially constructed by hand, whereas the covariate data were obtained using a more accurate representation of the Hampshire border. This discrepancy between the inferential owin-type observation window and the covariate SpatialPolygonsDataFrame leads to missing covariate values because there are cells touching the observation window for which covariate values cannot be inferred. In principle, this can be avoided by making the inferential observation window equal to the covariate SpatialPolygonsDataFrame, but this is not always possible in practice. The issue in this dataset partly arises because Hampshire is not landlocked; had it been landlocked we could have used covariate data from Hampshire plus a buffer to allow the interpolation step, called by getZmat, to function without a warning being issued.

Typing plot(Zmat,misscol="yellow",obswin=xyt$window) at the console allows the user to check which cells have had covariate values imputed (this will affect all covariate values for those cells). Running this command plots the covariates and highlights by a yellow + symbol which cells had missing values, allowing the user to check that the imputed median is a sensible choice. We emphasise that median imputation is only one option. The user can also replace any missing value value manually; note that attr(Zmat,"missingind") is a vector with a 1 in position i if the covariates Zmat[i,] were imputed, a 0 if the covariates were not imputed and NA if the cell is not part of the inferential observation window.

We now construct a second data frame, tdata, from which the temporal effects will be extracted and Zmat turned into the required list object, each element of which is the design matrix for the appropriate time point in the analysis. Day 1 of the AEGISS project was the 1st January 2001, a Monday. The data frame tdata contains the time variable t and the day-of-the-week variable dotw:

```
R> days <- c("Mon", "Tue", "Wed", "Thur", "Fri", "Sat", "Sun")
R> tvec <- xyt$tlim[1]:xyt$tlim[2]
R> da <- rep(days, length.out = length(tvec))
R> tdata <- data.frame(t = tvec, dotw = da)
```

The subset of data we use in this analysis ends on time point 595, stored in the object T below, and includes the preceding 13 days of data, specified by the object LAGLENGTH.

```
R> T <- 595
R> LAGLENGTH <- 13
R> ZmatList <- addTemporalCovariates(temporal.formula = TEMPORAL.FORMULA,
+    T = T, laglength = LAGLENGTH, tdata = tdata, Zmat = Zmat)
```

Note that as in the construction of the `Zmat` argument, in general the `poisson.offset` for the spatiotemporal LGCP must also be provided as a list of offset terms, see Appendix B for further details. The last task before we can run the MCMC algorithm is to specify the priors and initial values. We initialise the MCMC routine using the minimum contrast estimates above. For the priors, the main difference compared with the previous two analyses is that the prior for η is now a multivariate normal with 3 parameters, as opposed to 2. The prior for γ, is an informative prior, and based on the initial value of $\log\theta$ either specified by the user, or by default equal to the exponential of the mean of the prior for $\log\theta$:

$$\log\pi(\gamma) = \text{const} - \frac{1}{2}\sum_{i=1}^{n}\frac{||\gamma_i||^2}{g_i} - \frac{||\gamma_0||^2}{2}, \tag{7}$$

where $g_i = 1 - \exp\{-2\Delta t_i\theta_{\text{init}}\}$, θ_{init} is the initial value of θ and Δt_i is the time difference between the observations at time t_i and those at t_{i-1}, with the convention that $g_0 = 1$, see Taylor and Diggle (2013).

We choose an exponential covariance function to model the spatial dependency in Y, stored in the object `CF`.

```
R> INITS <- lgcpInits(etainit = log(c(sqrt(2.8), 1286, 0.5)),
+    betainit = NULL)
R> lgprior <- PriorSpec(LogGaussianPrior(mean = log(c(1, 2000, 1)),
+    variance = diag(0.2, 3)))
R> gprior <- PriorSpec(GaussianPrior(mean = rep(0, 9),
+    variance = diag(1e6, 9)))
R> priors <- lgcpPrior(etaprior = lgprior, betaprior = gprior)
R> CF <- CovFunction(exponentialCovFct)
```

Now there is sufficient information to run the MALA algorithm. In the code below, we run the chain for 1,000,000 iterations including a 100,000 iteration burn in and retain every 900th post-burn-in sample for inference.

```
R> DIRNAME <- getwd()
R> lg <- lgcpPredictSpatioTemporalPlusPars(formula = FORM, xyt = xyt, T = T,
+    laglength = LAGLENGTH, ZmatList = ZmatList, model.priors = priors,
+    model.inits = INITS, spatial.covmodel = CF, cellwidth = CELLWIDTH,
+    mcmc.control = mcmcpars(mala.length = 1000000, burnin = 100000,
+      retain = 900, adaptivescheme = andrieuthomsh(inith = 1, alpha = 0.5,
+        C = 1, targetacceptance = 0.574)),
+    output.control = setoutput(gridfunction = dump2dir(
+      dirname = file.path(DIRNAME, "aegiss"), forceSave = TRUE)),
+    ext = EXT)
R> save(list = ls(), file = file.path(DIRNAME, "aegiss", "aegissout.RData"))
```

Parameter	Median	Lower 95% CRI	Upper 95% CRI
σ	0.7797	0.4475	1.164
ϕ	1848	787.2	4513
θ	0.9249	0.4084	2.319
$\exp(\beta_{(Intercept)})$	5.268×10^{-13}	1.396×10^{-13}	1.694×10^{-12}
$\exp(\beta_{pop})$	3.462	2.974	4.108
$\exp(\beta_{IMD})$	0.99	0.9683	1.01
$\exp(\beta_{dotwMon})$	1.485	0.8402	2.701
$\exp(\beta_{dotwSat})$	1.152	0.6674	2.169
$\exp(\beta_{dotwSun})$	1.352	0.7298	2.561
$\exp(\beta_{dotwThur})$	1.141	0.5918	2.089
$\exp(\beta_{dotwTue})$	1.287	0.7168	2.378
$\exp(\beta_{dotwWed})$	1.038	0.5549	1.93

Table 3: Table of parameter estimates from the AEGISS aexample.

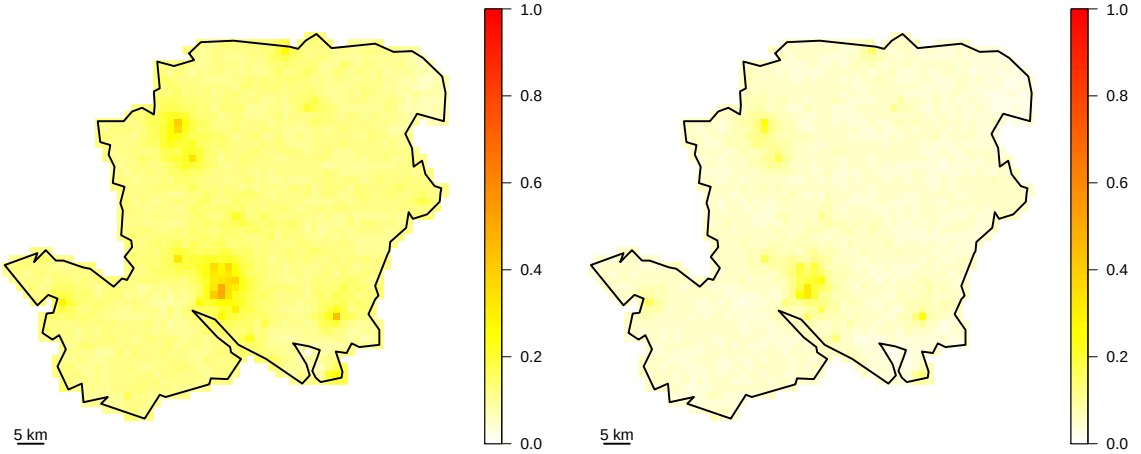

Figure 15: Plot of the posterior probability that the relative risk exceeds 1.5 (left) and 2 (right).

Having run the MCMC routine, we next establish convergence and good mixing as in the previous two sections, before summarising the results in the form of tables (e.g., Table 3) and computing Monte Carlo estimates of exceedance probabilities (Figure 15).

We edited text from the output to `textsummary(lg, digits = 4)` to produce the following summary of the main effects model and parameters of the latent field.

A summary of the parameters of the latent field Y is as follows. The parameter σ had median 0.78 (95% CRI 0.448 to 1.16); the parameter ϕ had median 1848 metres (95% CRI 787 to 4513); and the parameter θ had median 0.925 days (95% CRI 0.408 to 2.32). Examining plots of the priors and posteriors for these paramters suggested that whilst there was information in the data on σ, the parameters ϕ and θ were not well-identified by the data, and the confidence intervals for these parameters follow what would have been obtained from the prior. A plot of the posterior spatial covariance and temporal correlation in Y was produced using `postcov(lg)`, see Figure 16.

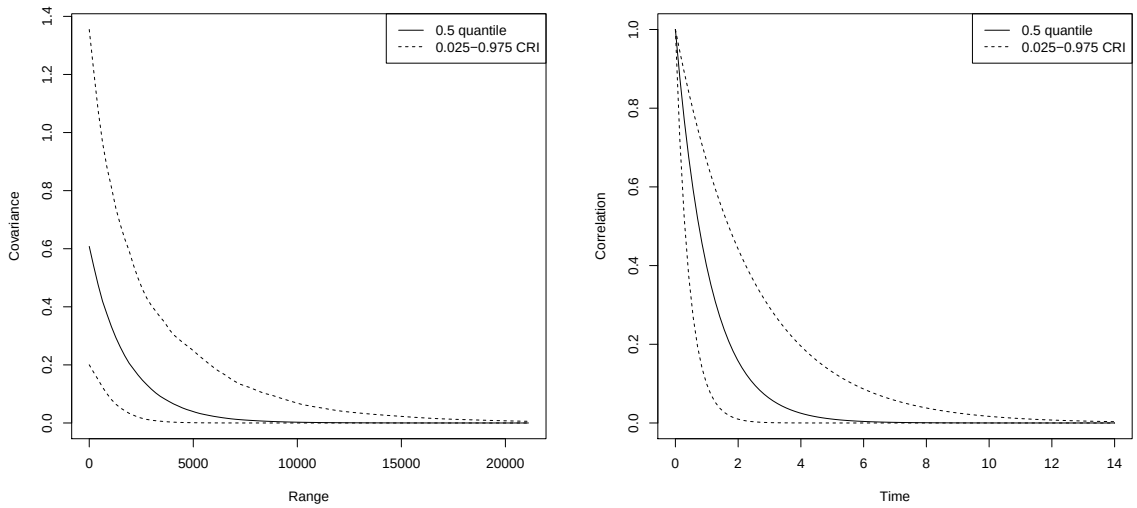

Figure 16: Plots of the posterior spatial covariance (left) and temporal correlation (right).

The following effects were found to be significant: each unit increase in log-population led to a increase in relative risk with median 3.46 (95% CRI 2.97 to 4.11). The remainder of the main effects were not found to be significant. Reporting rates were lower in areas of higher deprivation: each unit increase in `IMD` led to a reduction in relative risk with median 0.99 (95% CRI 0.968 to 1.01). The other effects in Table 3 show the relative risk on other days of the week compared with Friday.

4.4. BTB in Cornwall, a multivariate log-Gaussian Cox process

Introduction

In this section, we discuss the fitting of a multivariate log-Gaussian process to a multitype point pattern. The general form of our model is:

$$\begin{aligned} X_k(s) &= \text{Poisson}[R_k(s)] \\ R_k(s) &= C_A \lambda_k(s) \exp\{Z(s)_k \beta_k + Y_k(s) + Y_{K+1}(s)\} \qquad k \in 1, \ldots, K \end{aligned} \tag{8}$$

Here $X_k(s)$ is the number of events of type k in the computational grid cell containing the point s, $R_k(s)$ is the Poisson rate, C_A is the cell area, $\lambda_k(s)$ is a known offset, $Z_k(s)$ is a vector of measured covariates and $Y_i(s)$ where $i = 1, \ldots, K+1$ are latent Gaussian processes on the computational grid. The other parameters in the model are β_k, the covariate effects for the kth type; and $\eta_i = \{\log(\sigma_i), \log(\phi_i)\}$, the parameters of the process Y_i for $i = 1, \ldots, K+1$ on an appropriately transformed (again, in this case log) scale.

Our model for the kth data stream (type) is a log-Gaussian Cox process in which the stochastic part is composed of two elements: a within-stream and a between-stream component. The idea of this model is that it allows us to decompose the spatial variation in events of multiple types into variation associated with a particular type of event and variation common to all types. Thus, although each point type may display an individual spatial pattern, the process Y_{K+1} captures areas of high or low intensity that are common to all types.

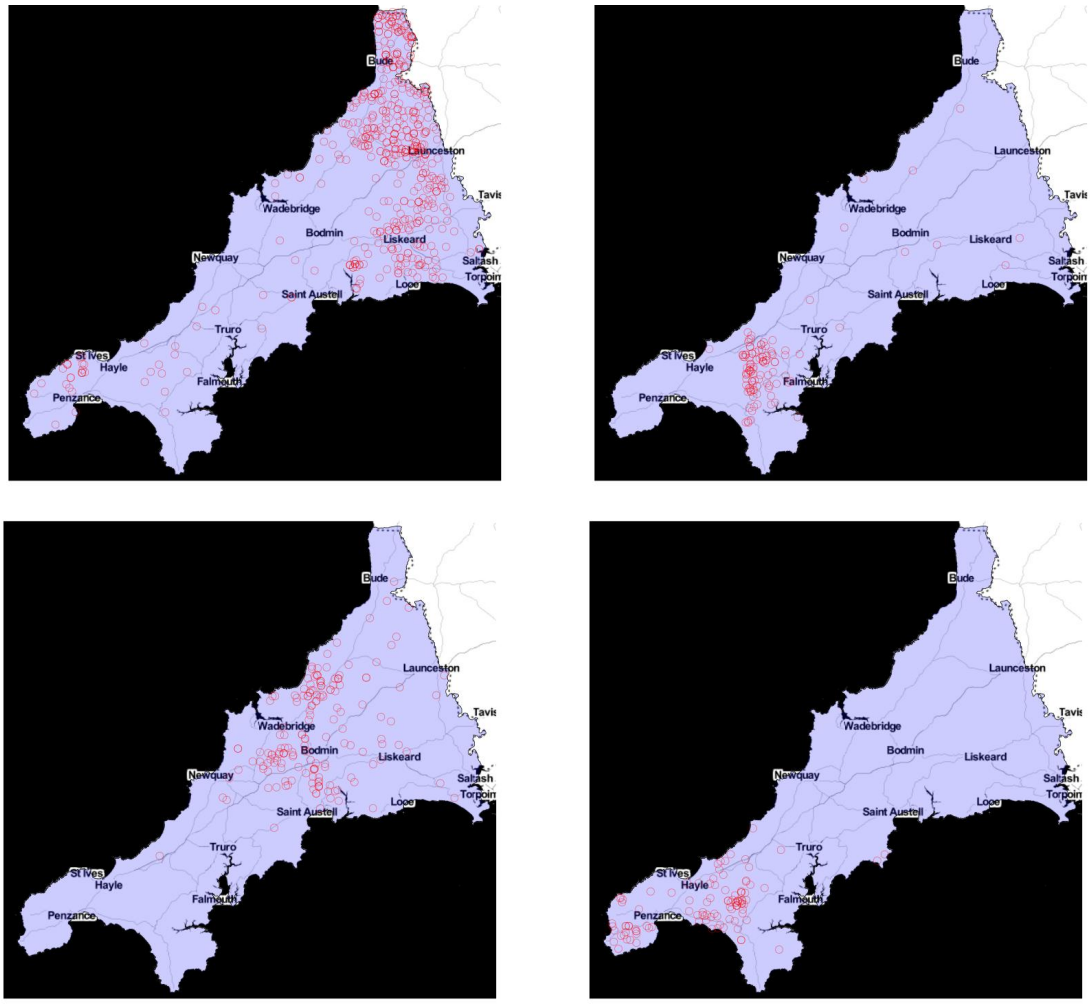

Figure 17: Maps showing cases of bovine tuberculosis between 1989–2002: genotype 9 (top left), genotype 12 (top right), genotype 15 (bottom left), genotype 20 (bottom right).

In Diggle *et al.* (2013), the authors introduce the idea of using the LGCP as a form of density estimation and provide a multivariate example in which they model cases of bovine tuberculosis (BTB) in Cornwall. BTB is endemic in parts of the United Kingdom, and as part of a national control strategy for the disease, herds undergo regular inspection. If the disease is found in a herd, scientists then attempt to determine the genotype of the tuberculosis bacterium that caused the outbreak.

In the present article, we revisit the BTB data discussed in Diggle, Zheng, and Durr (2005b) and Diggle *et al.* (2013). These data, shown in Figure 17, are locations of breakdowns in herds in the period 1989–2002. In the present article, we will consider the 873 such events corresponding to the four most common genotypes (there were a further 46 cases classified as one of six additional genotypes). It is of interest to scientists to determine whether there is segregation in the spatial distribution of BTB, since this is informative about potential transmission mechanisms of the disease.

Analysis of a multitype dataset

We begin by loading the point pattern dataset, a marked `ppp` object. Here, we extend the analysis presented in Diggle *et al.* (2013) by including covariate information to help explain some of the spatial variation in cases of BTB.

```
R> load("BTBppp.RData")
R> load("farmspdf.RData")
```

The covariates in the object `farmspdf`, loaded above, are a proxy for cattle density in Cornwall derived from the June Census in 2010. We would ideally like to have used the true cattle densities for this example, but these data are not available to us. The June census provides animal counts on 5 kilometre grid squares; for more common animal types, these counts are further divided into subsets. For cattle, as well as the total number of animals, there are counts on a further 11 age/sex/useage subsets. Since some of these 11 divisions are highly correlated, we used the `findCorrelation` function from the package **caret** to select a subset of these variables so as to reduce pairwise correlations between the putative covariates and hence improve identifiability.

```
R> library("caret")
R> d <- farmspdf@data[, 30:40]
R> d <- d[!is.na(d[, 1]), ]
R> findCorrelation(cor(d))

[1]  1  5  4  2 11  3  6
```

The resulting vector gives the column indices to be excluded: 1–6 and 11. The following variables were chosen for the main analysis:

- `K207` Male cattle >2 years old.
- `K208` Female beef cattle > 2 years old, no offspring.
- `K209` Female dairy cattle > 2 years old, no offspring.
- `K210` Female beef cattle > 2 years old, with offspring.

We begin according to Algorithm 1 by computing approximate values of the parameters of the process. The function `minimum.contrast` also handles multivariate data on a type-by-type basis (i.e., ignoring correlation between types). Because some of the genotypes in the original data have a low number of counts, `minimum.contrast` fails for these types, we therefore run the approximate estimation procedure on the subset of data we are interested in:

```
R> W <- pppdata$window
R> simpW <- simplify.owin(W, 1000)
R> ttx <- pppdata$x[pppdata$marks == "x1" | pppdata$marks == "x4" |
+    pppdata$marks == "x5" | pppdata$marks == "x7"]
R> tty <- pppdata$y[pppdata$marks == "x1" | pppdata$marks == "x4" |
+    pppdata$marks == "x5" | pppdata$marks == "x7"]
```

```
R> ttm <- as.character(pppdata$marks[pppdata$marks == "x1" |
+    pppdata$marks == "x4" | pppdata$marks == "x5" | pppdata$marks == "x7"])
R> tempppp <- ppp(x = ttx, y = tty, window = simpW, marks = as.factor(ttm))
R> denls <- lapply(as.character(levels(tempppp$marks)), function(x) {
+    density.ppp(tempppp[tempppp$marks == x]) })
R> mn <- minimum.contrast(tempppp,model = "exponential", method = "g",
+    intens = denls, transform = log)

$estimates
       scale variance
x4 1118.5076 1.558636
x1  446.3812 1.742495
x7 1938.9902 1.226359
x5 2207.9740 2.131337

$discrepancy
   Squared discrepancy
x4            1465.281
x1           43508.423
x7            2635.801
x5           10467.575
```

In the above, we used `simplify.owin` to speed up the computation time for the inhomogeneous pair correlation function, which depends on how complex the observation window is.

The function `minimum.contrast` is not able to identify parameters from a type-specific random component, Y_k, and the cross-type random component, Y_{K+1}. Therefore the approximate estimates obtained refer to the stochastic process $Y_k + Y_{K+1}$. In the case that the correlation function is exponential and $Y_k \perp\!\!\!\perp Y_{K+1}$, we would have,

$$\begin{aligned} \mathsf{Var}[Y_k + Y_{K+1}] &= \mathsf{Var}[Y_k] + \mathsf{Var}[Y_{K+1}], \\ &= \sigma_k^2 \exp\{-d/\phi_k\} + \sigma_{K+1}^2 \exp\{-d/\phi_{K+1}\}, \\ &= 2\sigma^2 \exp\{-d/\phi\}, \end{aligned}$$

under the assumption that $\sigma = \sigma_k = \sigma_{K+1}$ and $\phi = \phi_k = \phi_{K+1}$ and where d is the Euclidean distance. We do not know *a priori* that this is the correct thing to do, but it does provide a pragmatic way of understanding which cell widths and initial values might be appropriate for the MCMC run. The above implies that as a rule of thumb, we *could* think of the variance of Y_k (or Y_{K+1}) to be approximately $1/2$ the estimated variance and the scale parameter to be approximately equal to the estimated scale.

In our analyses in this section, we use the following settings:

```
R> CELLWIDTH <- 1800
R> EXT <- 2
```

This is just sufficient for three of the processes, based on the approximate estimates of the scale parameters. Since minimum contrast methods tend to underestimate the true parameter

values (Davies and Taylor 2014), for reasons of computational cost we opted for a cell width of 1,800 metres instead of one of 900 metres. This was a risk in our analysis, and had the posterior showed a strong preference for a small scale parameter in the `x5` data stream, then we would have had to re-run it on a finer computational grid.

We next compute an overlay of the computational grid onto the `SpatialPixelsDataFrame` containing the animal counts:

```
R> polyolay <- getpolyol(data = pppdata, pixelcovariates = farmspdf,
+    cellwidth = CELLWIDTH, ext = EXT)
```

We specify a model for the main effects in each of the data streams as below:

```
R> fl <- formulaList(list(
+    x1 ~ K207 + K208 + K209 + K210,
+    x4 ~ K207 + K208 + K209 + K210,
+    x5 ~ K207 + K208 + K209 + K210,
+    x7 ~ K207 + K208 + K209 + K210))
```

Note that each of these models can comprise different sets of terms. We next set the interpolation type for each of our covariates. Since these are population counts, we chose the `ArealWeightedSum` interpolation to be the most appropriate.

```
R> farmspdf@data <- guessinterp(farmspdf@data)
R> farmspdf@data <- assigninterp(df = farmspdf@data,
+    vars = c("K207", "K208", "K209", "K210"), value = "ArealWeightedSum")
```

In our multivariate algorithm, in order to perform the interpolation step we must first collect together the set of unique independent variables from the list `fl` and create a new main effects model, in this case,

```
R> FORM <- X ~ K207 + K208 + K209 + K210
R> Zmat <- getZmat(formula = FORM, data = pppdata, cellwidth = CELLWIDTH,
+    pixelcovariates = farmspdf, ext = EXT, overl = polyolay)
R> for(x in c("K207", "K208", "K209", "K210")) {
+    Zmat[, x] <- log(Zmat[, x])
+    Zmat[, x][is.infinite(Zmat[, x])] <- min(
+      Zmat[, x"][!is.infinite(Zmat[, x])])
+  }
```

The purpose of the object `Zmat` in this case is purely to store information on the spatial covariates (regardless of the type of point with which they are associated): the internal routines of the package **lgcp** will construct appropriate design matrices, Z_k, for each of the types. Note that we again transform the animal covariates to the log scale since these are population counts.

We next set up priors for this analysis as follows:

```
R> pr.mn <- log(c(1, 1500))
R> pr.vr <- c(0.2, 0.05)
```

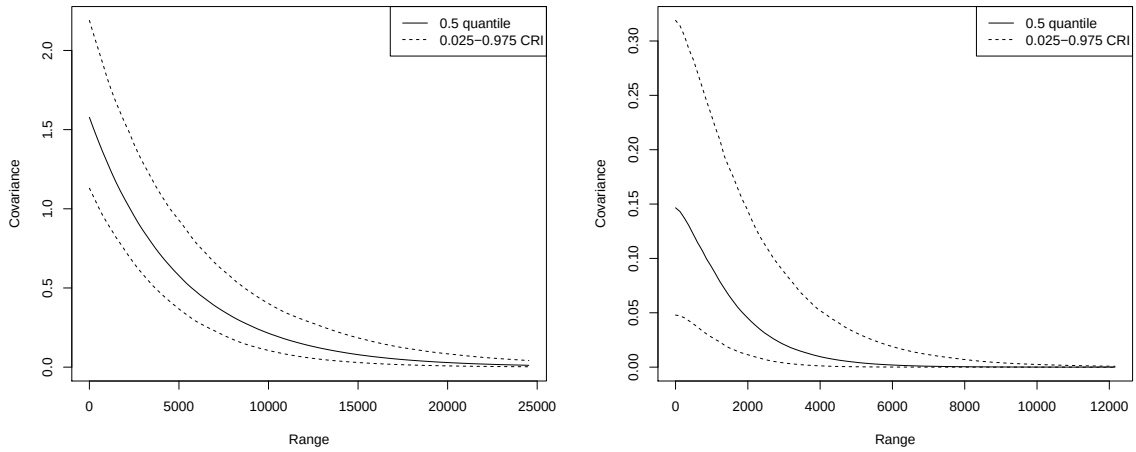

Figure 18: Plot of the posterior estimated covariance function of Y_1 (left) and Y_5 (right) for the bovine TB data.

```
R> priors <- list()
R> for(i in 1:4) {
+    priors[[i]] <- lgcpPrior(etaprior = PriorSpec(
+        LogGaussianPrior(mean = pr.mn, variance = diag(pr.vr))),
+      betaprior = PriorSpec(GaussianPrior(mean = rep(0,5),
+        variance = diag(10^6,5))))
+  }
R> priors[[5]] <- lgcpPrior(etaprior = PriorSpec(
+      LogGaussianPrior(mean = pr.mn, variance = diag(pr.vr))),
+    betaprior = NULL)
```

Notice that in the prior for Y_{K+1}, no prior is specified for β. As for previous examples, the package **lgcp** provides multivariate Gaussian priors for β and η (the latter on the log-scale), but in this case the structure of the prior variance is block diagonal: each pair of parameters, $\eta_k = \{\log \sigma_k, \log \phi_k\}$, has its own multivariate prior independent of the other types. In this example, we will use the prior mean to initialise the MCMC algorithm.

The last remaining task before we run the MCMC algorithm is to choose a covariance function for each of the processes $Y_1, \ldots, Y_{K+1}$:

```
R> cfs <- list()
R> for(i in 1:4) cfs[[i]] <- CovFunction(exponentialCovFct)
R> cfs[[5]] <- CovFunction(RandomFieldsCovFct(model = "matern",
+    additionalparameters = 1))
```

This defines an exponential covariance function for $Y_1, \ldots, Y_4$ and a Matérn covariance function for $Y_{K+1} \equiv Y_5$ with differentiability parameter, `nu`, set to 1. The choice of the Matérn covariance function was for illustrative purposes, see Appendix C for further details. The MCMC routine can now be run:

```
R> BASEDR <- getwd()
R> lg <- lgcpPredictMultitypeSpatialPlusPars(formulaList = fl, sd = pppdata,
```

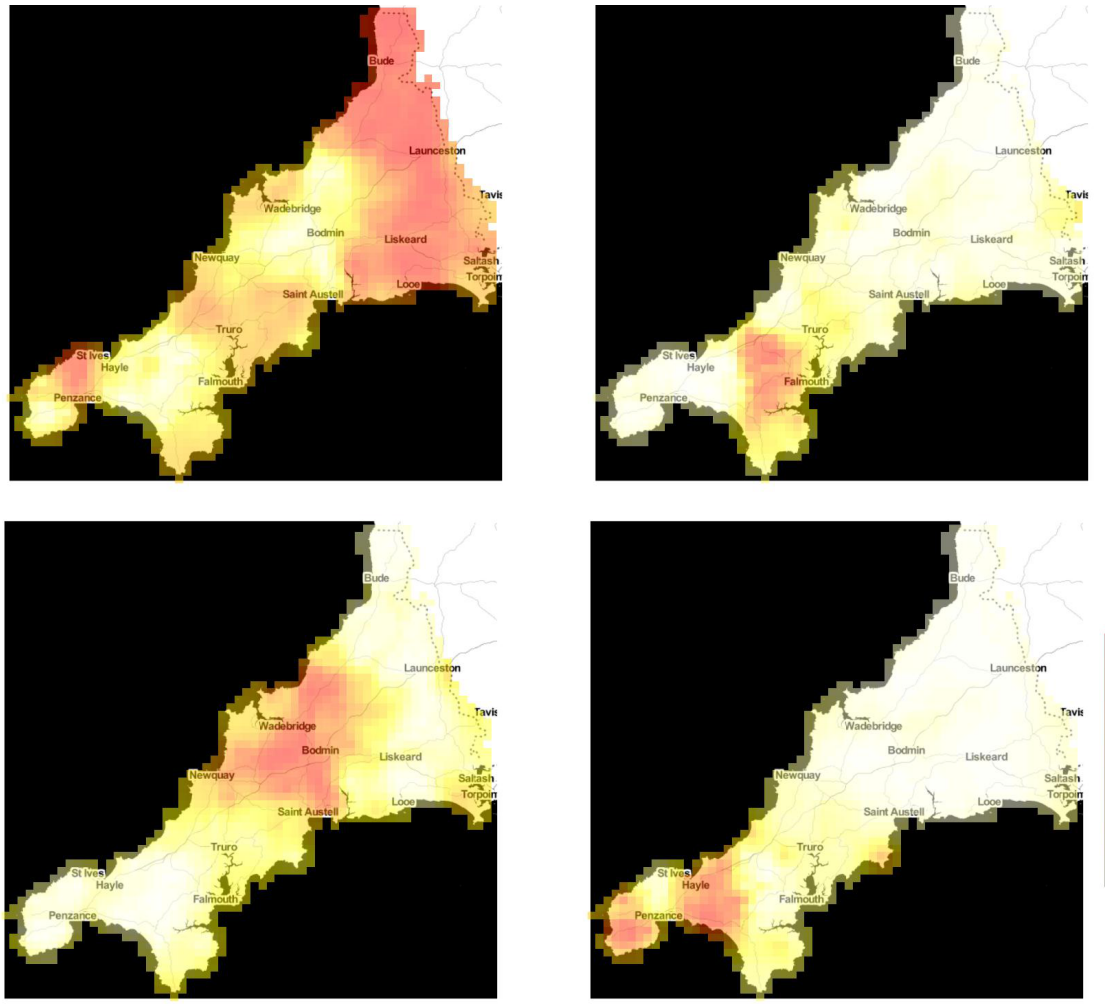

Figure 19: Maps showing the conditional probability that a point at each location is of a particular type: genotype 9 (top left), genotype 12 (top right), genotype 15 (bottom left), genotype 20 (bottom right). The code for this figure appears in the replication materials.

```
+     Zmat = Zmat, model.priorsList = priors, spatial.covmodelList = cfs,
+     cellwidth = CELLWIDTH, mcmc.control = mcmcpars(mala.length = 1000000,
+       burnin = 100000, retain = 900, adaptivescheme = andrieuthomsh(inith = 1,
+         alpha = 0.5, C = 1 ,targetacceptance = 0.574)),
+     output.control = setoutput(gridfunction = dump2dir(
+       dirname = file.path(BASEDR, "BTB"), forceSave = TRUE)),
+     ext = EXT)
R> save(list = ls(), file = file.path(BASEDR, "BTB", "BTB.RData"))
```

As in the other analyses, we first examine convergence and mixing diagnostics; this proceeds in the same way as before, with the exception of the autocorrelation of the latent fields, Y_i, which is achieved using the `autocorrMultitype` function as follows

```
R> for(i in 1:4) {
```

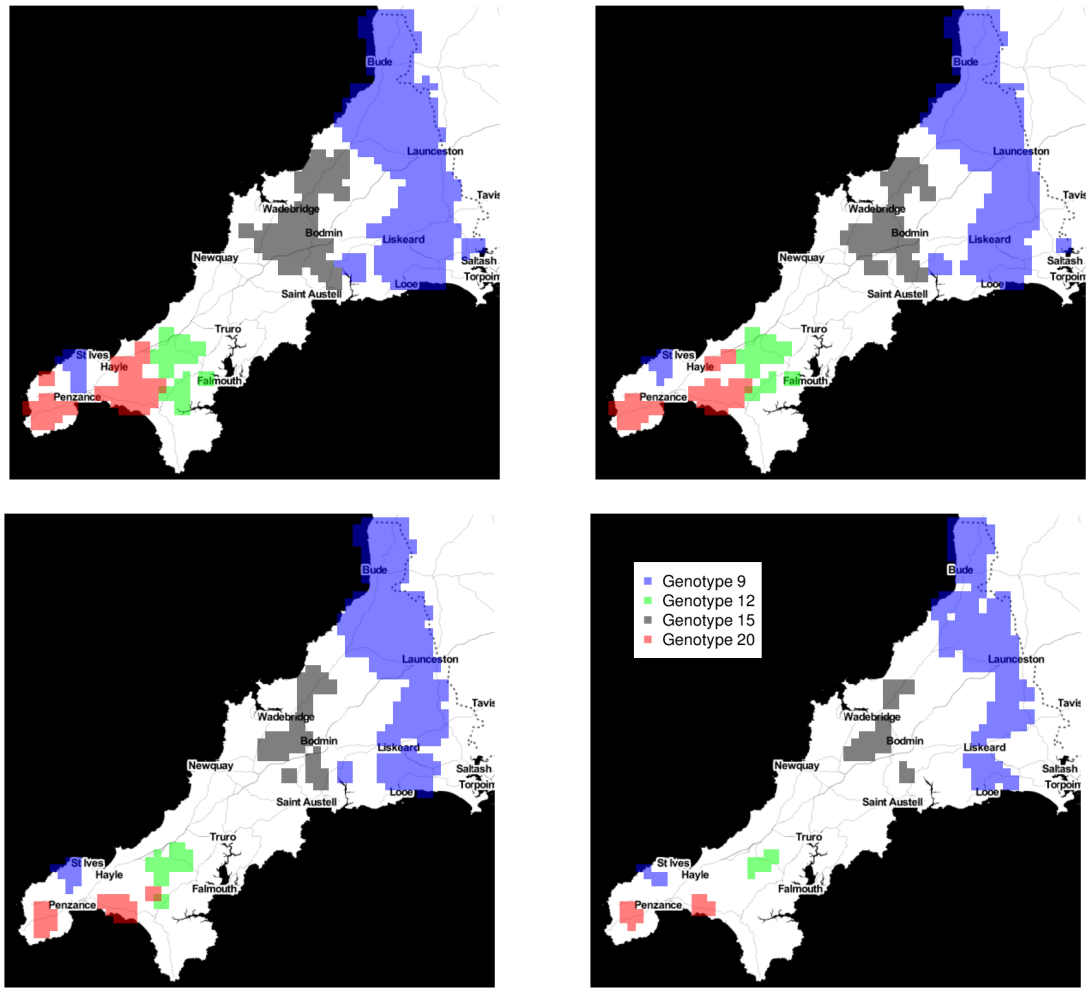

Figure 20: Illustrating the areas $A_k(0.8, q)$ for each of $q = 0.6$ (top left), 0.7 (top right), 0.8 (bottom left) and 0.9 (bottom right). The code for this figure appears in the replication materials.

```
+    Yi <- autocorrMultitype(lg, lags = c(1, 5, 15), fieldno = i,
+      inWindow = NULL)
+    plot(Yi, zlim = c(-1, 1), xlab = "", ylab = "")
+  }
```

The argument `fieldno` specifies the Y_i on which to perform computation. The command `postcov(lg)` plots the posterior covariance function of each Y_i in turn; in Figure 18, we show the plot for fields 1 and 5.

The package **lgcp** has a function to compute the conditional probability that at a particular location there will be an event of type k, which we denote p_k. This function requires `dumpdir` to have been set in the call to the MCMC routine:

```
R> cp <- condProbs(lg)
```

Parameter	Median	Lower 95% CRI	Upper 95% CRI
σ_1	1.257	1.073	1.485
ϕ_1	5056	3829	6620
σ_2	1.641	1.257	2.257
ϕ_2	3408	2387	4840
σ_3	1.484	1.19	1.82
ϕ_3	4546	3311	6339
σ_4	1.979	1.572	2.506
ϕ_4	4851	3757	6387
σ_5	0.3619	0.2031	0.5438
ϕ_5	1592	997.4	2485
$\exp[\beta(1)((Intercept))]$	1.178×10^{-8}	4.907×10^{-9}	2.693×10^{-8}
$\exp[\beta(1)(K207)]$	1.027	0.968	1.085
$\exp[\beta(1)(K208)]$	1.033	0.9193	1.155
$\exp[\beta(1)(K209)]$	1.023	0.9912	1.059
$\exp[\beta(1)(K210)]$	1.016	0.992	1.041
$\exp[\beta(2)((Intercept))]$	2.298×10^{-9}	5.422×10^{-10}	9.345×10^{-9}
$\exp[\beta(2)(K207)]$	0.7945	0.6654	0.925
$\exp[\beta(2)(K208)]$	1.547	1.173	2.07
$\exp[\beta(2)(K209)]$	1.019	0.955	1.086
$\exp[\beta(2)(K210)]$	0.9719	0.9154	1.031
$\exp[\beta(3)((Intercept))]$	8.327×10^{-9}	2.723×10^{-9}	2.344×10^{-8}
$\exp[\beta(3)(K207)]$	1.029	0.9554	1.107
$\exp[\beta(3)(K208)]$	1.011	0.8417	1.209
$\exp[\beta(3)(K209)]$	0.9875	0.9413	1.039
$\exp[\beta(3)(K210)]$	1.014	0.971	1.058
$\exp[\beta(4)((Intercept))]$	3.825×10^{-9}	9.023×10^{-10}	1.455×10^{-8}
$\exp[\beta(4)(K207)]$	1.039	0.9381	1.154
$\exp[\beta(4)(K208)]$	0.9265	0.7057	1.202
$\exp[\beta(4)(K209)]$	1.074	1.008	1.145
$\exp[\beta(4)(K210)]$	1.012	0.9468	1.078

Table 4: Table of parameter estimates from the BTB analysis.

The result, `cp`, is an object of class `lgcpgrid`, which can be converted to an array object using `as.array(cp)`, or a raster object using `raster(cp)`. The latter is particularly useful for adding the results of analyses to background maps, where a change of projection is required. An example of what can be produced is shown in Figure 19.

Similarly, it is of interest to scientists to be able to illustrate spatial regions where a genotype dominates *a posteriori*. Let $A_k(c,q)$ denote the set of locations x for which $\mathsf{P}\{p_k(x) > c|X\} > q$. As the quantities c and q tend to 1 each area $A_k(c,p)$ shrinks towards the empty set; this happens more slowly in a highly segregated pattern compared with a weakly segregated one. We compute $\mathsf{P}\{p_k(x) > c|X\}$ using the following code:

```
R> sr <- segProbs(lg, domprob = 0.8)
```

Again, the result is an `lgcpgrid`. In Figure 20, the areas $A_k(0.8,q)$ are illustrated for each of

$q = 0.6, 0.7, 0.8$ and 0.9. A table of parameter estimates from this model is given in Table 4.

The main points of interest from the main effects are as follows. For types 1 and 3 (respectively genotypes 9 and 15), none of the covariate effects are statistically significant. For type 2 (genotype 12), the rate of cases are significantly lower areas with larger numbers of Male cattle over 2 years old. For type 4 (genotype 20), the rate of cases are significantly higher in areas with larger numbers of Female dairy cattle over 2 years old with no offspring. We emphasise that these results are illustrative and should not be interpreted scientifically, since our covariate information was collected much later than the cases were observed.

5. Discussion

In this article, we have introduced four statistical models based around the log-Gaussian Cox process: a spatial point process model, an aggregated count model, a spatiotemporal point process model and a multivariate point process model. Implementing Bayesian inference for each of these LGCP models is extremely challenging, both from a coding perspective and from the perspective of algorithm heuristics. However, our open-source algorithms and data structures, together with the walk-through examples in this article, provide practitioners with a good basis to start using MCMC for Bayesian spatial and spatiotemporal modelling.

Although it takes patience and experience to get to grips with MCMC, we would argue that this is time worth spending. In an epidemiology study that might last for many years, it is surely justifiable that a statistical analysis might take of the order of weeks: especially when the results of such an analysis can lead to a better understanding of the phenomenon of interest. Computational time is the main perceived disadvantage of MCMC, but in compensation for this, we are able to work in a transparent modelling framework, provide joint inference for all model parameters and add additional hierarchies to our models with only modest effort; furthermore, the convergence and mixing diagnostics offer insight into the statistical problem we are addressing.

If we were to re-run the two liver examples, we would likely exclude `propmale` and `employment`: we note again that these examples are purely to illustrate the code. Principled model selection can be achieved with reversible jump MCMC (Green 1995) but this is likely to be challenging due to having to calibrate cross-model jumps. As a practical solution, we would recommend the user attempts model selection via other means before running the MCMC, for example with a simple call to `glm`, ignoring spatial correlation.

Acknowledgements

This work was supported by MRC research grant reference G0902153, "Statistical modelling for real-time spatial surveillance and forecasting", Peter J. Diggle and Barry S. Rowlingson.

The IMD, LSOA and farm livestock data in this article are subject to the Open Government License: http://www.nationalarchives.gov.uk/doc/open-government-licence/.

Contains National Statistics data: Crown copyright and database right 2001, 2004, 2007.

Contains Ordnance Survey data: Crown copyright and database right 2001, 2004, 2007.

The data in Section 4.3 used in this article was from project AEGISS (Diggle *et al.* 2005a). AEGISS was supported by a grant from the Food Standards Agency, UK, and from the

National Health Service Executive Research and Knowledge Management Directorate.

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

References

Baddeley A, Turner R (2005). "**spatstat**: An R Package for Analyzing Spatial Point Patterns." *Journal of Statistical Software*, **12**(6), 1–42. URL http://www.jstatsoft.org/v12/i06/.

Beale L, Abellan J, Hodgson S, Jarup L (2008). "Methodologic Issues and Approaches to Spatial Epidemiology." *Environmental Health Perspectives*, **116**(8), 1105–1110.

Bernardo JM, Smith AFM (2008). *Bayesian Theory*. John Wiley & Sons.

Besag J, York J, Mollié A (1991). "Bayesian Image Restoration, With Two Applications in Spatial Statistics." *Annals of the Institute of Statistical Mathematics*, **43**(1), 1–20.

Bivand R, Rundel C (2013). ***rgeos**: Interface to Geometry Engine Open Source (GEOS)*. R package version 0.2-17, URL http://CRAN.R-project.org/package=rgeos.

Bivand RS, Pebesma E, Gómez-Rubio V (2013). *Applied Spatial Data Analysis with R*. 2nd edition. Springer-Verlag. URL http://www.asdar-book.org/.

Brix A, Diggle PJ (2001). "Spatiotemporal Prediction for Log-Gaussian Cox Processes." *Journal of the Royal Statistical Society B*, **63**(4), 823–841.

Brix A, Diggle PJ (2003). "Corrigendum: Spatiotemporal Prediction for Log-Gaussian Cox Processes." *Journal of the Royal Statistical Society B*, **65**(4), 946.

Brown PE (2015). "Model-Based Geostatistics the Easy Way." *Journal of Statistical Software*, **63**(12), 1–24. URL http://www.jstatsoft.org/v63/i12/.

Davies TM, Bryant D (2013). "On Circulant Embedding for Gaussian Random Fields in R." *Journal of Statistical Software*, **55**(9). URL http://www.jstatsoft.org/v55/i09/.

Davies TM, Hazelton ML (2013). "Assessing Minimum Contrast Parameter Estimation for Spatial and Spatiotemporal Log-Gaussian Cox Processes." *Statistica Neerlandica*, **67**(4), 355–389.

Davies TM, Taylor BM (2014). "A Systematic Comparison of Second-Order Parameter Estimation Techniques for the Stationary Log-Gaussian Cox Process." Submitted for publication.

Diggle P, Guan Y, Hart A, Paize F, Stanton M (2010). "Estimating Individual-Level Risk in Spatial Epidemiology Using Spatially Aggregated Information on the Population at Risk." *Journal of the American Statistical Association*, **105**(492), 1394–1402.

Diggle P, Ribeiro P (2003). *Model-Based Geostatistics*. Springer-Verlag.

Diggle P, Rowlingson B, Su T (2005a). "Point Process Methodology for On-line Spatio-Temporal Disease Surveillance." *Environmetrics*, **16**(5), 423–434.

Diggle P, Zheng P, Durr P (2005b). "Nonparametric Estimation of Spatial Segregation in a Multivariate Point Process: Bovine Tuberculosis in Cornwall, UK." *Journal of the Royal Statistical Society C*, **54**(3), 645–658.

Diggle PJ, Knorr-Held L, Rowlingson B, Su T, Hawtin P, Bryant T (2003). *Monitoring the Health of Populations: Statistical Methods for Public Health Surveillance*, chapter Towards On-line Spatial Surveillance. Oxford University Press.

Diggle PJ, Moraga P, Rowlingson B, Taylor BM (2013). "Spatial and Spatio-Temporal Log-Gaussian Cox Processes: Extending the Geostatistical Paradigm." *Statistical Science*, **28**(4), 542–563.

Fellows I (2013). ***OpenStreetMap**: Access to Open Street Map Raster Images*. R package version 0.3.1, URL http://CRAN.R-project.org/package=OpenStreetMap.

Gamerman D, Lopes HF (2006). *Markov Chain Monte Carlo: Stochastic Simulation for Bayesian Inference*. 2nd edition. Chpman & Hall/CRC.

Gilks WR, Richardson S, Spiegelhalter D (eds.) (1995). *Markov Chain Monte Carlo in Practice*. Chapman & Hall/CRC.

Girolami M, Calderhead B (2011). "Riemann Manifold Langevin and Hamiltonian Monte Carlo Methods." *Journal of the Royal Statistical Society B*, **73**(2), 123–214.

Green P (1995). "Reversible Jump Markov Chain Monte Carlo Computation and Bayesian Model Determination." *Biometrika*, **82**, 711–732.

Guan Y, Sherman M (2007). "On Least Squares Fitting for Stationary Spatial Point Processes." *Journal of the Royal Statistical Society B*, **69**, 31–49.

Haran M, Tierney L (2012). "On Automating Markov Chain Monte Carlo for a Class of Spatial Models." URL http://arXiv.org/abs/1205.0499/.

Hastings WK (1970). "Monte Carlo Sampling Methods Using Markov Chains and Their Applications." *Biometrika*, **57**(1), 97–109.

Hijmans RJ, van Etten J (2013). ***raster**: Geographic Data Analysis and Modeling*. R package version 2.1-25, URL http://CRAN.R-project.org/package=raster.

Illian JB, Sorbye SH, Rue H (2012a). "A Toolbox for Fitting Complex Spatial Point Process Models Using Integrated Nested Laplace Approximation (**INLA**)." *The Annals of Applied Statistics*, **6**(4), 1499–1530.

Illian JB, Sorbye SH, Rue H, Hendrichsen D (2012b). "Using **INLA** to Fit a Complex Point Process Model with Temporally Varying Effects – A Case Study." *Journal of Environmental Statistics*, **3**(7).

Kelsall J, Wakefield J (2002). "Modeling Spatial Variation in Disease Risk: A Geostatistical Approach." *Journal of the American Statistical Association*, **97**(459), 692–701.

Li Y, Brown P, Gesink DC, Rue H (2012). "Log-Gaussian Cox Processes and Spatially Aggregated Disease Incidence Data." *Statistical Methods in Medical Research*, **21**(5), 479 – 507.

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.

Metropolis N, Rosenbluth AW, Rosenbluth MN, Teller AH, Teller E (1953). "Equation of State Calculations by Fast Computing Machines." *Journal of Chemical Physics*, **21**(6), 1087–1092.

Møller J, Syversveen AR, Waagepetersen RP (1998). "Log-Gaussian Cox Processes." *Scandinavian Journal of Statistics*, **25**(3), 451–482.

Neal RM (2011). "MCMC Using Hamiltonian Dynamics." In S Books, A Gelman, G Jones, XL Meng (eds.), *Handbook of Markov Chain Monte Carlo*, pp. 113–162. Chapman & Hall/CRC.

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/.

Pierce D (2014). "**ncdf**: Interface to Unidata netCDF Data Files." R package version 1.6.8, URL http://CRAN.R-project.org/package=ncdf.

Prince MI, Chetwynd A, Diggle P, Jarner M, Metcalf JV, James OFW (2001). "The Geographical Distribution of Primary Biliary Cirrhosis in a Well-Defined Cohort." *Hepatology*, **34**(6), 1083–1088.

R Core Team (2014). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.

Ripley BD (1977). "Modelling Spatial Patterns." *Journal of the Royal Statistical Society B*, **39**, 172–212.

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, Menck PJ, Oesting M, Strokorb K (2015). "Analysis, Simulation and Prediction of Multivariate Random Fields with Package **RandomFields**." *Journal of Statistical Software*, **63**(8), 1–25. URL http://www.jstatsoft.org/v63/i08/.

Simpson D, Illian J, Lindgren F, Sorbye S, Rue H (2011). "Going off Grid: Computationally Efficient Inference for Log-Gaussian Cox Processes." Preprint Statistics No. 10/2011.

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/.

Taylor BM, Diggle PJ (2013). "Corrigendum: Spatiotemporal Prediction for Log-Gaussian Cox Processes." *Journal of the Royal Statistical Society B*, **75**(3), 601–602.

Taylor BM, Diggle PJ (2014). "INLA or MCMC? A Tutorial and Comparative Evaluation for Spatial Prediction in Log-Gaussian Cox Processes." *Journal of Statistical Computation and Simulation*, **84**(10), 2266–2284.

van Dyk D, Meng XL (2001). "The Art of Data Augmentation." *Journal of Computational and Graphical Statistics*, **10**, 1–111.

Wakefield J, Haneuse S, Dobra A, Teeple E (2011). "Bayes Computation for Ecological Inference." *Statistics in Medicine*, **30**(12), 1381–1396.

Wakefield J, Lyons H (2010). *Handbook of Spatial Statistics: Spatial Aggregation and the Ecological Fallacy*, chapter 30, pp. 541–558. Chapman and Hall.

Wall MM (2004). "A Close Look at the Spatial Structure Implied by the CAR and SAR Models." *Journal of Statistical Planning and Inference*, **121**, 311–324.

Wood ATA, Chan G (1994). "Simulation of Stationary Gaussian Processes in $[0, 1]^d$." *Journal of Computational and Graphical Statistics*, **3**(4), 409–432.

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

A. Interpolation of covariate information

The role of the interpolation step is to put covariate information onto the computational grid. This information may come from an object that is more or less spatially refined than the computational grid. Here, we assume that the covariate information is on a finer scale than the computational grid, so that information in a computational grid cell is comprised of multiple pieces of information from the covariate dataset.

The three types of interpolation methods employed in the package **lgcp** are:

- `Majority` The interpolated value corresponds to the value of the covariate occupying the largest area of the computational cell.
- `ArealWeightedMean` The interpolated value corresponds to the mean of all covariate values contributing to the computational cell weighted by their respective areas.
- `ArealWeightedSum` The interpolated value is the sum of all contributing covariates weighed by the proportion of area with respect to the covariate polygons. For example, suppose region A has the same area as a computational grid cell and has 500 inhabitants. If that region occupies half of a computational grid cell, then this interpolation type assigns 250 inhabitants from A to the computational grid cell.

B. Poisson offsets in lgcp

In the package **lgcp**, the user specifies a Poisson offset by setting $\lambda(s)$, or $\lambda(s,t)$ directly (rather than the log of those quantities, as would be the case in a call to `glm` with `family = "poisson"` for example).

We specify offsets using the `SpatialAtRisk` class of objects. Note that this object class was originally designed so that $\int_W \lambda(s)\mathrm{d}s = 1$ over the observation window, W. However, since the normalising constant of this operation is stored, the class can also be used to define Poisson offsets.

The example below can be constructed using data from the example in Section 4.1. In this example, we set up an offset on the computational grid with values corresponding to the population counts as measured by the census data. We interpolate the census counts onto the computational grid, then use an object of class `SpatialAtRisk.fromXYZ` to encode the information.

```
R> FORM_pop <- X ~ pop - 1
R> Zmat_pop <- getZmat(formula = FORM_pop, data = sd,
+    regionalcovariates = popshape, cellwidth = CELLWIDTH,
+    ext = EXT, overl = polyolay)
R> mm <- length(attr(Zmat_pop, "mcens"))
R> nn <- length(attr(Zmat_pop, "ncens"))
R> poisson.offset <- spatialAtRisk(list(X = attr(Zmat_pop, "mcens"),
+    Y = attr(Zmat_pop, "ncens"), Zm = matrix(Zmat_pop, mm, nn)))
```

It is always best to construct the offset directly onto the computational grid: the x-values of the grid are in this case `attr(Zmat_pop, "mcens")` and the y values are `attr(Zmat_pop, "ncens")`.

In the case of a spatiotemporal or multivariate model, the user must supply a list of such `SpatialAtRisk` offsets.

C. More choices of covariance function

In Section 4.4 we saw the following example of code:

```
R> cfs[[5]] <- CovFunction(RandomFieldsCovFct(model = "matern",
+    additionalparameters = 1))
```

used to specify a Matérn covariance function. In fact, the function `RandomFieldsCovFct` gives the user access to all covariance models from the **RandomFields** package in the function `CovarianceFct` (see the help file). The user simply has to choose the argument `model` from the available models and in the call to `RandomFieldsCovFct` above and set the values of any additional parameters for the family of choice in the order they appear in the documentation of `CovarianceFct`.

D. More complex expectations

For spatial processes, including point process and aggregate process models, the package **lgcp** enables the user to compute joint expectations over all model parameters using the function `lgcp:::expectation.lgcpPredictSpatialOnlyPlusParameters`. In the example below, we examine the function `numCases`, a function with parameters `Y`, `beta`, `eta`, `Z` and `otherargs`.

```
R> numCases

function (Y, beta, eta, Z, otherargs){
   ca <- diff(otherargs$mcens[1:2]) * diff(otherargs$ncens[1:2])
   return(ca * otherargs$poisson.offset[1:otherargs$M, 1:otherargs$N] *
      exp(matrix(Z %*% t(beta), otherargs$M * otherargs$ext,
      otherargs$N * otherargs$ext)[1:otherargs$M, 1:otherargs$N] + Y))
}

R> ex <- expectation(lg, numCases)[[1]]
```

The function `numCases` returns the expected number of cases in each computational grid cell from one iteration of the algorithm,

$$C_A \lambda(s) \exp\{Z(s)\beta + Y(s)\};$$

the object `ex` is a 128×64 grid with the expected number of events in each cell. Note that whilst `Y`, `beta`, `eta` and `Z` have obvious definitions, the argument `otherargs` is not so obvious. In fact, this will take the output prediction object `lg` as input, so any information returned in `lg` can be used in constructing these expectations.

E. MCMC warnings

Sometimes during an MCMC run, the following warning message will be issued:

```
Warning: One possible cause of this warning is that there may
be evidence in the data for quite large values of the spatial
correlation parameter. If this is the case, then this warning
means that the MCMC chain has wandered into a region of the
phi-space that causes the variance matrix of Y (computed by
the DFT) to become non positive-definite. One possible way of
rectifying this issue is to restart the chain using a larger
value of 'ext' in the call to lgcpPredictSpatialPlusPars. You
should do this if the warning message is repeated many times.
If this warning message appears at all then you should be
warned that inference from this run may not be reliable: the
proposed move at this iteration was rejected.
```

To find out on which iterations this message appeared, type `lg$reject_its` at the console (assuming the MCMC object is called `lg`). If this message only appears during the burn-in phase, then it is fine to continue analysing the data as normal, however, if this message appeared during the 'run' phase of the algorithm, then formally the run should be repeated with different settings.

The message appears because the η chain has wandered into an area for which the covariance matrix as computed by the FFT is not positive definite. As the message suggests, one way of rectifying this is by increasing the value of `ext` used in the call to the MCMC algorithm; this incurs computational expense.

An alternative is to tighten the prior on ϕ to discourage moves into regions which lead to instability. In practice, we have found that when the parameter `ext` is set to 2, priors for ϕ that do not place too much probability mass outside the range of approximately 1/5 of the width of the observation window are sufficiently tight to avoid this warning messsage.

Short pilot runs of 1,000–5,000 iterations are useful to discover issues like this before committing serious computation time.

Affiliation:

Benjamin M. Taylor, Barry Rowlingson, Peter J. Diggle
Faculty of Health and Medicine
Lancaster University
Lancaster, LA1 4YF, United Kingdom
E-mail: b.taylor1@lancaster.ac.uk,
b.rowlingson@lancaster.ac.uk,
p.diggle@lancaster.ac.uk
URL: http://www.maths.lancs.ac.uk/~taylorb1/,
http://www.maths.lancs.ac.uk/~rowlings/,
http://www.lancs.ac.uk/~diggle/

Tilman M. Davies
Department of Mathematics and Statistics
University of Otago
Science III, PO Box 56
Dunedin 9054, New Zealand
E-mail: tdavies@maths.otago.ac.nz
URL: http://tinyurl.com/tilman-davies/

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume 63, Issue 7 *Submitted:* 2013-06-02
January 2015 *Accepted:* 2014-08-25