Journal of Statistical Software

January 2015, Volume 63, Issue 11. *http://www.jstatsoft.org/*

geoCount: An R Package for the Analysis of Geostatistical Count Data

Liang Jing
Quant Tech

Victor De Oliveira
The University of Texas at San Antonio

Abstract

We describe the R package **geoCount** for the analysis of geostatistical count data. The package performs Bayesian analysis for the Poisson-lognormal and binomial-logitnormal spatial models, which are subclasses of the class of generalized linear spatial models proposed by Diggle, Tawn, and Moyeed (1998). The package implements the computational intensive tasks in C++ using an R/C++ interface, and has parallel computation capabilities to speed up the computations. **geoCount** also implements group updating, Langevin-Hastings algorithms and a data-based parameterization, algorithmic approaches proposed by Christensen, Roberts, and Sköld (2006) to improve the efficiency of the Markov chain Monte Carlo algorithms. In addition, the package includes functions for simulation and visualization, as well as three geostatistical count datasets taken from the literature. One of those is used to illustrate the package capabilities. Finally, we provide a side-by-side comparison between **geoCount** and the R packages **geoRglm** and **INLA**.

Keywords: Bayesian inference, geostatistics, hierarchical models, kriging, Markov chain Monte Carlo, parallel computing.

1. Introduction

Spatial data are nowadays routinely collected and analyzed in numerous scientific disciplines such as ecology, epidemiology, forestry, geography and meteorology to name a few. Following the classification of spatial data in Cressie (1993), we consider in this work geostatistical (also called point-referenced) data that contain information about both location and an attribute of interest. The basic data structure consists of pairs $\{(\boldsymbol{x}_i, Y_i) : i = 1, \ldots, n\}$, where $\boldsymbol{x}_i$ are the coordinates of the i-th sampling location within some region of interest $A \subset \mathbb{R}^2$, and Y_i is the observation taken at $\boldsymbol{x}_i$. Each observed value Y_i is either a direct measurement of, or statistically related to, a spatially varying attribute of interest at $\boldsymbol{x}_i$, where the latter will be modeled by transforming an underlying continuous spatial process.

Spatial datasets often display non-Gaussian features when either the distribution of the attribute of interest or the response variable are non-Gaussian, and this is the case for spatial count data. A useful and popular class of models is that of *generalized linear spatial models* (GLSM), first proposed by Diggle, Tawn, and Moyeed (1998); see also Diggle and Ribeiro (2007). These models assume that the spatially varying attribute of interest is functionally related to a realization of a Gaussian random field. Due to the likelihood complexity of GLSM, numerical algorithms are needed to fit them. Among these, Markov chain Monte Carlo (MCMC) algorithms have been proposed and used, but their practical implementation faces several numerical and statistical challenges. The most notable are:

(1) High computational efforts and memory needs due to repeated inversion of $n \times n$ matrices, which is an $O(n^3)$ hard problem. This is an intrinsic problem in most algorithms that perform likelihood-based inference for geostatistical data.

(2) Some of the previously proposed MCMC algorithms to fit GLSM display very slow mixing and high auto- and cross-correlations among the parameters and latent values. This is the case for the algorithms proposed by Diggle, Tawn, and Moyeed (1998) and Christensen and Waagepetersen (2002), as shown by Christensen, Roberts, and Sköld (2006) and Jing (2011); see also Section 7.1.

(3) Some of the previously proposed MCMC algorithms fail to converge in some cases, when the datasets consist mostly of either large or small counts. This is the case for the algorithms proposed by Diggle, Tawn, and Moyeed (1998), as shown by Christensen, Roberts, and Sköld (2006) and Jing (2011).

The package **geoCount** for the R system for statistical computing (R Core Team 2014) ameliorates the above problems by combining programming and algorithmic strategies. Two programming strategies are used to ameliorate problem (1). First, the heavy computational tasks are written in C++ and use C++ libraries **Armadillo** (Sanderson, Curtin, Cullinan, Bouzas, and Funiak 2014) and GNU scientific library (**GSL**, The **GSL** Team 2013) to perform linear algebra and other scientific/statistical computations. The R packages **Rcpp** and **RcppArmadillo** (Eddelbuettel and François 2011; Eddelbuettel and Sanderson 2014) are used to provide seamless integration/communication between R and C++. Second, parallel programming is used to generate several independent chains with different starting values, which are merged after their convergence is assessed. The R packages **snow** (Rossini, Tierney, and Li 2007) and **snowfall** (Knaus 2013) are used to implement the parallel computations.

Three algorithmic strategies are used to ameliorate problems (2) and (3): Group updating, Langevin-Hastings MCMC algorithms and a data-based parameterization. These strategies were proposed by Christensen, Roberts, and Sköld (2006), who showed increased efficiency of the resulting MCMC algorithms by improving mixing and convergence, and reducing the auto- and cross-correlations among parameters and latent values. These findings were explored and confirmed by Jing (2011). In this article we overview the above strategies and describe how they are implemented in the R package **geoCount**.

The organization of the paper is as follows. Sections 2 and 3 describe, respectively, the sampling models and priors. Sections 4 and 5 describe, respectively, the programing and algorithmic strategies used by the package for the Bayesian fitting of the models. Section 6 describes the package capabilities and illustrates them using a spatial dataset of weed counts

collected from an agricultural field in Sweden. Sections 7 and 8 provide comparisons between **geoCount** and, respectively, the R packages **geoRglm** and **INLA**, that can also fit geostatistical count data. Finally, Section 9 provides a summary of the **geoCount** capabilities and limitations.

2. Models

The spatial data to be modeled here consist of triplets $\{(\boldsymbol{x}_i, Y_i, t_i);\ i = 1, \ldots, n\}$, where $\boldsymbol{x}_i$ are the coordinates of the i-th sampling location within the region of interest $A \subset \mathbb{R}^2$, Y_i is the measurement taken at $\boldsymbol{x}_i$, and t_i is a known positive value associated with $\boldsymbol{x}_i$ that depends on the sampling design. There may also be available location-dependent covariates $d_1(\cdot), \ldots, d_p(\cdot)$ (see below). The class of GLSM introduced by Diggle, Tawn, and Moyeed (1998) is based on the following general assumptions:

(A1) The spatially varying attribute of interest, which is unobservable, is related by a one-to-one function to a Gaussian random field with certain parametric mean and covariance functions.

(A2) For any set of locations the observations of the response variable at these locations are conditionally independent given the values of the attribute of interest at these locations. In addition, the response variable at any location is stochastically related to the attribute of interest *only* at that location.

In this section we provide the model definition for the attribute of interest and response at the sampling locations, and defer the model definition at the prediction locations until Section 6.5. Let $\{S(\boldsymbol{x}) : \boldsymbol{x} \in A\}$ be the Gaussian random field that is functionally related to the spatially varying attribute of interest, and $\boldsymbol{S} = (S(\boldsymbol{x}_1), \ldots, S(\boldsymbol{x}_n))^\top$. Although the latter is not observable, it is assumed that each observed value Y_i is stochastically related to the attribute of interest at $\boldsymbol{x}_i$. The general class of GLSM introduced by Diggle, Tawn, and Moyeed (1998) is hierarchically specified as follows:

$$\begin{aligned} Y_i \mid S(\boldsymbol{x}_i) &\sim p(\cdot \mid \mu_i), \qquad i = 1, \ldots, n \\ \mu_i &= t_i g^{-1}(S(\boldsymbol{x}_i)) \\ \boldsymbol{S} &\sim \mathrm{N}_n(D\boldsymbol{\beta}, \boldsymbol{\Sigma}) \\ (\boldsymbol{\beta}, \boldsymbol{\theta}) &\sim \pi(\boldsymbol{\beta}, \boldsymbol{\theta}), \end{aligned} \tag{1}$$

where:

- $\{Y_i : i = 1, \ldots, n\}$ are conditionally independent given $\boldsymbol{S}$, and have marginal pdfs/pmfs $p(\cdot \mid \mu_i)$.
- $\mu_i = E(Y_i \mid S(\boldsymbol{x}_i))$ and $g(\cdot)$ is a known one-to-one link function.
- $D = (\mathbf{1}, \boldsymbol{d}_1, \ldots, \boldsymbol{d}_p)$ is a known $n \times (p+1)$ design matrix, assumed of full rank, with $\mathbf{1}$ the $n \times 1$ vector of ones and $\boldsymbol{d}_j = (d_j(\boldsymbol{x}_1), \ldots, d_j(\boldsymbol{x}_n))^\top$, where $d_j(\boldsymbol{x}_i)$ is the value of the j-th covariate at the i-th sampling location, and $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_p)^\top$ are unknown regression parameters.

- $\boldsymbol{\Sigma} = (\sigma_{ij})$ is an $n \times n$ positive definite variance-covariance matrix with $\sigma_{ij} = \sigma^2 \rho(u_{ij})$, where $\sigma^2 > 0$ is the unknown constant variance, $\rho(u_{ij})$ is a parametric isotropic correlation function, and $u_{ij} = ||\boldsymbol{x}_i - \boldsymbol{x}_j||$ is Euclidean distance.
- $\pi(\cdot)$ is the prior distribution of the model parameters $(\boldsymbol{\beta}, \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ contains the covariance parameters that specify $\boldsymbol{\Sigma}$, i.e., σ^2 and the parameters appearing in $\rho(\cdot)$.

Three families of correlation functions are implemented in **geoCount**: the *spherical* family given by

$$\rho(u) = \left(1 - \frac{3}{2}\left(\frac{u}{\phi}\right) + \frac{1}{2}\left(\frac{u}{\phi}\right)^3\right) \mathbf{1}_{[0,\phi]}(u), \qquad u \geq 0, \quad \phi > 0,$$

where $\mathbf{1}_A(u)$ is the indicator function of the set A; the *Matérn* family given by

$$\rho(u) = \frac{1}{2^{\kappa-1}\Gamma(\kappa)}\left(\frac{u}{\phi}\right)^{\kappa} K_{\kappa}\left(\frac{u}{\phi}\right), \qquad u \geq 0, \quad \phi > 0, \quad \kappa > 0, \tag{2}$$

where $\Gamma(\cdot)$ is the gamma function and $K_{\kappa}(\cdot)$ is a modified Bessel function of order κ; and the *power exponential* family given by

$$\rho(u) = \exp\left\{-\left(\frac{u}{\phi}\right)^{\kappa}\right\}, \qquad u \geq 0, \quad \phi > 0, \quad \kappa \in (0, 2]. \tag{3}$$

For all of the above families the so-called range parameter ϕ controls the rate of correlation decay, and for the latter two families the so-called smoothness parameter κ controls smoothness of the random field $S(\cdot)$.

Although the above general framework can be used to model a variety of non-Gaussian spatial data, it has been mostly used to model spatial count data. The two most widely used GLSM for spatial count data are the *Poisson-lognormal* and *binomial-logitnormal* spatial models; see Diggle, Tawn, and Moyeed (1998), Christensen and Waagepetersen (2002), and Zhang (2002). For the first model, the top two levels of the hierarchical model state that

$$\begin{aligned} Y_i \mid S(\boldsymbol{x}_i) &\sim \text{Poisson}(\mu_i), \qquad i = 1, \ldots, n \\ \mu_i &= t_i \exp(S(\boldsymbol{x}_i)), \end{aligned} \tag{4}$$

while for the second model

$$\begin{aligned} Y_i \mid S(\boldsymbol{x}_i) &\sim \text{Binomial}(t_i, \mu_i/t_i), \qquad i = 1, \ldots, n \\ \mu_i &= t_i \frac{\exp(S(\boldsymbol{x}_i))}{1 + \exp(S(\boldsymbol{x}_i))}; \end{aligned} \tag{5}$$

the bottom two levels of the above models are both equal to the corresponding levels in the general model (1). The above models use the parametrization in Christensen, Roberts, and Sköld (2006), which is slightly different from the one used in Diggle, Tawn, and Moyeed (1998). The R package **geoCount** has been written to make different kinds of Bayesian statistical inferences about these two GLSM. Among other tasks, it simulates samples from the posterior distribution of the model parameters and latent values at the sampling locations, and simulates samples from the posterior predictive distribution of the latent variables and potential counts at unsampled locations.

3. Priors

Specification of prior distributions for the parameters of these GLSM is a relatively unexplored area. Previous works have mostly glossed over this aspect of the model construction and have used priors chosen somewhat ad hoc and in analogy to priors used for hierarchical and non-hierarchical Gaussian models. In the present absence of a better alternative the implementation in **geoCount** follows the same approach and uses some default "non-informative" priors. The parameters $\boldsymbol{\beta}$, σ, ϕ and κ are assumed independent a priori, with marginal distributions given by

$$\begin{aligned} \pi(\boldsymbol{\beta}) \propto 1 \;\text{ for }\; \boldsymbol{\beta} \in \mathbb{R}^{p+1} \quad &, \quad \pi(\sigma) \;\text{ for }\; \sigma > 0 \;\text{ (see below)} \\ \pi(\phi) \propto \mathbf{1}_{[b_1,b_2]}(\phi) \quad &, \quad \pi(\kappa) \propto \mathbf{1}_{[c_1,c_2]}(\kappa). \end{aligned}$$

geoCount provides two options for $\pi(\sigma)$: the 'half-t' family proposed by Gelman (2006)

$$\pi^{HT}(\sigma) \propto \Big(1 + \frac{1}{a_2}\Big(\frac{\sigma}{a_1}\Big)^2\Big)^{-(a_2+1)/2},$$

where a_1, a_2 are user-defined hyper-parameters, which includes the (improper) uniform distribution ($a_2 = -1$) and the (proper) half-Cauchy distribution ($a_2 = 1$); and the inverse gamma family, $\pi^{IG}(\sigma) \propto \sigma^{-(a_1+1)} \exp\big(-a_2/\sigma\big)$, which includes the (improper) reciprocal distribution $\pi(\sigma) \propto 1/\sigma$ ($a_1 = a_2 = 0$). The latter option yields an improper posterior distribution though; see Natarajan and Kass (2000). The user-defined hyper-parameters b_1 and b_2 that determine the support of the range parameter ϕ can be chosen either subjectively or based on exploratory data analysis, e.g., by inspecting empirical semivariogram plots. On the other hand, the values that determine the support of the smoothness parameter κ are fixed in **geoCount**: $(c_1, c_2) = (0.05, 4.95)$ for the Matérn family, and $(c_1, c_2) = (0.05, 1.95)$ for the power exponential family.

4. Programming strategies

4.1. R and C++

Implementation of MCMC algorithms can be computationally very intensive for some models. This is the case for the algorithms we consider for GLSM because of the large number of latent variables to be sampled, and the handling and computation of large matrices. Both computations are very time-consuming for R to handle solely, and demand a lower level and faster language. Our choice for this is C++ which is one of the most popular programming languages with both low-level and high-level language features. It provides fast computation as well as good portability and extendibility.

In **geoCount** the programs for the implementation of MCMC algorithms and computation of large matrices are written in C++, with the help of **GSL** and **Armadillo** libraries. The former is a numerical library for C and C++ which provides a wide range of mathematical routines, such as random number generators and computation of special functions. The latter is an open-source C++ linear algebra library (matrix maths) which supports common data types and a subset of trigonometric and statistical functions.

For efficient communication between R and C++ we employ **Rcpp** API developed by Eddelbuettel and François (2011). In this new API, `Rcpp::RObject` is the basic class which actually encapsulates `SEXP` objects with a thin wrapper and comes with methods that are appropriate for all types of objects and transparently manage garbage collection. In fact the `SEXP` is indeed the only data member of an `RObject`. `RObject` manages the life cycle of underlying `SEXP` wisely. The constructor of `RObject` takes necessary measures to guarantee the protection from garbage collection during C++ scope, and the destructor takes the responsibility to withdraw that protection. It also defines several member functions common to all objects (e.g., isS4(), attributeNames, ...) and the derived classes define specific member functions.

4.2. Parallel computing in R

There are many tasks for which parallel computation can be quite useful for the analysis of geostatistical data with GLSM. Some of these are

- Simultaneous simulation of several datasets.
- Simultaneous posterior simulation of several Markov chains.
- Generation of replicated data.
- Simultaneous prediction at several locations.

Multi-processor and multi-core computers, either in the form of personal computers (PC) or high performance computing (HPC) clusters, have recently become much more accessible. So it is desirable, and sometimes even necessary, to optimize the computations in our package with the help of parallel computation techniques. R will only utilize one processor under the default build, regardless of the number of available processors. So we employ several R packages to solve this shortcoming, including **snow** by Rossini, Tierney, and Li (2007) and **snowfall** by Knaus (2013).

5. Algorithmic strategies

This section describes three algorithmic strategies advocated and developed by Christensen, Roberts, and Sköld (2006) that result in efficient and robust MCMC algorithms; these were implemented in R by Jing (2011). In what follows, $\boldsymbol{Y} = (Y_1, \ldots, Y_n)^\top$, $\boldsymbol{S} = (S(\boldsymbol{x}_1), \ldots, S(\boldsymbol{x}_n))^\top$, $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_p)^\top$ and $\boldsymbol{\theta} = (\sigma, \phi, \kappa)^\top$.

5.1. Group updating

It is by now known that when performing Gibbs or Metropolis-Hastings sampling, grouping usually leads to faster convergence rates, specially when the variables to be simulated are highly correlated. This is the case in GLSM where the components of $\boldsymbol{S}$ and $\boldsymbol{\theta}$ are often highly correlated a posteriori. In addition, grouping usually reduces the overall computational work in high dimensional settings, which is also the case in GLSM since there are n latent variables. The package **geoCount** groups the latent variables and parameters in three groups: $\boldsymbol{S}$, $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$, where the components of the first two groups are updated jointly while the components of the last group are updated individually.

5.2. Langevin-Hastings algorithm

Metropolis-Hastings algorithms that use random walk proposals are the most commonly used algorithms due to the ease of implementation in many problems, but they often suffer from slow mixing and convergence issues. This is the case for the algorithm used by Diggle, Tawn, and Moyeed (1998), as shown by Christensen, Roberts, and Sköld (2006) and Jing (2011). This behavior is due to the fact that the chain moves from the current point following a proposal distribution that completely ignores the information in the target distribution. In contrast, Langevin-Hastings algorithms use local information of the target density to make proposals, which can be significantly more efficient, specially in high dimensional problems. Some properties of these algorithms for performing simulation-based Bayesian inference were investigated by Roberts and Tweedie (1996) and Roberts and Rosenthal (1998), while the application of Langevin-Hastings algorithms for fitting GLSM appeared in Christensen and Waagepetersen (2002), Christensen, Roberts, and Sköld (2006) and Jing (2011). It was shown in Roberts and Rosenthal (1998) that when a proper scale is chosen for the proposal, algorithms that use Langevin proposals take on average $O(r^{1/3})$ steps to converge, with r the dimension of the vector being updated. This compares quite favorably to the $O(r)$ steps that take algorithms that use random walk proposals to converge, and the benefit of Langevin-Hastings algorithms increase with the dimension r. This is specially relevant for the current models where the possibly high-dimensional vector $\boldsymbol{S}$ (for which $r = n$) is one of the groups to be jointly updated.

5.3. Data-based parameterization

It is by now well recognized that convergence of MCMC algorithms may crucially depend on the choice of parameterization; see Roberts and Sahu (1997), Papaspiliopoulos, Roberts, and Sköld (2003) and Papaspiliopoulos, Roberts, and Sköld (2007). In GLSM the components of $\boldsymbol{S}$ and $\boldsymbol{\theta}$ are strongly dependent a priori, and depending on the observed value of $\boldsymbol{Y}$, $\boldsymbol{S}$ and $\boldsymbol{\theta}$ may also be strongly dependent a posteriori, which will reduce the efficiency of MCMC algorithms. Papaspiliopoulos, Roberts, and Sköld (2007) described two basic types of parameterizations in hierarchical models, called centered and non-centered parameterizations, and showed that none of them is uniformly better than the other. These parameterizations are complementary in the sense that when an MCMC algorithm displays slow convergence under one parameterization, it often converges much faster under the other parameterization. But which one occurs for a particular dataset depends on how informative the observed value of $\boldsymbol{Y}$ is about $\boldsymbol{S}$. As a compromise Papaspiliopoulos, Roberts, and Sköld (2007) proposed the use of data-based parameterizations aimed at reducing the dependence among $\boldsymbol{S}$ and $\boldsymbol{\theta}$ a posteriori, and argued that MCMC algorithms based on these parameterizations would display good convergence behavior regardless of the observed data. In this sense, these parameterizations and the MCMC algorithms based on them are considered *robust*.

Based on this idea, Christensen, Roberts, and Sköld (2006) proposed a data-based parameterization in GLSM aimed at making the components of $(\boldsymbol{S}, \boldsymbol{\beta}, \boldsymbol{\theta})$ approximately uncorrelated a posteriori, with equal means and equal variances. The latter is recommended since the efficiency of Langevin-Hastings algorithms is sensitive to variance inhomogeneities of the components, a situation expected to occur due to different meanings of $\boldsymbol{S}, \boldsymbol{\beta}$ and $\boldsymbol{\theta}$.

Usually it is not possible to find explicitly the desired parameterization, unless the posterior distribution is multivariate Gaussian. Based on a $N(\boldsymbol{\xi}, \Omega)$ prior for $\boldsymbol{\beta}$, Christensen,

Roberts, and Sköld (2006) proposed the use of a Gaussian approximation to the distribution of $(\boldsymbol{S}, \boldsymbol{\beta} \mid \boldsymbol{\theta}, \boldsymbol{y})$ to orthogonalize and standardize these components, with $\boldsymbol{y}$ being the observed data. This data-based parameterization, denoted by $(\tilde{\boldsymbol{S}}, \tilde{\boldsymbol{\beta}}, \tilde{\boldsymbol{\theta}})$, is given by

$$\begin{aligned} \tilde{\boldsymbol{S}} &= \tilde{\boldsymbol{S}}(\boldsymbol{S}, \boldsymbol{\beta}, \boldsymbol{\theta}, \boldsymbol{y}) \\ &= (\tilde{\Sigma}^{1/2})^{-1}(\boldsymbol{S} - \tilde{\Sigma}(\Lambda(\hat{\boldsymbol{S}})\hat{\boldsymbol{S}} + \Sigma^{-1}D\boldsymbol{\beta})) \qquad (6) \\ \tilde{\boldsymbol{\beta}} &= \tilde{\boldsymbol{\beta}}(\boldsymbol{\beta}, \boldsymbol{\theta}, \boldsymbol{y}) \\ &= (\tilde{\Omega}^{1/2})^{-1}(\boldsymbol{\beta} - \tilde{\Omega}(D^\top\Sigma^{-1}\tilde{\Sigma}\Lambda(\hat{\boldsymbol{S}})\hat{\boldsymbol{S}} + \Omega^{-1}\boldsymbol{\xi})), \qquad (7) \end{aligned}$$

with

$$\begin{aligned} \tilde{\Sigma} &= (\Sigma^{-1} + \Lambda(\hat{\boldsymbol{S}}))^{-1} \\ \tilde{\Omega} &= (\Omega^{-1} + D^\top(\Lambda(\hat{\boldsymbol{S}})\Sigma + I_n)^{-1}\Lambda(\hat{\boldsymbol{S}})D)^{-1}, \end{aligned}$$

where $\tilde{\Sigma}^{1/2}$ and $\tilde{\Omega}^{1/2}$ are the Cholesky decompositions of $\tilde{\Sigma}$ and $\tilde{\Omega}$, respectively; when $\pi(\boldsymbol{\beta}) \propto 1$ is used, the terms Ω^{-1} and $\Omega^{-1}\boldsymbol{\xi}$ are set to zero. Also, $\Lambda(\boldsymbol{S})$ is the diagonal matrix with diagonal entries $-\frac{\partial^2}{\partial S_i^2}\log p(y_i \mid S(\boldsymbol{x}_i))$, $i = 1, \ldots, n$, and $\hat{\boldsymbol{S}}$ is a typical value of $\boldsymbol{S}$ (see below). By construction the components of $\tilde{\boldsymbol{S}}$ and $\tilde{\boldsymbol{\beta}}$ will be, a posteriori, approximately uncorrelated with mean 0 and variance 1.

Christensen, Roberts, and Sköld (2006) suggested using $\hat{S}_i = \text{argmax}\{p(y_i \mid S(\boldsymbol{x}_i))\}$, which is implemented in **geoCount**. For the Poisson-lognormal spatial model, $\hat{S}_i = \log(y_i/t_i)$, so $\Lambda(\hat{\boldsymbol{S}})_{ii} = y_i$, while for the binomial-logitnormal spatial model $\hat{S}_i = \text{logit}(y_i/t_i)$, so $\Lambda(\hat{\boldsymbol{S}})_{ii} = y_i(1-y_i/t_i)$. When $y_i = 0$ in the Poisson-lognormal spatial model ($y_i \in \{0, t_i\}$ in the binomial-logitnormal spatial model), both $\Lambda(\hat{\boldsymbol{S}})_{ii}$ and $(\Lambda(\hat{\boldsymbol{S}})\hat{\boldsymbol{S}})_i$ are set to 0, the limit that results when $y_i \to 0$ ($y_i \to 0$ or $y_i \to t_i$). **geoCount** uses a Metropolis-within-Gibbs algorithm to update $\tilde{\boldsymbol{S}}$ and $\tilde{\boldsymbol{\beta}}$ as separate blocks based on a Langevin proposal with scalings chosen by trial and error aimed at obtaining an acceptance rate of about 0.57, which is close to optimal, Roberts and Rosenthal (1998). Christensen, Roberts, and Sköld (2006) reported that using the default scalings $1.65^2/n^{1/3}$ and $1.65^2/p^{1/3}$ for $\tilde{\boldsymbol{S}}$ and $\tilde{\boldsymbol{\beta}}$, respectively, works often reasonably well, which was also found to be so in Jing (2011); see Christensen, Roberts, and Sköld (2006) for details.

For the covariance parameters it has been found that the posterior correlation between ϕ and σ is usually strong, but the technique used above to reparametrize $\boldsymbol{S}$ and $\boldsymbol{\beta}$ is not feasible for $\boldsymbol{\theta}$ due to the complex way in which the covariance parameters enter into the likelihood. Following Christensen, Roberts, and Sköld (2006), **geoCount** uses the parameterization

$$\tilde{\theta}_1 = \log(\sigma), \qquad \tilde{\theta}_2 = \log(\sigma^2/\phi^{2\kappa}), \qquad \tilde{\theta}_3 = \kappa,$$

where these parameters are updated using individual one-dimensional Metropolis-Hastings steps with random walk proposals having scalings chosen by trial and error aimed at obtaining an acceptance rate of about 0.45, which is close to optimal, Gelman, Roberts, and Gilks (1996).

6. Package functionality

6.1. Integrated datasets

The package **geoCount** includes three geostatistical count datasets:

`Rongelap`: This dataset appeared in Diggle, Tawn, and Moyeed (1998) and was modeled using the Poisson-lognormal spatial model. It consists of photon emission counts emitted from radioactive caesium and collected at 157 locations in the Rongelap Island, a part of the Marshall Islands. This dataset is a four-component list: the first component contains the coordinates of the sampling locations, $\boldsymbol{x}_i$, the second contains the measured counts, y_i, the third contains the times (in seconds) over which counts were accumulated, t_i, and the last the coordinates for the Rongelap Island boundary. For further details see Diggle, Harper, and Simon (1997).

`Weed`: This dataset appeared in Guillot, Lorén, and Rudemo (2009) and was modeled using the Poisson-lognormal spatial model. It consists of weed counts of non-crop plants collected over 100 frames all of 0.5 by 0.7 meters at the Bjertorp Farm in the south-west of Sweden. This dataset is a 100 by 4 data frame: the first two columns contain the coordinates of the sampling locations, $\boldsymbol{x}_i$, the third contains the observed counts, y_i, and the last are estimates of the counts obtained from photographs. The areas of the frames over which the counts were collected (the t_is in this case) are all equal, so they can be assumed equal to 1.

`Rhizoc`: This dataset appeared in Zhang (2002) and was modeled using the binomial-logitnormal spatial model. It consists of counts of the root disease *Rhizoctonia* root rot present in barley, and collected at 100 sampling locations in the Cunningham Farm in the north-west of the US. For each sampling location 15 plants were pulled from the ground for inspection. This dataset is a 100 by 5 data frame: the first two columns contain the coordinates of the sampling locations, $\boldsymbol{x}_i$, the third contains the total number of crown roots in the plants pulled (the t_is in this case), the fourth contains the total number of infected crown roots in the plants pulled, y_i, and the last is the barley yields at the sampling locations.

6.2. Data simulation and visualization

The function `simData` simulates geostatistical data from either the Poisson-lognormal or binomial-logitnormal spatial model. An illustration of the former is:

```
R> library("geoCount")
R> loc <- expand.grid(1:10, 1:10)
R> dat <- simData(loc, L = 0, X = NULL, beta = 2, Y.family = "Poisson",
+    rho.family = "rhoPowerExp", cov.par = c(1, 0.2, 1))
```

where `loc` is an n by 2 matrix containing the coordinates of the sampling locations and `L` is a vector of length n containing the t_is; the default value 0 indicates that $t_i = 1$ for all i. For internal computations **geoCount** decomposes the trend part of the model as

$$D\boldsymbol{\beta} = \beta_0 \mathbf{1} + \mathtt{X}\boldsymbol{\beta}_+,$$

where `X` is a $n \times p$ matrix ($p \geq 0$) containing the values of the covariates, if there are any, and $\boldsymbol{\beta}_+ = (\beta_1, \ldots, \beta_p)^\top$. With this convention `X = NULL` indicates that there are no covariates, and `beta` is a vector of length $p + 1$ containing the intercept and the covariate coefficients, if there are any. `Y.family` sets the model type to be simulated, `"Poisson"` or `"Binomial"`.

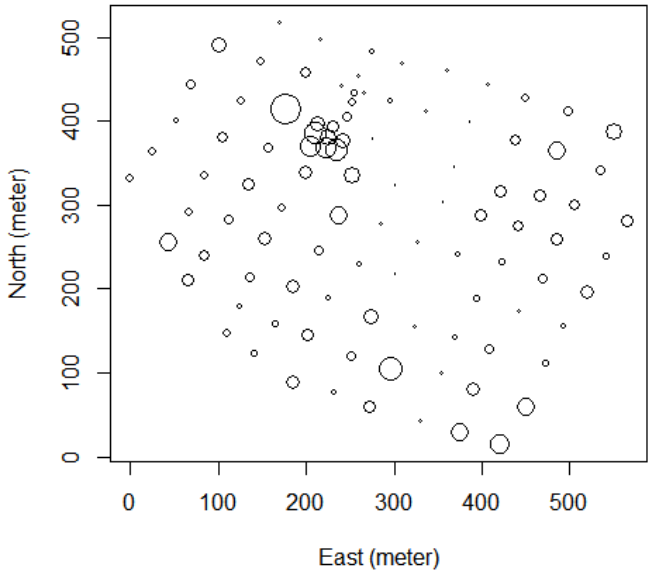

Figure 1: `Weed` locations and counts at the Bjertorp farm in the south-west of Sweden.

The argument `rho.family` sets the family of correlation functions, `"rhoSph"`, `"rhoMatern"` or `"rhoPowerExp"` for, respectively, the spherical, Matérn and power exponential families, and `cov.par` is a vector of length 3 containing the covariance parameters (σ, ϕ, κ).
The output of this function is a list with two vector elements: `data` containing the counts $\boldsymbol{Y}$, and `latent` containing the latent variables $\boldsymbol{S}$.

We will use throughout the paper the `Weed` dataset described above to illustrate the package functionality which will be fit using the Poisson-lognormal spatial model (4) with constant mean and power exponential correlation function (3).

The function `plotData` produces a graphical display of geostatistical data. An illustration using the `Weed` dataset is given in Figure 1:

```
R> data("Weed")
R> plotData(Weed[, 3], Weed[, 1:2], xlab = "East (meter)",
+    ylab = "North (meter)")
```

The "bubbles" are centered at the sampling locations and their diameters are proportional to the magnitude of the observations (this default shape can be changed by setting the `pch` parameter). `plotData` can also display the boundary of the region of interest A, when this information is provided by the two-column matrix argument `bdry`.

The starting point to simulate or analyze geostatistical data is a set of sampling locations. **geoCount** provides three functions that generate "regular" arrangements of sampling locations: `locGrid` generates a rectangular lattice of locations, `locCircle` generates a set of locations on a circle centered at the origin, and `locSquad` generates a set of locations on a square centered at the origin. For instance, `locGrid(a, b, n1, n2)` generates a rectangular lattice on $[0, \mathtt{a}] \times [0, \mathtt{b}]$ having `n1` equally spaced locations per row and `n2` equally spaced locations per column. An illustration on the use of these functions and the display of the datasets

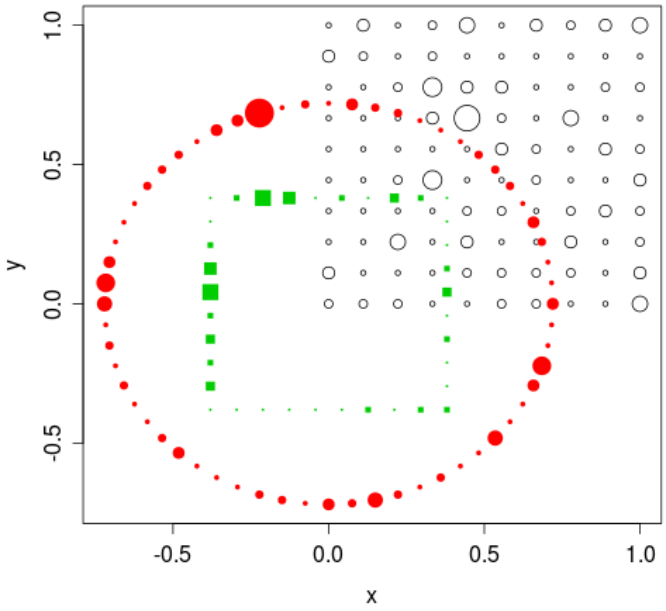

Figure 2: Plot of three geostatistical data sets: `dat1` was simulated at sampling locations `loc1` (black); `dat2` was simulated at `loc2` (red); and `dat3` was simulated at `loc3` (green).

simulated at the respective sets of sampling locations appears in Figure 2. `plotData` can display up to three datasets on a single plot:

```
R> loc1 <- locGrid(1, 1, 10, 10)
R> loc2 <- locCircle(0.72, 60)
R> loc3 <- locSquad(0.38, 10)
R> plotData(dat1$data, loc1, dat2$data, loc2, dat3$data, loc3,
+    xlab = "x", ylab = "y", pchs = c(1, 16, 15))
```

where the datasets `dat1`, `dat2` and `dat3` were simulated from the same model as `dat` above, but using as sampling locations `loc1`, `loc2` and `loc3`, respectively.

6.3. Posterior sampling

The function `MCMCinput` is used to set up the sampling model, prior and environmental settings needed to run the MCMC algorithm. An example for the `Weed` dataset is:

```
R> input.Weed <- MCMCinput(Y.family = "Poisson", rho.family = "rhoPowerExp",
+    run = 2200, run.S = 1, phi.bound = c(10, 300), priorSigma = "Halft",
+    parSigma = c(1, 1), ifkappa = 0, initials = list(c(4), 0.7, 90, 1),
+    scales = c(0.55, 3.5, 0.5, 0.4, 1))
```

where `run` is the number of iterations of the MCMC algorithm and `run.S` is the number of times $\boldsymbol{S}$ is updated as a group within each MCMC iteration to improve accuracy and reduce autocorrelations. Increasing `run.S` usually does not increase dramatically the running

time of the algorithm. `phi.bound` sets the support for the range parameter ϕ, which needs to be chosen with care to avoid deteriorating the accuracy and efficiency of the algorithms. `priorSigma` sets the type of prior for σ, with options `"Halft"` or `"InvGamma"`, and `parSigma` sets its hyper-parameters; see Section 3. `ifkappa` is an indicator of whether the covariance parameter κ must be sampled from its posterior distribution (`ifkappa = 1`) or fixed at a value (`ifkappa = 0`). `initials` sets the starting values for $\boldsymbol{\beta}$, σ, ϕ and κ which can be chosen from an exploratory data analyses, and `scales` sets the scalings of the Langevin and random walk proposals for $\boldsymbol{S}$, $\boldsymbol{\beta}$, σ, ϕ and κ; the last component of the last two arguments are ignored when `ifkappa = 0`. In general the algorithm converges fast, and it can be run a few times with different sets of starting values as a numerical check.

The function `runMCMC` samples from the posterior distribution in Poisson-lognormal and binomial-logitnormal spatial models using MCMC algorithm described in Section 5. Specifically, this function simulates posterior samples for $(\tilde{\boldsymbol{S}}, \tilde{\boldsymbol{\beta}}, \tilde{\boldsymbol{\theta}})$ which are later transformed back to obtain posterior samples for the latent variables $\boldsymbol{S}$ and parameters $(\boldsymbol{\beta}, \boldsymbol{\theta})$ in the original parameterization using (6) and (7). An example for the `Weed` dataset is:

```
R> res.Weed <- runMCMC(loc = Weed[, 1:2], Y = Weed[, 3], L = 0, X = NULL,
+    MCMCinput = input.Weed)
```

where `Y` is the $n \times 1$ vector of observed counts. `runMCMC` has additional arguments that can set the sampling model, prior and environmental settings, but these do not need to be specified or are disregarded when `MCMCinput` is given.
The output from `runMCMC` is a list with the following elements:

- `S.posterior`: An n by `run` matrix containing the posterior sample of $\boldsymbol{S}$.
- `m.posterior`: A $(p+1)$ by `run` matrix (in case there are $p \geq 1$ covariates) or a vector of length `run` (in case there are no covariates), containing the posterior sample of $\boldsymbol{\beta}$.
- `s.posterior`: A vector of length `run` containing the posterior sample of σ.
- `a.posterior`: A vector of length `run` containing the posterior sample of ϕ.
- `k.posterior`: A vector of length `run` containing the posterior sample of κ, in the case `ifkappa = 1`.
- `AccRate`: A vector of length 4 or 5 containing the acceptance rates for $\boldsymbol{S}, \boldsymbol{\beta}, \sigma, \phi$ and κ.

The above function uses one CPU to perform the computational tasks. When several CPUs are available (as in the case of multi-core PCs or high performance computing clusters), the function `runMCMC.sf` performs the robust MCMC algorithms to generate several posterior samples in parallel, each of them of length `run`. This function enables different CPUs to run the `runMCMC` function simultaneously with different initial values, and performs the parallel computation using the packages **snow**, **snowfall**, and **rlecuyer** (Sevcikova and Rossini 2012). The arguments of this function are the same as those in `runMCMC`, plus additional arguments with settings for the parallel computation. An example for the `Weed` dataset is:

```
R> library("snowfall")
R> library("rlecuyer")
```

```
R> res.Weed.p <- runMCMC.sf(loc = Weed[, 1:2], Y = Weed[, 3], L = 0,
+    X = NULL, MCMCinput = input.Weed, n.chn = 4, n.cores = 4,
+    cluster.type = "SOCK")
```

where `n.chn` is the number of chains that want to be simulated in parallel and `n.core` is the number of CPUs used for the computation. `cluster.type` sets the type of cluster used in the parallel computation, with options `"SOCK"`, `"MPI"`, `"PVM"` or `"NWS"`. The output from this function is a list of length `n.chn`, in which each element is itself a list like the output from `runMCMC`.

In practice we recommend to first simulate short pilot chains using `runMCMC` to discover values for the argument `scales` in `MCMCinput` that make the MCMC algorithm efficient (see Section 5.3), and then use these to simulate several longer chains in parallel using `runMCMC.sf`.

6.4. Chain handling and diagnostics

The function `cutChain` takes as input the output of `runMCMC` to perform the desired amount of "burn-in" and "thinning" on the simulated chain. For the `Weed` dataset we have:

```
R> res.Weed <- cutChain(res.Weed, chain.ind = 1:4, burnin = 100,
+    thinning = 10)
```

where `chain.ind` indicates the element numbers in `res.Weed` that are processed. The output of this function is the same as the output of `runMCMC`, except that the element `AccRate` is absent. When parallel chains are available from the output of `runMCMC.sf`, `lapply` can be used to apply `cutChain` to each chain individually:

```
R> res.Weed.p <- lapply(res.Weed.p, cutChain, chain.ind = 1:4,
+    burnin = 200, thinning = 10)
```

These parallel chains need to be examined for mixing and convergence before they are combined to be used for inference. The R package **coda** (Plummer, Best, Cowles, and Vines 2006) is a well-known package for output analysis and diagnostics of MCMC simulations. It provides a variety of functions for visualizing posterior samples and performing convergence tests. In addition, **geoCount** includes the function `plotACF` that plots the auto-correlation curves of the simulated latent variables and the function `findMode` that computes the modes of the estimated marginal posterior densities. An example of the above functions applied to the output `res.Weed.p` appears below and in Figure 3:

```
R> par(mfrow = c(2, 2))
R> phi.est <- c()
R> for (i in 1:4) {
R>   plotACF(res.Weed.p[[i]]$S.posterior)
R>   phi.est[i] <- findMode(res.Weed.p[[i]]$a.posterior)
R> }
```

Finally, once the mixing and convergence of the processed chains have been satisfactorily assessed, the function `mixChain` merges the processed parallel chains into a single chain:

	β_0	σ	ϕ	$\exp(S(\boldsymbol{x}_7))$	$\exp(S(\boldsymbol{x}_{38}))$	$\exp(S(\boldsymbol{x}_{82}))$
2.5%	2.924	0.871	56.578	122.977	8.525	154.779
50%	4.105	1.238	128.547	147.378	14.512	179.468
97.5%	5.581	1.878	284.482	173.296	22.965	207.265

Table 1: Posterior quantiles for the model parameters and three latent variables for the `Weed` dataset.

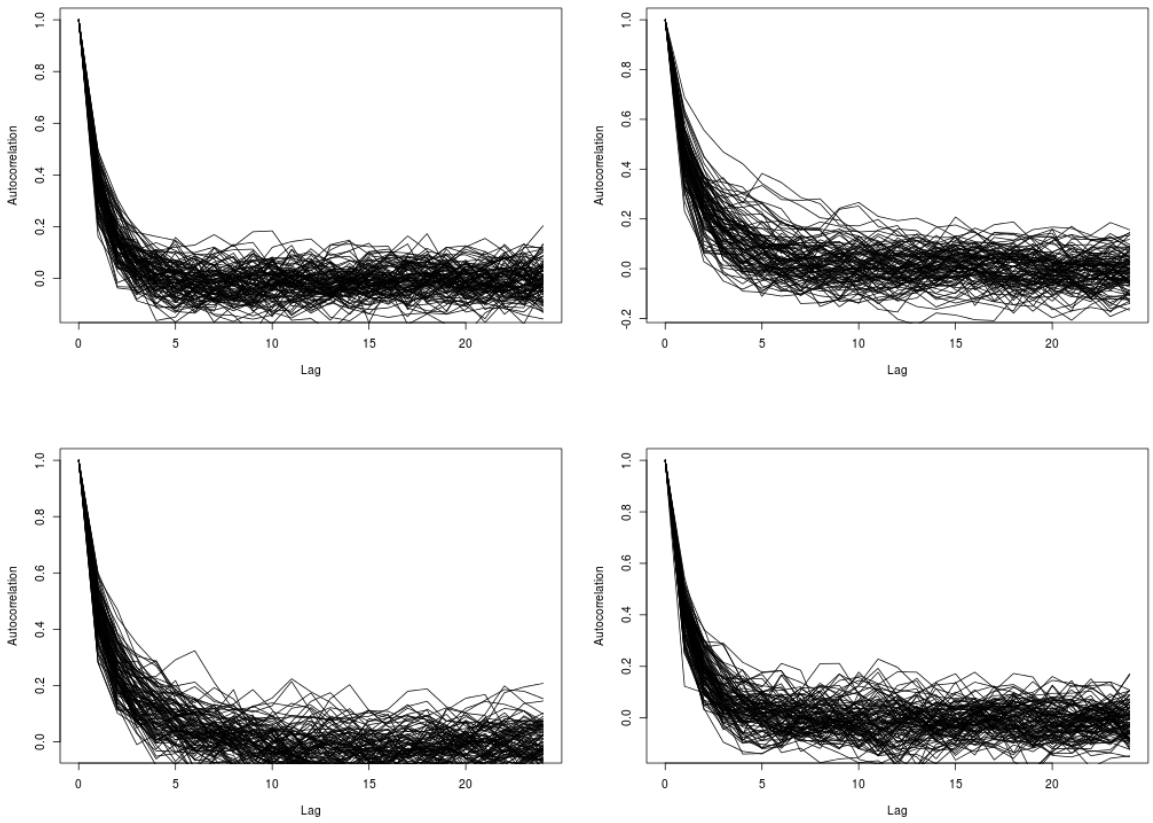

Figure 3: Auto-correlation curves for the posteriors latent variables at the sampling locations from the four simulated chains in `Weed` dataset.

```
R> res.Weed.p <- mixChain(res.Weed.p)
```

Based on this posterior sample inference about any of the model components can be summarized in the usual way. For instance, for the `Weed` dataset we use posterior quantiles to summarize all parameters and three latent variables, which are reported in Table 1:

```
R> quantile(res.Weed.p$m.posterior, probs = c(0.025, 0.5, 0.975))
R> quantile(res.Weed.p$s.posterior, probs = c(0.025, 0.5, 0.975))
R> quantile(res.Weed.p$a.posterior, probs = c(0.025, 0.5, 0.975))
R> apply(exp(res.Weed.p$S.posterior[c(7, 38, 82), ]), 1, quantile,
+    probs = c(0.025, 0.5, 0.975))
```

6.5. Prediction

In this section we describe how **geoCount** performs predictive inference about the spatially varying attribute of interest and potential response counts. Let $\boldsymbol{x}_1^p, \boldsymbol{x}_2^p, \ldots, \boldsymbol{x}_m^p$ be the set of prediction locations where inference about the spatially varying attribute is sought. This is given by $g^{-1}(S(\cdot)) = e^{S(\cdot)}$ (an intensity) for the Poisson-lognormal spatial model (4), and $g^{-1}(S(\cdot)) = e^{S(\cdot)}/(1 + e^{S(\cdot)})$ (a probability) for the binomial-logitnormal spatial model (5). Also, let Y_j^p be the response count that could have been measured (but was not) at $\boldsymbol{x}_j^p$, following the top level of the hierarchy in either model (4) or (5). Before describing how **geoCount** performs predictive inference we first create a regular grid of prediction locations that covers the convex hull of the sampling locations of the `Weed` dataset:

```
R> eastings <- seq(from = min(Weed[, 1]), to = max(Weed[, 1]), length = 30)
R> northings <- seq(from = min(Weed[, 2]), to = max(Weed[, 2]), length = 30)
R> locpp <- expand.grid(eastings, northings)
R> loc <- Weed[, 1:2]
R> d.min <- 40
R> re <- numeric(nrow(locpp))
R> for(i in 1:nrow(locpp)) {
+    t <- locpp[i, ]
+    dist <- min(apply(cbind(t, loc), 1,
+      function(l) { sqrt((l[1] - l[3])^2 + (l[2] - l[4])^2) }))
+    re[i] <- ifelse(dist < d.min, 1, 0)
+  }
R> locp <- locpp[re == 1, ]
```

For prediction location $\boldsymbol{x}_j^p$, the posterior predictive distribution of $(Y_j^p, S(\boldsymbol{x}_j^p))$ is given by

$$\begin{aligned} p(y_j^p, s_j^p \mid \boldsymbol{y}) &= \int\int\int p(y_j^p, s_j^p, \boldsymbol{s}, \boldsymbol{\beta}, \boldsymbol{\theta} \mid \boldsymbol{y}) d\boldsymbol{s} d\boldsymbol{\beta} d\boldsymbol{\theta} \\ &= \int\int\int p(y_j^p \mid s_j^p) p(s_j^p \mid \boldsymbol{s}, \boldsymbol{\beta}, \boldsymbol{\theta}) p(\boldsymbol{s}, \boldsymbol{\beta}, \boldsymbol{\theta} \mid \boldsymbol{y}) d\boldsymbol{s} d\boldsymbol{\beta} d\boldsymbol{\theta}, \end{aligned}$$

where $p(y_j^p \mid s_j^p)$ is given by the top level of the hierarchy in either model (4) or (5), with a conjectured value t_j^p, and $p(s_j^p \mid \boldsymbol{s}, \boldsymbol{\beta}, \boldsymbol{\theta})$ is the normal pdf with parameters obtained from the standard formulas for the conditional mean and variance of Gaussian processes; the absence of some of the conditioning variables in the densities of the integrand above follow from the general assumptions (A1) and (A2) in Section 2. Hence, simulation from this posterior predictive distribution is easily obtained from a sample from the posterior distribution of $(\boldsymbol{S}, \boldsymbol{\beta}, \boldsymbol{\theta})$.

The function `predY` simulates a sample from the posterior predictive distribution of $\boldsymbol{S}^p = (S(\boldsymbol{x}_1^p), \ldots, S(\boldsymbol{x}_m^p))^\top$ and $\boldsymbol{Y}^p = (Y_1^p, \ldots, Y_m^p)^\top$. This function can perform the computations serially or in parallel. An illustration of the latter using the prediction locations `locp` created above is:

```
R> Ypred <- predY(res.m = res.Weed.p, loc = Weed[, 1:2], locp = locp,
+    X = NULL, Xp = NULL, Lp = rep(1, nrow(locp)), k = 1,
+    Y.family = "Poisson", rho.family = "rhoPowerExp",
+    parallel = "snowfall", n.cores = 4, cluster.type = "SOCK")
```

where `run.m` is the output of the function `runMCMC`, or of its parallel counterpart, `locp` is an $m \times 2$ matrix with the coordinates of the prediction locations, `Xp` is an $m \times p$ matrix with the values of the covariates (if there are any) at the prediction locations, and `Lp` is a vector of length m with the conjectured values t_j^ps. The optional argument `k` specifies the value of the covariance parameter κ, when this is assumed fixed, and the argument `parallel` is included when parallel computation is desired; when the latter is absent the computation is done serially.
The output of this function is a list with two elements, `latent.predict` and `Y.predict`, which are $m \times q$ matrices containing samples from the posterior predictive distribution of $\boldsymbol{S}^p$ and $\boldsymbol{Y}^p$, respectively, in which q is the length of the chain in the input `run.m`.

Once the above posterior predictive samples are computed, they can be summarized and mapped in the usual ways, say by the mean and standard deviation of the (marginal) posterior predictive distributions. For the `Weed` dataset the code below performs these tasks for the attribute of interest $e^{S(\cdot)}$, whose predictive summaries are mapped in Figure 4 (additionally using **colorspace**, Zeileis, Hornik, and Murrell 2009):

```
R> int.pred <- rowMeans(exp(Ypred$latent))
R> int.unc <- apply(exp(Ypred$latent), 1, FUN = sd)
R> tt1 <- numeric(length(re))
R> tt2 <- numeric(length(re))
R> tt1[re == 1] <- int.pred
R> tt2[re == 1] <- int.unc
R> size <- c(0.3, 2.7)
R> Y <- Weed[, 3]
R> library("colorspace")
R> filled.contour(eastings, northings, matrix(tt1, 30, ),
+    zlim = c(3, 300), color.palette = sequential_hcl, nlevels = 30,
+    xlab = "Eastings (meter)", ylab = "Northings (meter)",
+    plot.axes = { axis(1); axis(2); points(Weed[, 1:2], col = 1,
+      cex = size[1] + size[2] * (Y - min(Y))/(max(Y) - min(Y)))})
R> filled.contour(eastings, northings, matrix(tt2, 30, ),
+    zlim = c(0, 200), color.palette = sequential_hcl, nlevels = 30,
+    xlab = "Eastings (meter)", ylab = "Northings (meter)",
+    plot.axes = {axis(1); axis(2); points(Weed[, 1:2], col = 1,
+      cex = size[1] + size[2] * (Y - min(Y))/(max(Y) - min(Y)))})
```

Likewise, a similar procedure can be used to summarize and map the possible response counts $\boldsymbol{Y}^p$ that could have been measured at the prediction locations. Such maps are useful tools in precision farming for designing a judicious strategy to spread herbicides over an agricultural field, for instance, spreading larger doses over the subregions in Figure 4 (top) that are whitish in color, and smaller doses over the subregions that are bluish.

6.6. Multiple chain generation with parallel computing

In this section we report the results of experiments to assess running times when the parallel capabilities of **geoCount** are used. The experiments were run in the following computer environment:

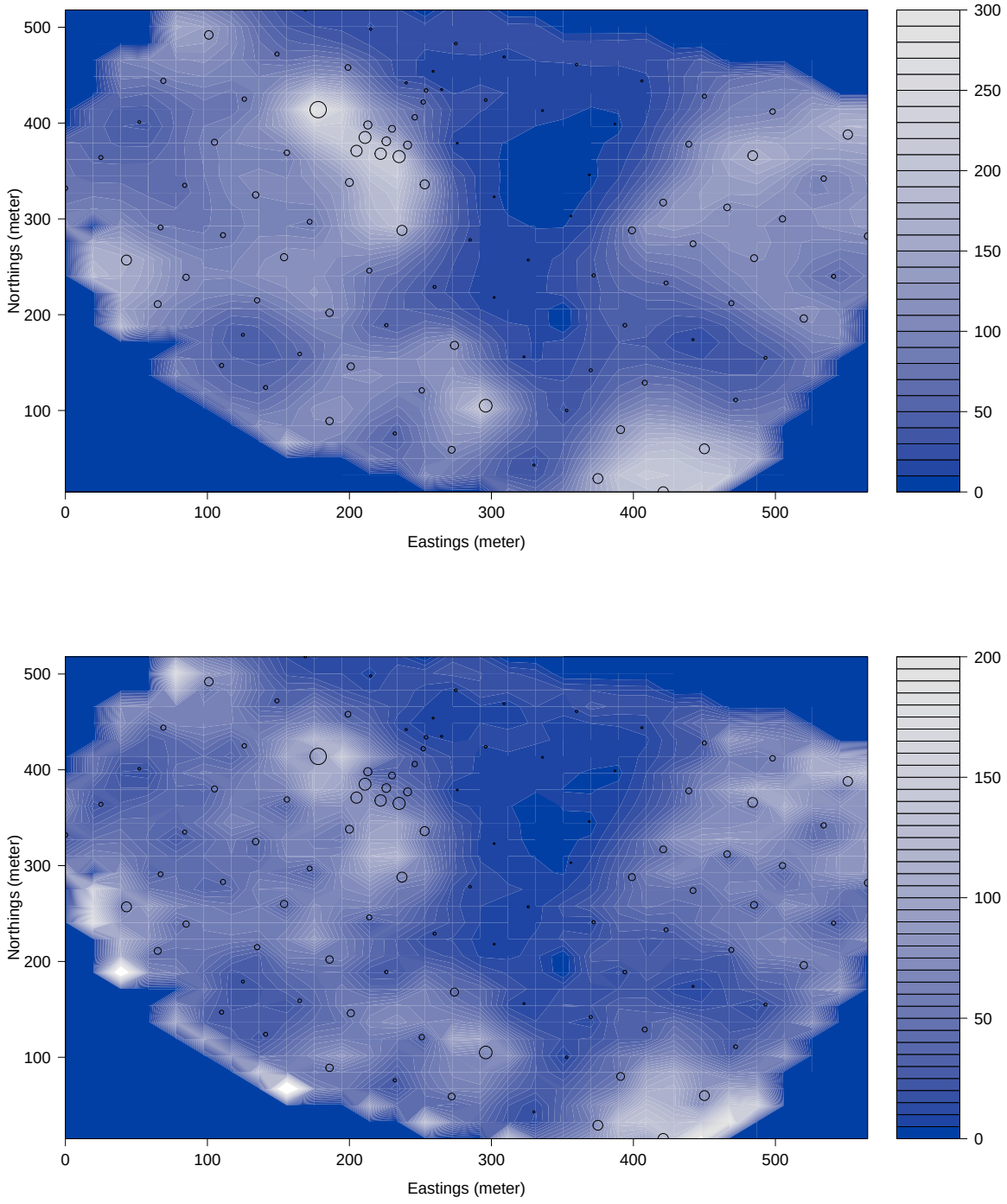

Figure 4: Predictive summary maps for the `Weed` dataset: Map of predicted intensity (top), and map of uncertainty about the predictive intensity (bottom).

Chains	1	2	4	6	8	10	12
Total	31.793	32.664	35.060	49.138	60.010	88.451	90.848
Average	31.793	16.332	8.765	8.189	7.501	8.845	7.571

Table 2: Running times (in seconds) for the generation of 1000 posterior samples per each simulated chain.

- Computer: Toshiba Satellite A505-S6030.
- CPU: Intel Core i7-720QM (6M Cache, 1.60 GHz), 4 Cores and 8 Threads (which means the maximum number of processing units for parallel computing is 8).
- Memory: 4GB DDR3 1066MHz memory.
- Operation system: Linux Ubuntu 11.04.
- Software: R 2.13.0, **RStudio** 0.94.84.
- Packages: **snow** 0.3-7, **snowfall** 1.84.

To perform the experiments we used a dataset simulated at 100 sampling locations in a regular grid of $[0,1] \times [0,1]$ using the Poisson-lognormal spatial model with exponential covariance function, constant mean, and parameters $\beta = 5$, $\sigma = 0.5$ and $\phi = 0.2$.

Based on the above model and dataset we simulated posterior samples for the parameters and latent variables using different number of chains of size 1000 each (with cluster type "SOCK", see documentation of **snow** for details); the results are summarized in Table 2. It shows that the running times increase with the number of chains, but the average time for generating 1000 posterior samples decreases and achieves the minimum for 8 chains (there were a total of 8 available processors).

7. Comparison between geoCount and geoRglm

The R package **geoRglm**, developed by Christensen and Ribeiro (2002), is an extension of the well-known R package **geoR** developed by Ribeiro and Diggle (2001), that also provides functions for Bayesian inference and prediction for the Poisson-lognormal and binomial-logitnormal models (4) and (5). Like **geoCount**, **geoRglm** uses MCMC algorithms to sample from the posterior distribution of $(\boldsymbol{S}, \boldsymbol{\beta}, \boldsymbol{\theta})$, where group updating is done based on Langevin-Hasting updates for the first two blocks and one-dimensional random walk Metropolis for the components of the last block. But unlike **geoCount**, **geoRglm** uses the so-called non-centered parameterization (see Papaspiliopoulos, Roberts, and Sköld (2007) for details) that seeks to turn the components of $\boldsymbol{S}$ and $\boldsymbol{\beta}$ uncorrelated a *priori*, rather than a *posteriori* as **geoCount** does, and no capability for parallel computation is available. **geoRglm** implements the computationally expensive MCMC sampling and computation of large matrices in C; for details see Christensen and Waagepetersen (2002), Diggle and Ribeiro (2007), and the **geoRglm** webpage `http://gbi.agrsci.dk/~ofch/geoRglm/`.

Christensen, Roberts, and Sköld (2006) found that the performance of MCMC algorithms based on the centered and non-centered parameterizations depend greatly on the observed data, and for some datasets the algorithms based on these parameterizations may perform

quite poorly. To illustrate this they used two real datasets, one consisting of large counts (the `Rongelap` dataset described in Section 6.1) and the other of small counts, and compared the efficiency of MCMC algorithms based on the centered, non-centered and data-based parameterizations. They found that neither the centered nor the non-centered parameterizations were particularly efficient for these datasets, since posterior samples of parameters and latent values displayed very strong autocorrelations. Nevertheless, they found that the centered parameterization was more efficient than the non-centered one for the large count dataset, while the opposite held true for the small count dataset. On the other hand, the data-based parameterization was substantially more efficient than the other two for both datasets, since posterior autocorrelations of parameters and latent values were substantially smaller.

In this section we carry out a similar efficiency comparison between **geoRglm** that implements the non-centered parameterization and **geoCount** that implements the data-based parameterization, using the `Weed` and `Rhizoc` datasets described in Section 6.1. For each combination of dataset and package we simulated a single chain of size 55000, removed the first 5000 simulated values (burn-in), and then kept every 10th simulated value to assess mixing and convergence of the simulated chain. We do not use the parallel capability of **geoCount** to make the fitting comparable with **geoRglm** in most possible respects.

7.1. Fitting the `Weed` dataset

Following the analysis in Guillot, Lorén, and Rudemo (2009), we fit the `Weed` dataset using the Poisson-lognormal spatial model (4) with constant mean and exponential covariance function. The chosen priors for the parameters are $\pi(\beta) \propto 1$, $\pi(\sigma) \propto \sigma^{-2}\exp(-2/\sigma)$ and $\pi(\phi) \propto \mathbf{1}_{[10,300]}(\phi)$. For both packages the tuning parameters in the Langevin-Hastings and Metropolis-Hastings algorithms in **geoRglm** and **geoCount** were selected by trial and error on short pilot chains to achieve acceptance rates as close as possible to the optimal acceptance rates (see Section 5.3). Below we display first the **geoRglm** code and later the **geoCount** code to fit the `Weed` dataset:

```
R> library("geoR")
R> library("geoRglm")
R> MOD.w <- model.glm.control(trend.d = "cte", cov.model = "exponential")
R> MCc.w <- mcmc.control(S.scale = 0.0013, phi.scale = 4.5, phi.start = 150,
+    burn.in = 5000, thin = 10, n.iter = 50000)
R> PGC.w <- prior.glm.control(beta.prior = "flat", phi.prior = "uniform",
+    phi.discrete = seq(10, 300, by = 0.15), sigmasq.prior = "sc.inv.chisq",
+    sigmasq = 2, df.sigmasq = 2)
R> OUT.w <- output.glm.control(messages = TRUE)
R> pkb.w <-  pois.krige.bayes(coords = Weed[, 1:2], data = Weed[, 3],
+    units.m = rep(1, 100), model = MOD.w, prior = PGC.w, output = OUT.w,
+    mcmc.input = MCc.w)
R> library("geoCount")
R> MMCc.w <- MCMCinput(Y.family = "Poisson", rho.family = "rhoPowerExp",
+    run = 55000, run.S = 1, phi.bound = c(10, 300), priorSigma = "InvGamma",
+    parSigma = c(1, 2), ifkappa = 0, initials = list(c(4), 0.7, 100, 1),
+    scales = c(0.55, 3.5, 0.5, 0.4, 1))
```

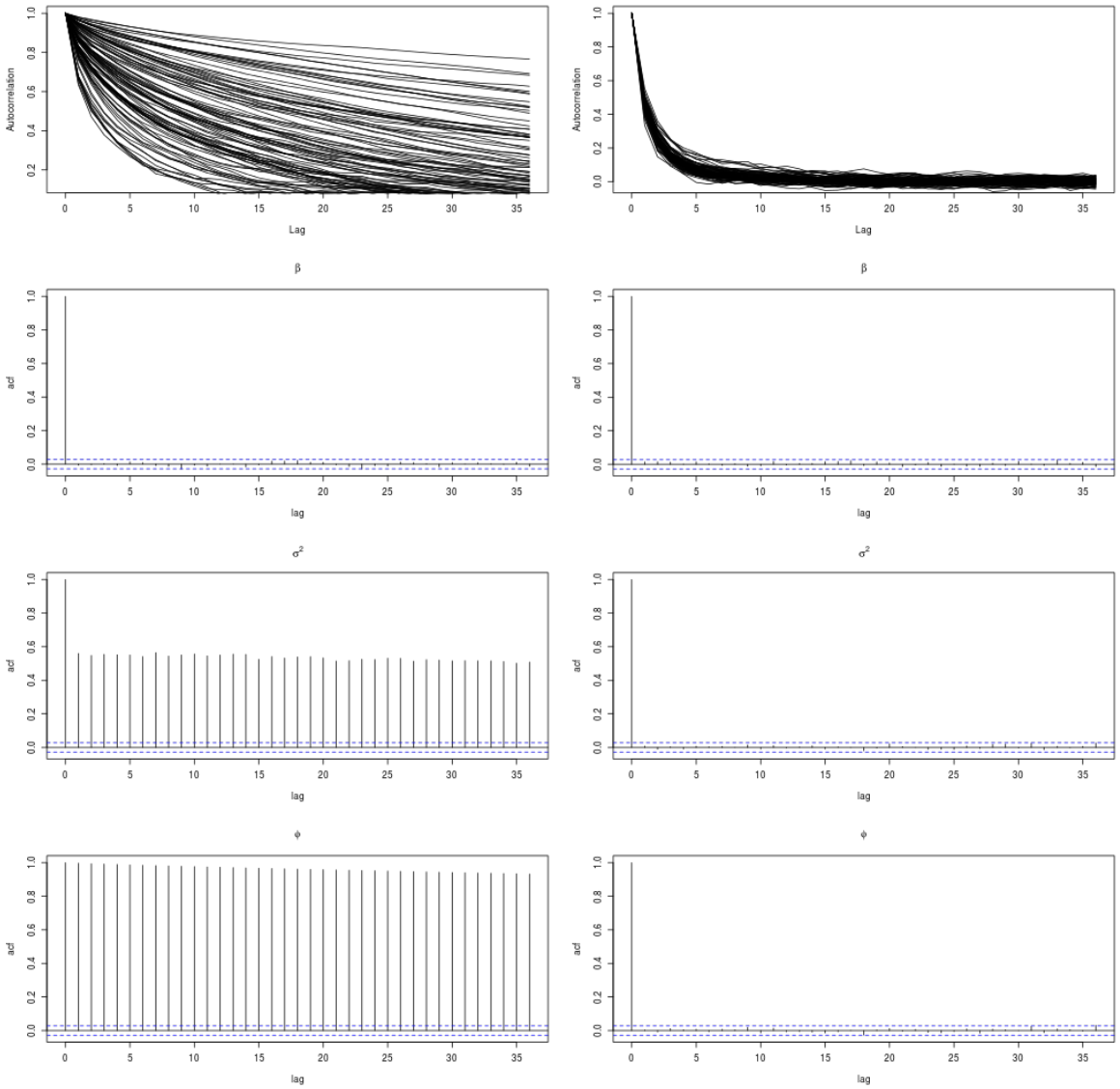

Figure 5: `Weed` dataset: Autocorrelation plots of the latent variables and parameters of the simulated chains from the output of **geoRglm** (left) and **geoCount** (right).

```
R> ppk.w <- runMCMC(loc = Weed[, 1:2], Y = Weed[, 3], L = 0, X = NULL,
+    MCMCinput = MMCc.w)
R> ppk.w <- cutChain(ppk.w, chain.ind = 1:4, burnin = 5000, thinning = 10)
```

The above **geoRglm** code took about 54 seconds to run, while **geoCount**'s took about 1230 seconds; the latter would be reduced if parallel chains are run by using `runMCMC.sf` rather than `runMCMC` (see Section 6.6). Figure 5 displays side-by-side autocorrelation plots of the latent variables at the sampling locations and parameters of the simulated chains from the output of **geoRglm** (left) and **geoCount** (right). For **geoRglm** the autocorrelation among latent variables and covariance parameters is extremely high, making inferences based on this output unreliable. On the other hand, for **geoCount** the autocorrelation among all latent variables and parameters is either low or negligible, so inferences based on this output are

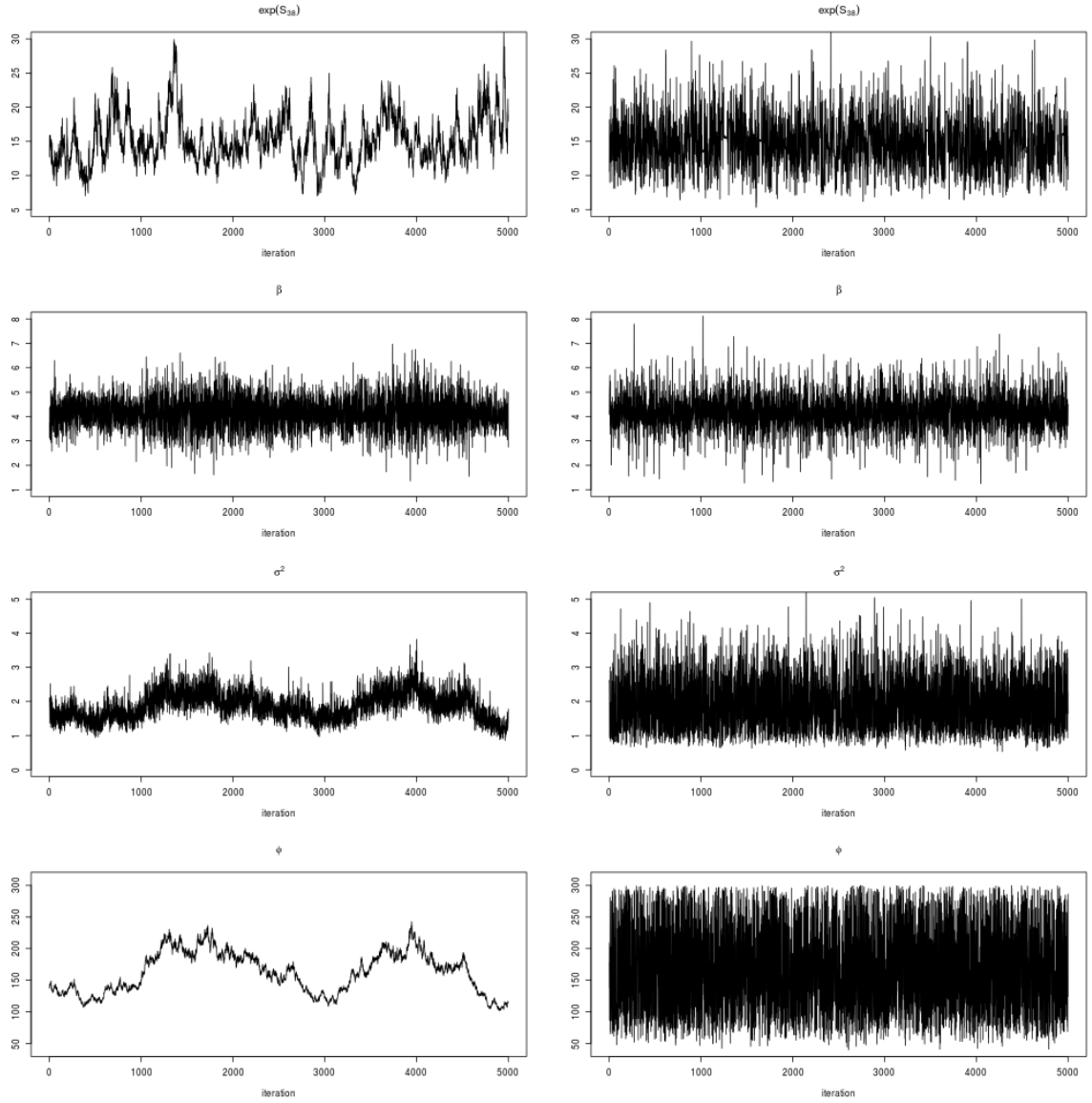

Figure 6: `Weed` dataset: Trace plots of the latent variable $e^{S_{38}}$ and parameters of the simulated chains from the output of **geoRglm** (left) and **geoCount** (right).

much more reliable. Likewise, Figure 6 displays trace plots for one of the latent variables and the model parameters. It is clear from these plots that **geoRglm** chains mix very slowly for this dataset, and the convergence of some of them is even doubtful. On the other hand, **geoCount** chains appear to mix well and their convergence is apparent.

A referee pointed out that the above comparison might not be fair to **geoRglm** since it does not take into account computational speed, and the faster **geoRglm** should be allowed to run longer to have a better chance to reach convergence. To investigate this we perform a second run of **geoRglm**, but now with the number of iterations, burn-in and thinning lengths all multiplied by 10, so we end up with the same effective simulation size as in the previous run:

```
R> MCc.w2 <- mcmc.control(S.scale = 0.0013, phi.scale = 4.5, phi.start = 150,
+    burn.in = 50000, thin = 100, n.iter = 500000)
```

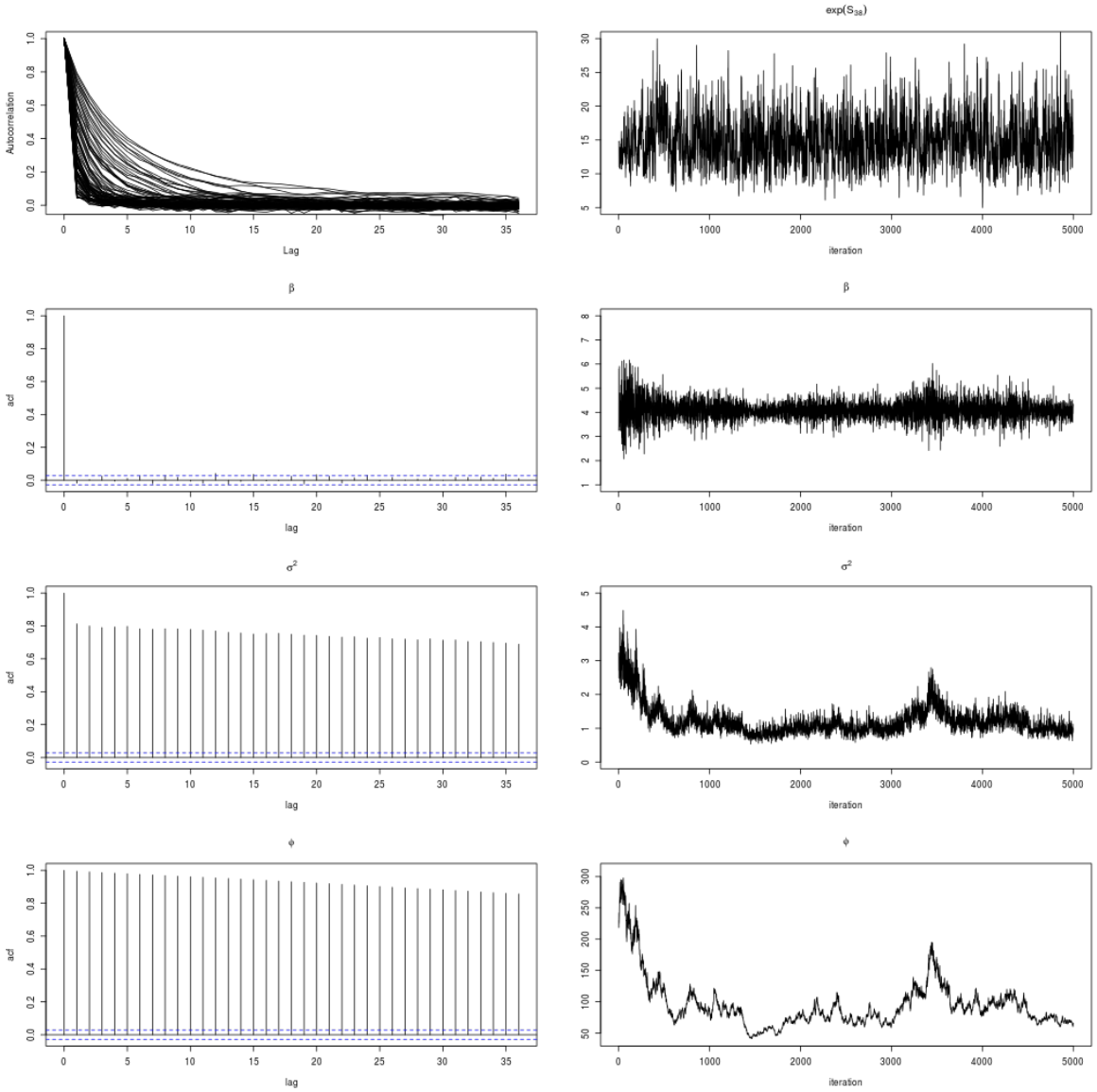

Figure 7: `Weed` dataset (2nd run): Autocorrelation plots of the latent variables and parameters of the simulated chains from the output of **geoRglm** (left), and trace plots of the latent variable $e^{S(\boldsymbol{x}_{38})}$ and parameters of the simulated chains from the output of **geoRglm** (right).

```
R> pkb.w2 <- pois.krige.bayes(coords = Weed[, 1:2], data = Weed[, 3],
+     units.m = rep(1, 100), model = MOD.w, prior = PGC.w, output = OUT.w,
+     mcmc.input = MCc.w2)
```

Figure 7 displays autocorrelation plots of the latent variables at the sampling locations and parameters of the simulated chains (left) and trace plots for one of the latent variables and the model parameters (right). These plots suggest that, for the `Weed` dataset, the MCMC algorithm implemented in **geoRglm** has serious convergence problems, and running longer chains seems to be of no avail.

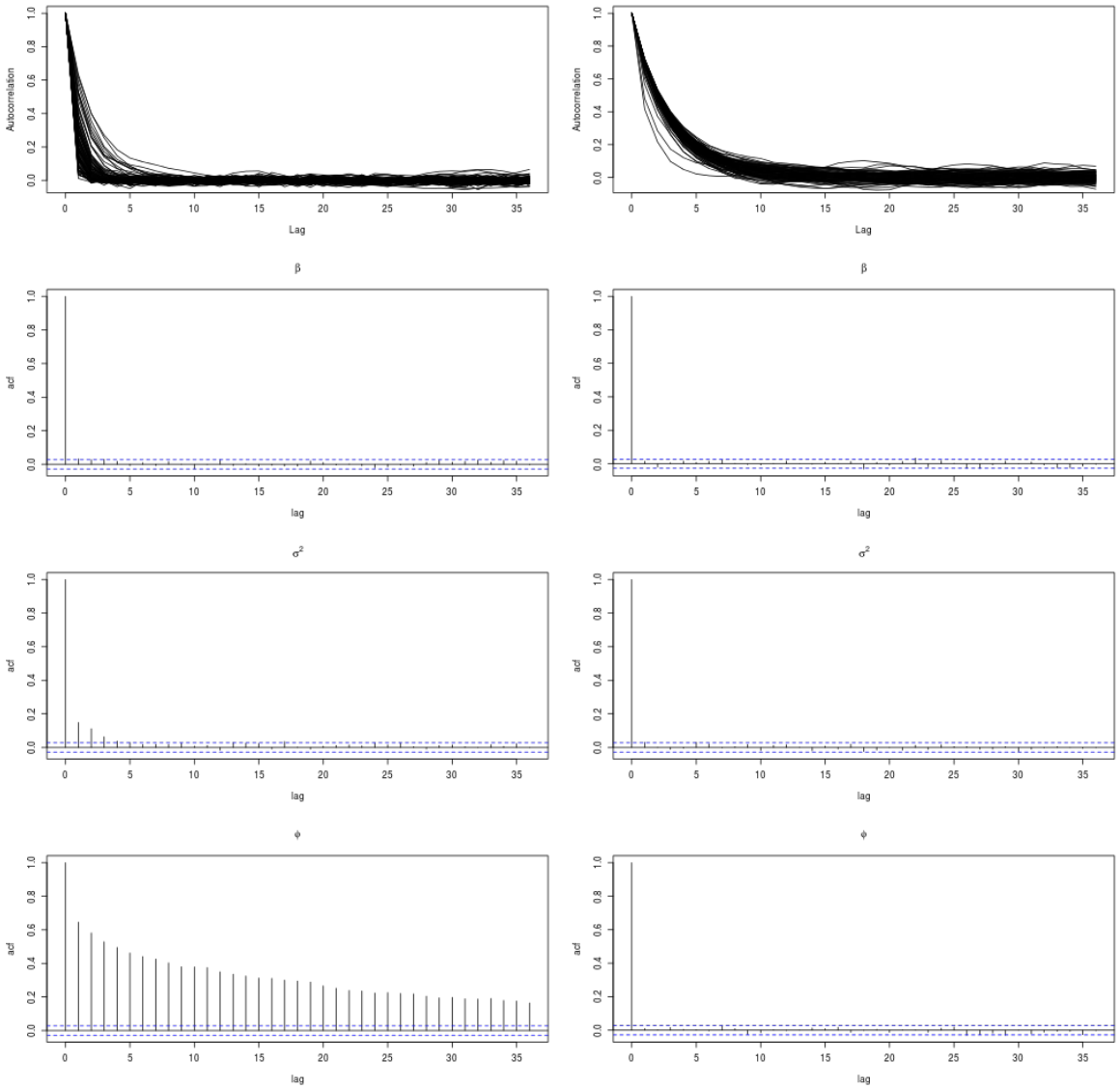

Figure 8: `Rhizoc` dataset: Autocorrelation plots of the latent variables and parameters of the simulated chains from the output of **geoRglm** (left) and **geoCount** (right).

7.2. Fitting the `Rhizoc` dataset

Following the analysis in Zhang (2002), we fit the `Rhizoc` dataset using the binomial-logitnormal spatial model (5) with constant mean and spherical covariance function. The chosen priors for the parameters are the same as those used to fit the `Weed` dataset. Below we display first the **geoRglm** code and later the **geoCount** code to fit the `Rhizoc` dataset:

```
R> data("Rhizoc")
R> MOD.rh <- model.glm.control(trend.d = "cte", cov.model = "spherical")
R> MCc.rh <- mcmc.control(S.scale = 0.025, phi.scale = 300, phi.start = 40,
+    burn.in = 5000, thin = 10, n.iter = 50000)
R> PGC.rh <- prior.glm.control(beta.prior = "flat", phi.prior = "uniform",
```

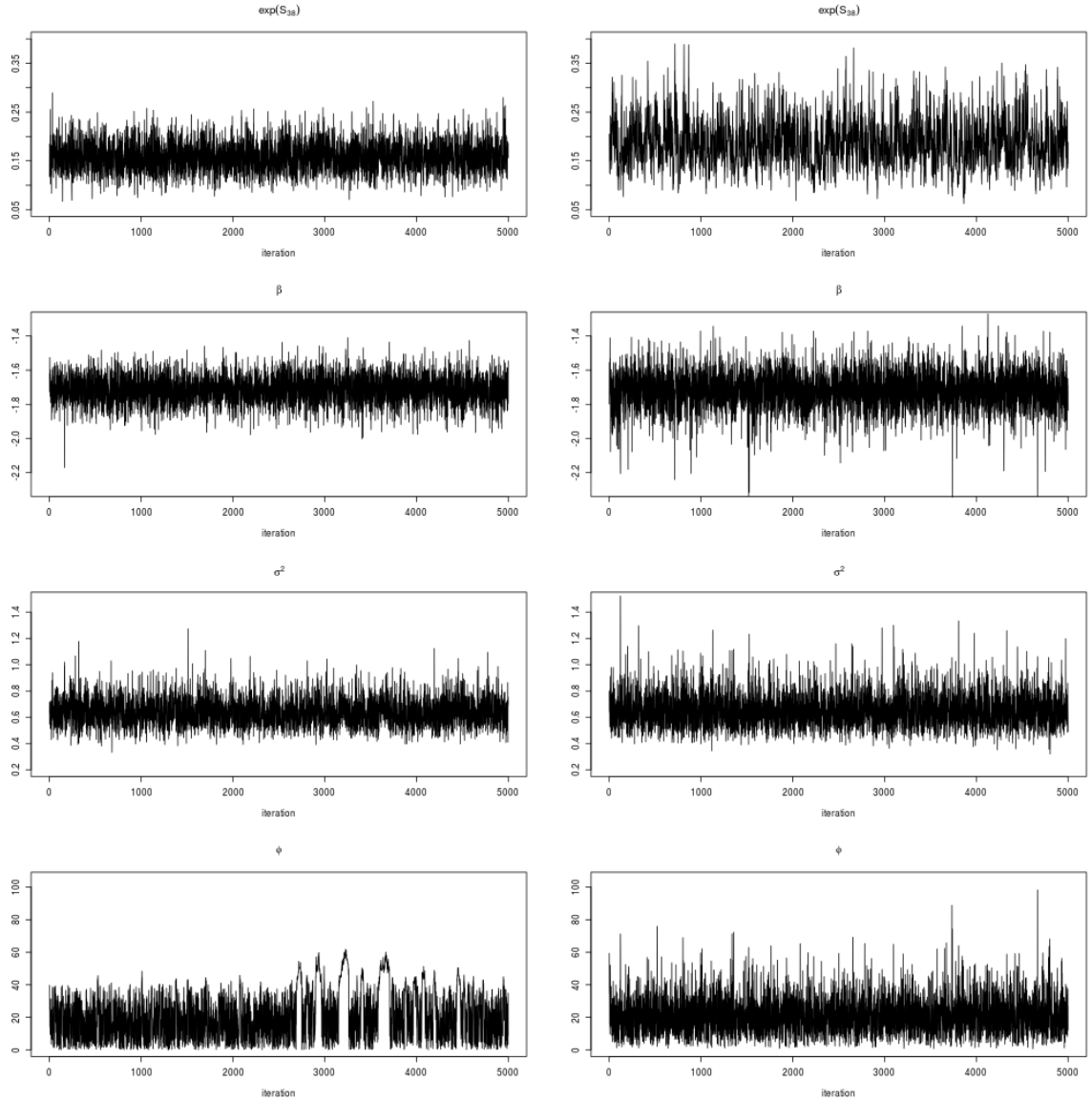

Figure 9: `Rhizoc` dataset: Trace plots of the latent variables and parameters of the simulated chains from the output of **geoRglm** (left) and **geoCount** (right).

```
+    phi.discrete = seq(0, 100, by = 0.05), sigmasq.prior = "sc.inv.chisq",
+    sigmasq = 2, df.sigmasq = 2)
R> OUT.rh <- output.glm.control(messages = TRUE)
R> pkb.rh <- binom.krige.bayes(coords = Rhizoc[, 1:2], data = Rhizoc[, 4],
+    units.m = Rhizoc[, 3], model = MOD.rh, prior = PGC.rh, output = OUT.rh,
+    mcmc.input = MCc.rh)
R> MMCc.rh <- MCMCinput(Y.family = "Binomial", rho.family = "rhoPowerExp",
+    run = 55000, run.S = 1, phi.bound = c(0, 100), priorSigma = "InvGamma",
+    parSigma = c(1, 2), ifkappa = 0, initials = list(c(-1), 0.7, 40, 1),
+    scales = c(0.15, 3.3, 0.2, 1.1, 1))
R> ppk.rh <- runMCMC(loc = Rhizoc[, 1:2], Y = Rhizoc[, 4], L =  Rhizoc[, 3],
+    X = NULL, MCMCinput = MMCc.rh)
```

```
R> ppk.rh <- cutChain(ppk.rh, chain.ind = 1:4, burnin = 5000, thinning = 10)
```

Figures 8 and 9 display, respectively, side-by-side autocorrelation and trace plots of the latent variables at the sampling locations and parameters of the simulated chains from the output of **geoRglm** (left) and **geoCount** (right). For this dataset the efficiency of the algorithms in both packages is comparable since the autocorrelation among all latent variables and parameters is either low or negligible (with the exception of ϕ in **geoRglm**). In regard to mixing, the chains produced by **geoCount** seem to mix a bit better though, as they do a better job at visiting the tails of the (marginal) posterior distributions.

The efficiency advantages of **geoCount** over **geoRglm** do not come without a price though: the running times of the former are larger than the latter. This difference is due mainly to two reasons. First, the data-based parameterization implemented in **geoCount** involves more computational overhead and a larger amount of matrix computations. Second, a fair amount of the running time savings in **geoRglm** are achieved by precomputing and storing the inverse of the correlation matrix $\big(\rho(u_{ij})\big)$ for a grid of points of the range parameter ϕ. A side effect of this choice is that the efficiency of the algorithm depends somewhat on the selected grid and poor mixing may result, even when using the maximum grid size (currently at 2001). In addition, it is not possible under this approach to account for the uncertainty in the smoothness parameter κ, which needs to be assumed fixed.

To offset this slowdown, **geoCount** allows the simulation of several posterior samples in parallel, which can roughly divide the running times by the number of CPUs that are used; see Section 6.6. In addition, shorter chains are usually sufficient for adequate inference due to the faster convergence of the MCMC algorithm implemented in **geoCount**.

8. Comparison between geoCount and INLA

The R package **INLA** performs Bayesian inference for a large class of hierarchical models called latent Gaussian models, and uses the computational approach proposed by Rue, Martino, and Chopin (2009) called Integrated Nested Laplace Approximation. This class of models includes the Poisson-lognormal spatial model (4). But unlike geostatistical models where the latent Gaussian model is specified in terms of covariance functions, **INLA** specification uses Gaussian Markov random fields (GMRF). This is computationally advantageous as no inversions of covariance matrices are required, but has the drawback that it is difficult to construct GMRF with covariance functions having common desired behaviors, such as stationarity or isotropy. **INLA** performs Bayesian inference by approximating marginal distributions of interest and their summaries deterministically (rather than stochastically as is done in MCMC algorithms), using a combination of Laplace approximations and numerical quadrature rules. The combination of these two features makes **INLA** an alternative that is computationally much faster than the MCMC algorithms implemented in **geoCount** and **geoRglm**. At the time of writing **INLA** is not available from the Comprehensive R Archive Network (CRAN), but can be obtained from the website `http://www.R-INLA.org/`.

Recently, Lindgren, Rue, and Lindström (2011) provided a partial solution to the aforementioned drawback, based on a result in Whittle (1963) which states that Gaussian random fields with covariance functions in a certain subfamily of the Matérn family are solutions of some stochastic partial differential equations (SPDE) driven by Gaussian white noise. Lindgren, Rue, and Lindström (2011) found that these solutions can be expanded in terms of

Parameter	**geoCount**	**INLA**
Variance	σ^2	σ^2
Smoothness	κ	$\nu = \alpha - 1$
Range*	ϕ	$1/\kappa$

Table 3: Matching of **geoCount** and **INLA** parametrizations of the Matérn covariance function in $\mathbb{R}^2$ (*__INLA__ calls κ a scaling parameter and $\sqrt{8\nu}/\kappa$ a range parameter).

certain basis functions with random coefficients given by a certain GMRF. As a result, they proposed to do the modeling using covariance functions from this Matérn subfamily and the computations using the GMRF representation of the corresponding random field. This is the approach implemented in **INLA**.

Below we carry out a comparison between **geoCount** and **INLA** using the `Weed` dataset. The covariance functions in $\mathbb{R}^2$ that are implemented in **INLA** using the aforementioned SPDE approach are the Matérn subfamily of (2) obtained by restricting $\kappa \in \mathbb{N}$ (natural numbers), but **INLA** uses a different parametrization. Table 3 provides the matching between the two parametrizations of the covariance function. It should be further noted that **INLA** carries out the internal computations using a parametrization of the covariance parameters different from that in Table 3, and sets their priors by default. As a result, it does not seem possible to set the same prior for the covariance parameters in the current implementations of **geoCount** and **INLA**. With this caveat in mind, we fit the Poisson-lognormal spatial model (4) with constant mean and Matérn covariance function (2) with $\kappa = 1$ (so $\alpha = 2$). Below we display first the **INLA** code and later the **geoCount** code to fit the `Weed` dataset:

```
R> library("INLA")
R> data("Weed")
R> coords <- as.matrix(Weed[, 1:2])
R> prdomain <- inla.nonconvex.hull(coords)
R> prmesh <- inla.mesh.2d(boundary = prdomain, max.edge = c(30, 50))
R> A <- inla.spde.make.A(prmesh, loc = coords)
R> spde <- inla.spde2.matern(prmesh, alpha = 2)
R> stk.dat <- inla.stack(data = list(y = Weed[,3]),
+    A = list(A, 1), tag = "dat", effects = list(
+      list(i = 1:spde$n.spde),
+      data.frame(Intercept = 1, gEastings = inla.group(coords[, 1]),
+        gNorthings = inla.group(coords[, 2]))))
> f.0 <- y ~ 0 + Intercept + f(i, model = spde)
> r.0 <- inla(f.0, family = "poisson", control.compute = list(dic = TRUE),
+   data = inla.stack.data(stk.dat),
+   control.predictor = list(A = inla.stack.A(stk.dat), compute = TRUE))
R> library("geoCount")
R> input.Weed <- MCMCinput(Y.family = "Poisson", rho.family = "rhoMatern",
+    run = 2200, run.S = 1, phi.bound = c(10, 300), priorSigma = "Halft",
+    parSigma = c(1, 1), ifkappa = 0, initials = list(c(4), 0.7, 90, 1),
+    scales = c(0.55, 3.5, 0.5, 0.4, 1))
R> res.Weed <- runMCMC(loc = Weed[, 1:2], Y = Weed[, 3], L = 0, X = NULL,
+    MCMCinput = input.Weed)
```

Parameter	**INLA**	**geoCount**
β	(3.57, 4.04, 4.55)	(3.48, 4.07, 4.75)
σ	(0.65, 0.99, 1.52)	(0.83, 1.01, 1.44)
ϕ	(24.75, 35.56, 51.02)	(29.68, 41.81, 62.63)

Table 4: Summary of posterior inference about the model parameters using **INLA** and **geoCount**: 2.5%, 50%, 97.5% percentiles of marginal posteriors.

```
R> res.Weed <- cutChain(res.Weed, chain.ind = 1:4, burnin = 200,
+    thinning = 10)
R> r.f <- inla.spde2.result(r.0, "i", spde, do.transf = TRUE)
R> r.0$summary.fixed
R> exp(r.f$summary.log.variance.nominal)
R> exp(r.f$summary.log.range.nominal)/sqrt(8)
R> quantile(res.Weed$m.posterior, probs = c(0.025, 0.5, 0.975))
R> quantile(res.Weed$s.posterior, probs = c(0.025, 0.5, 0.975))
R> quantile(res.Weed$a.posterior, probs = c(0.025, 0.5, 0.975))
```

The above **INLA** code took about 15 seconds to run, while **geoCount**'s took about 141 seconds. The summary of inferences about the parameters is given in Table 4. Inferences about β and σ are quite close, but inference about ϕ differ more. We conjecture that the latter is due to the different priors for the covariance parameters used under both implementations, as it is known that posterior inferences about correlation parameters are often sensitive to the priors. Next, we compare **INLA** and **geoCount** predictive inferences about $S(\cdot)$ at 38 prediction locations (different from the sampling locations) placed throughout the region (additionally using the **splancs** package, Rowlingson and Diggle 2014):

```
R> stepsize <- 50
R> nxy <- round(c(diff(range(Weed[, 1])), diff(range(Weed[, 2])))/stepsize)
R> projgrid <- inla.mesh.projector(prmesh, xlim = range(Weed[, 1]),
+    ylim = range(Weed[, 2]), dims = nxy)
R> xmean <- inla.mesh.project(projgrid, r.0$summary.random$i$mean)
R> xsd <- inla.mesh.project(projgrid, r.0$summary.random$i$sd)
R> library("splancs")
R> table(xy.in <- inout(projgrid$lattice$loc,
+    cbind(Weed[, 1], Weed[, 2])))
R> res.pred <- cbind(x = projgrid$lattice$loc[, 1],
+    y = projgrid$lattice$loc[, 2], inout = xy.in,
+    mean = as.vector(xmean) + r.0$summary.fixed[1], sd = as.vector(xsd))
R> locp <- projgrid$lattice$loc[xy.in,]
R> plot(Weed[, 1:2])
R> points(locp, col = 5, pch = 10)
R> Lp <- rep(1, nrow(locp))
R> Ypred <- predY(res.Weed, loc = Weed[, 1:2], locp, X = NULL, Xp = NULL,
+    Lp = Lp, k = 1, rho.family = "rhoMatern", Y.family = "Poisson")
R> res.comp <- cbind(res.pred[xy.in, c(1:2, 4:5)],  mean.geoCount =
+    rowMeans(Ypred$latent), sd.geoCount = apply(Ypred$latent, 1, sd))
```

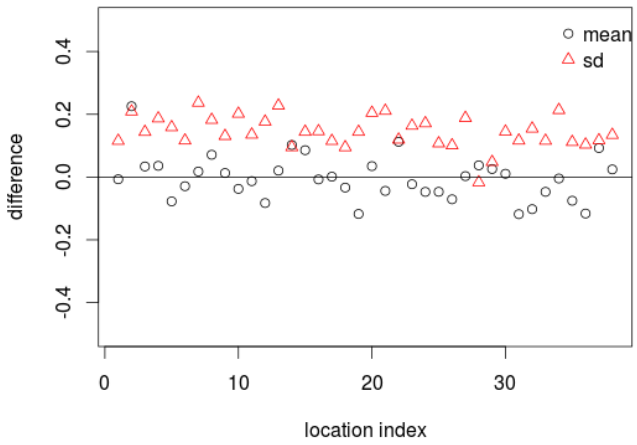

Figure 10: Differences between the means and standard deviations of the posterior predictive distributions at 38 locations obtained from **INLA** and **geoCount**.

Figure 10 plots, for each prediction location, the difference between the mean of the posterior predictive distributions computed using **INLA** and that using **geoCount** (○). The prediction discrepancies are small, with the average and maximum magnitude of the differences being, respectively, about 0.05 and 0.2; the latter is about 5% of the typical predicted value of $S(\cdot)$. Figure 10 also plots the corresponding differences between standard deviations of the posterior predictive distributions (△). In this case the discrepancies are substantial, with the average and maximum magnitude of the differences being, respectively, about 0.13 and 0.21; the latter is about 22% of the typical uncertainty value of $S(\cdot)$. We conjecture that the algorithm used by **INLA** might be overestimating the true predictive uncertainty, although we do not know why.

As a final point we note that the spatially varying attribute of interest is usually the intensity that drives the observed counts, which is represented by $e^{S(\cdot)}$ in the Poisson-lognormal spatial model. Currently, **INLA** provides predictive inference only about $S(\cdot)$, while the sampling-based approaches in **geoCount** and **geoRglm** allow predictive inference about any function of $S(\cdot)$.

9. Conclusions

9.1. Summary

This work describes the R package **geoCount** for the analysis of geostatistical count data. The package implements several tasks for the analysis of the Poisson-lognormal and binomial-logitnormal spatial models, two members in the class of generalized linear spatial models that are commonly used for the analysis of spatial count data. The main package capabilities are:

1. Simulation and visualization of geostatistical count data.
2. Simulation from the posterior distribution of the parameters and latent variables at the

sampling locations.

3. Simulation from the posterior predictive distribution of the latent variables and potential counts at unsampled locations.

On the programing side **geoCount** implements the computational intensive tasks in C++ and has parallel computation capability to speed up the computation processes, in case multi-processor or multi-core computers are available. On the algorithmic side **geoCount** implements group updating, Langevin-Hastings algorithms and a data-based parameterization to improve the efficiency of the MCMC algorithms. The MCMC algorithm implemented in **geoCount** appears more efficient than that implemented in **geoRglm**, and **geoCount** allows inference on the natural spatially varying quantity of interest, an intensity or probability, while **INLA** allows inference only on a specific transformation of it.

geoCount also implements some methods for assessing model adequacy based on posterior predictive Bayesian p values, proposed by Bayarri and Castellanos (2007), and transformed residuals, proposed by Jing (2011). These developments are still incipient and under current investigation, so they are not reported here; see the package documentation and Jing (2011) for a description of these methods.

9.2. Limitations

We end with a brief description of some limitations of the package **geoCount** and the models considered here, as well as possible areas for future improvement.

Package

The current implementation of **geoCount** allows the fitting of moderate size datasets (say in the hundreds) in a relatively short amount of time, but it will require substantially longer running times for large datasets (say in the thousands). In this regard, **geoRglm** and **INLA** run much faster, with the latter being the fastest. Implementing recent algorithms for storage, manipulation and computation with large non-structured covariance matrices is a possible avenue for substantial reduction of computation times in **geoCount**. Also, the Langevin-Hastings MCMC algorithm based on the data-based parameterization implemented in **geoCount** usually produces convergent and well mixed chains, but convergence issues may arise for some datasets. These are manifested by simulated chains with very strong autocorrelations. On the other hand, convergence is not an issue for **INLA** since it uses deterministic approximations. A possible avenue for improvement would be to also implement Hamiltonian Monte Carlo algorithms (Neal 2011), which seem to be capable to efficiently sample from posterior distributions with highly correlated components.

Models

The Poisson-lognormal spatial model uses the log-link function which implies that the spatially varying attribute of interest is log-normally distributed. Depending on the dataset, this may or may not be a reasonable assumption. Christensen (2004) extended it by using the Box-Cox family of link functions, and offered evidence that the identity-link function provided a better fit to the Rongelap dataset. De Oliveira (2013) proposed a class models for spatial count data that allows a more direct modeling of the distribution of the spatially varying

attribute of interest thought the use of copulas. It was shown that the models in this class, that includes the Poisson-lognormal spatial model, are not capable of representing moderate or strong spatial correlation in datasets consisting mostly of small counts. Similar comments and considerations might also apply to the binomial-logitnormal spatial model.

Finally, it is worthwhile to note that for this class of spatial models the prior distribution of the model parameters may have a sizeable influence on both the resulting posterior distribution and the efficiency of the MCMC algorithm. Because of this, it would be desirable to use objective or default priors aimed at minimizing the influence of the prior on the resulting posterior distribution (e.g., as those described in De Oliveira (2010) for Gaussian random fields), but such priors are yet to be developed for the models described here.

Acknowledgments

We thank an editor and two anonymous referees for helpful comments and suggestions that improved the article, Ole Christensen for MATLAB code and Hao Zhang for the Rhizoc dataset. This work was partially supported by the US National Science Foundation Grant DMS-1208896.

References

Bayarri MJ, Castellanos ME (2007). "Bayesian Checking of the Second Levels of Hierarchical Models." *Statistical Science*, **22**, 322–343.

Christensen OF (2004). "Monte Carlo Maximum Likelihood in Model-Based Geostatistics." *Journal of Computational and Graphical Statistics*, **13**, 702–718.

Christensen OF, Ribeiro PJ (2002). "**geoRglm**: A Package for Generalized Linear Spatial Models." *R News*, **2**(2), 26–28. URL http://CRAN.R-project.org/doc/Rnews/.

Christensen OF, Roberts GO, Sköld M (2006). "Robust Markov Chain Monte Carlo Methods for Spatial Generalized Linear Mixed Models." *Journal of Computational and Graphical Statistics*, **15**, 1–17.

Christensen OF, Waagepetersen R (2002). "Bayesian Prediction of Spatial Count Data Using Generalized Linear Mixed Models." *Biometrics*, **58**, 280–286.

Cressie N (1993). *Statistics for Spatial Data*. Revised edition. John Wiley & Sons, New York.

De Oliveira V (2010). "Objective Bayesian Analysis for Gaussian Random Fields." In M Chen, D Dey, P Muller, D Sun, K Ye (eds.), *Frontiers of Statistical Decision Making and Bayesian Analysis – In Honor of James O. Berger*, pp. 497–511. Springer-Verlag, New York.

De Oliveira V (2013). "Hierarchical Poisson Models for Spatial Count Data." *Journal of Multivariate Analysis*, **122**, 393–408.

Diggle PJ, Harper L, Simon S (1997). "Geostatistical Analysis of Residual Contamination from Nuclear Weapons Testing." In V Barnett, F Turkman (eds.), *Statistics for the Environment 3: Pollution Assessment and Control*, pp. 89–107. John Wiley & Sons, New York.

Diggle PJ, Ribeiro PJ (2007). *Model-Based Geostatistics*. Springer-Verlag, New York.

Diggle PJ, Tawn JA, Moyeed RA (1998). "Model-Based Geostatistics." *Journal of the Royal Statistical Society C*, **47**, 299–326.

Eddelbuettel D, François R (2011). "**Rcpp**: Seamless R and C++ Integration." *Journal of Statistical Software*, **40**(8), 1–18. URL http://www.jstatsoft.org/v40/i08/.

Eddelbuettel D, Sanderson C (2014). "**RcppArmadillo**: Accelerating R with High-Performance C++ Linear Algebra." *Computational Statistics & Data Analysis*, **71**, 1054–1063.

Gelman A (2006). "Prior Distributions for Variance Parameters in Hierarchical Models." *Bayesian Analysis*, **1**, 515–533.

Gelman A, Roberts GO, Gilks WR (1996). "Efficient Metropolis Jumping Rules." In J Bernardo, J Berger, A Dawid, A Smith (eds.), *Bayesian Statistics 5*, pp. 599–607. Oxford University Press, Oxford.

Guillot G, Lorén N, Rudemo M (2009). "Spatial Prediction of Weed Intensities from Exact Count Data and Image-Based Estimates." *Journal of the Royal Statistical Society C*, **58**, 525–542.

Jing L (2011). "Bayesian Model Checking for Generalized Linear Spatial Models for Count Data." *Unpublished Ph.D. Dissertation*, The University of Texas at San Antonio.

Knaus J (2013). ***snowfall**: Managing Parallel Execution of R Programs on a Compute Cluster*. R package version 1.84-4, URL http://CRAN.R-project.org/package=snowfall.

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**, 423–498.

Natarajan R, Kass RE (2000). "Reference Bayesian Methods for Generalized Linear Mixed Models." *Journal of the American Statistical Association*, **95**, 227–237.

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

Papaspiliopoulos O, Roberts GO, Sköld M (2003). "Non-centered Parameterizations for Hierarchical Models and Data Augmentation." In J Bernardo, M Bayarri, J Berger, A Dawid, D Heckerman, A Smith, M West (eds.), *Bayesian Statistics 7*, pp. 307–326. Oxford University Press, Oxford.

Papaspiliopoulos O, Roberts GO, Sköld M (2007). "A General Framework for the Parameterization of Hierarchical Models." *Statistical Science*, **22**, 59–73.

Plummer M, Best N, Cowles K, Vines K (2006). "**coda**: Convergence Diagnosis and Output Analysis for MCMC." *R News*, **6**(1), 7–11. URL http://CRAN.R-project.org/doc/Rnews/.

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/.

Ribeiro PJ, Diggle PJ (2001). "**geoR**: A Package For Geostatistical Analysis." *R News*, **1**(2), 15–18. URL http://CRAN.R-project.org/doc/Rnews/.

Roberts GO, Rosenthal JS (1998). "Optimal Scaling of Discrete Approximations to Langevin Diffusions." *Journal of the Royal Statistical Society B*, **60**, 255–268.

Roberts GO, Sahu SK (1997). "Updating Schemes, Correlation Structure, Blocking and Parameterization for the Gibbs Sampler." *Journal of the Royal Statistical Society B*, **59**, 291–317.

Roberts GO, Tweedie RL (1996). "Exponential Convergence for Langevin Diffusions and Their Discrete Approximations." *Bernoulli*, **2**, 341–363.

Rossini AJ, Tierney L, Li NM (2007). "Simple Parallel Statistical Computing in R." *Journal of Computational and Graphical Statistics*, **16**, 399–420.

Rowlingson B, Diggle PJ (2014). ***splancs**: Spatial and Space-Time Point Pattern Analysis*. R package version 2.01-36, URL http://CRAN.R-project.org/package=splancs.

Rue H, Martino S, Chopin N (2009). "Approximate Bayesian Inference for Latent Gaussian Models Using Integrated Nested Laplace Approximations." *Journal of the Royal Statistical Society B*, **71**, 319–353.

Sanderson C, Curtin R, Cullinan I, Bouzas D, Funiak S (2014). ***Armadillo**: C++ Linear Algebra Library*. Version 4.600, URL http://arma.sourceforge.net/.

Sevcikova H, Rossini AJ (2012). ***rlecuyer**: R Interface to RNG with Multiple Streams*. R package version 0.3-3, URL http://CRAN.R-project.org/package=rlecuyer.

The **GSL** Team (2013). *GNU Scientific Library – Reference Manual*. **GSL** 1.16, URL http://www.gnu.org/software/gsl/.

Whittle P (1963). "Stochastic Processes in Several Dimensions." *Bulleting of the International Statistical Institute*, **40**, 974–994.

Zeileis A, Hornik K, Murrell P (2009). "Escaping RGBland: Selecting Colors for Statistical Graphics." *Computational Statistics & Data Analysis*, **53**(9), 3259–3270.

Zhang H (2002). "On Estimation and Prediction for Spatial Generalized Linear Mixed Models." *Biometrics*, **58**, 129–136.

Affiliation:

Liang Jing
Quant Tech
RM 801 Xingfa Building
45 Zhongguancun Street
Haidian District, Beijing 100086, China
E-mail: ljing918@gmail.com

Victor De Oliveira
Department of Management Science and Statistics
The University of Texas at San Antonio
San Antonio TX 78249, United States of America
E-mail: victor.deoliveira@utsa.edu
URL: http://faculty.business.utsa.edu/vdeolive/

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume 63, Issue 11 *Submitted:* 2013-04-03
January 2015 *Accepted:* 2014-08-25