Extending `JAGS` for spatial data

Magno Tairone de Freitas Severino

Department of Statistics - ICEx - UFMG

February 2018

Extending JAGS for spatial data

Magno Tairone de Freitas Severino

Advisor: Vinícius Diniz Mayrink

Co-advisor: Fábio Nogueira Demarqui

Document submitted to the Graduate Program in Statistical Science of the Universidade Federal de Minas Gerais in partial fulfillment of the requirements to obtain the Degree of Master of Sciences.

Department of Statistics

Instituto de Ciências Exatas

Universidade Federal de Minas Gerais

Belo Horizonte, MG - Brasil

February 2018

Este trabalho é dedicado à todos que lutam por uma educação pública, gratuita e de qualidade nas universidades do Brasil.

Acknowledgements

Primeiro, meu agradecimento ao meu orientador, Vinícius Diniz Mayrink, e ao meu coorientador, Fábio Nogueira Demarqui, pela dedicação ao meu trabalho, pelas valorosas sugestões e pelo papel fundamental que tiveram na minha formação como Mestre em Estatística. A forma como me orientaram nesses últimos dois anos foi excepcional, com constante encorajamento e especialmente muita paciência nos diversos momentos em que o trabalho não fluía como o esperado. Um agradecimento especial ao Vinícius pelas incontáveis vezes que esteve disponível para sanar minhas dúvidas fora dos horários habituais das nossas reuniões.

Muita gratidão aos meus pais, Scheila e Magno, pelo suporte e incentivo aos meus mais de vinte anos de estudo. Agradeço aos meus avós, Therezinha e Francisco, que, de acordo suas palavras, torcem "para que eu realize todos os meus sonhos".

Um agradecimento cordial aos colegas da turma de mestrado, aos colegas do laboratório 3036 (o "cortiço") e aos membros do time Stats4Good, que se tornaram meus amigos: Augusto, Douglas, Caio, Guilherme Veloso, Guilherme Oliveira, Juliana, Jussiane, Larissa, Letícia, Luis, Matheus, Rumenick e Uriel. Foram ótimos momentos de estudo, trabalho e diversão.

Minha sincera gratidão ao Leonardo Azevedo pelo companherismo e motivação em todos os momentos.

Finalmente, um agradecimento à CAPES pelo suporte financeiro deste trabalho.

"Your deeds are your monuments".

R. J. Palacio

Abstract

Bayesian hierarchical modeling for spatial data is challenging for professionals from other areas than statistics. From a technical perspective, setting the model and the prior distributions are the simplest part of the process. What makes it difficult is the computation of the posterior full conditionals and the implementation of the Gibbs Sampler algorithm. The `BUGS` (Bayesian inference Using Gibbs Sampling) family of statistical softwares reduces the effort of modeling, since the user must indicate only the prior distributions and the likelihood function. However, in general these softwares do not implement several spatial models, although users of `WinBUGS` and `OpenBUGS` can enjoy from the spatial add-on called `GeoBUGS`. `JAGS` (Just Another Gibbs Sampler), the open-source `C++` developed version of the `BUGS` family, does not contain any function or distribution for spatial modeling. This project aims to fill this gap through the implementation of an extension to the `JAGS` software, allowing users from different fields to perform a spatial data modeling and analysis.

Keywords: spatial statistics, Bayesian inference, Gibbs sampling, `JAGS`.

Resumo

A modelagem hierarquica Bayesiana para dados espaciais pode ser bastante desafiadora para profissionais de áreas para além da estatística. Do ponto de vista técnico, a parte mais simples do processo é definir o modelo e as distribuições *a priori*. A dificuldade está no cálculo das distribuições condicionais completas *a posteriori* e na implementação do algoritmo amostrador de Gibbs. A família de *softwares* estatísticos `BUGS` (acrônimo para inferência Bayesiana usando o amostrador de Gibbs, em tradução literal) reduz o esforço de modelagem, uma vez que o usuário deve indicar apenas as distribuições *a priori* e a verossimilhança dos dados. Entretanto, em geral tais *softwares* não permitem análises mais complexas de dados espaciais, embora usuários do `WinBUGS` e do `OpenBUGS` podem usar o módulo espacial `GeoBUGS`. `JAGS` (apenas outro amostrador de Gibbs, em tradução literal), é uma alternativa à família `BUGS`, desenvolvida em `C++` com código aberto, não contém nenhuma função ou distribuição disponível que simplifique a modelagem de dados espaciais. O objetivo fundamental deste trabalho é suprir essa falta, implementando um módulo para o *software* `JAGS` que permita aos usuários de diferentes áreas realizar modelagem e análise de dados espacias de maneira simples.

Palavras-chave: estatística espacial, inferência Bayesiana, amostrador de Gibbs, `JAGS`.

List of Tables

List of Figures

Contents

Chapter 1

Introduction

The term spatial statistics refers to a wide range of statistical models and methods created for the analysis of spatially referenced data, generally observed at a discrete set of sampling locations within some region. Cressie (1993) presents the important aspects of theoretical spatial statistics. Chiles and Delfiner (1999) deal with models to describe natural variables distributed in space. Diggle and Ribeiro (2007) analyze applications of general statistical principles for modeling geostatistical problems, taking into account classical and Bayesian approaches. Bivand et al. (2008) showcases spatial data analysis presenting softwares, packages, functions, and methods for handling this type of data.

Hierarchical Bayesian modeling is often applied to spatial data, as remarkably explained by Banerjee et al. (2004). In this approach, it is necessary to specify the prior distributions, the likelihood function and compute the posterior full conditionals. In this situation, the Gibbs sampler algorithm, originated in the studies of Geman and Geman (1984) and Gelfand and Smith (1990), must be used to generate the posterior sample. Implementing this algorithm is not trivial for researchers not familiar with statistics, even for statisticians. An alternative to simplify this computational task arose with the family of softwares called `BUGS` (Bayesian inference Using Gibbs Sampling) that became popular to perform Bayesian analyses. The `WinBUGS` (www.mrc-bsu.cam.ac.uk/software/BUGS), released in mid-90s, and `OpenBUGS` (www.openbugs.net), released in 2005, are the part of this family that can handle some spatial data through `GeoBUGS` (an add-on that provides only two covariance functions and the CAR model; details in Section 3.1.1).

A compatible alternative software to the `BUGS` family is called `JAGS` (Just Another Gibbs Sampler), initially proposed in Plummer (2003), and released in December 2007. It was built to be extensible, allowing users to develop their own functions, distributions, and samplers for this software. `JAGS` currently lacks a module for performing spatial data analyses, but this work intends to fill this gap. Based on the paper from Wabersich and Vandekerckhove (2014), this dissertation aims to create an extension to `JAGS` that allows users to model point-referenced data and areal data.

Spatial modeling is widely used by several researchers such as statisticians, physicists, engineers, geologists, biologists, among others. Some recent studies on spatial statistics are listed bellow:

- Mohebbi et al. (2011) present a Poisson regression approach for modelling spatial autocorrelation between geographically referenced observations of esophageal cancer in Iran, considering areal and point-referenced data.

- Addis et al. (2015) conduct a study to assess the correlation among selected soil attributes and to illustrate the spatial pattern and dependence of neighboring observations in the Ethiopian highlands;

- Shen and Gong (2015) use the methods for exploratory spatial data analysis, semi-variogram function and other techniques to show the spatio-temporal evolution of economic disparities at county level in Guangdong, a Chinese province, from 1991 to 2013;

- de Melo et al. (2015) investigate the crime concentrations and the similarities among different crime types in the Brazilian city of Campinas using point pattern data;

- Gomes et al. (2016) compare the results of two spatial techniques applied to the area of transportation engineering. They model the transportation choice, estimating the probability of using private motorized (car or motorcycle) mean of travel in several geographical coordinates. The vehicle choice issues are strongly related to locations and the modeling is done by the use of the spherical and the Gaussian covariance functions.

Again, the previous examples are just a few cases. In the literature, it is possible to find a wide range of applications dealing with spatial statistics techniques.

This thesis is organized as follows. Chapter 2 reviews the theoretical framework of spatial statistics, focusing on point-referenced data and areal data. It also shows the definition of covariance function and the CAR distribution that will be used throughout this work.

Chapter 3 discusses the use of the Bayesian inference in spatial data through the softwares from the `BUGS` family and `JAGS`. We present the `BUGS` project, the `JAGS` software and a short tutorial on adding custom extensions to this open-source software. At the end, we introduce our module to simplify point-referenced and areal data modeling in `JAGS`.

After Chapter 3, we present a validation study divided into two parts, one for each type of spatial data. Chapter 4 is intended to validate the covariance functions implemented in the proposed module by using a simple data set of elevations from Diggle and Ribeiro (2007). Here, we compare the parameter estimates (obtained by using `JAGS`) with those obtained from the maximum likelihood method. There is no particular interest in the mentioned data set, it was used merely to confirm that the new module works accurately and in accordance with the parameter estimation done using the frequentist approach. In addition, we demonstrate how to perform prediction at arbitrary locations using the proposed module (comparing the results with those from `GeoBUGS`). Chapter 5 is focused on validating the CAR distribution available in the proposed module. We use a real data set about the gross domestic product in the municipalities of the Brazilian state of Minas Gerais. Additionally, we present two simulation studies in which we set different specifications for the variance parameters in the models. Comparison between the results from `GeoBUGS` and the proposed module are shown to confirm that the new module behaves correctly.

Chapter 6 is dedicated to present the usage of the `JAGS` spatial module with real data sets, demonstrating that our module can be used in a wide range of statistical applications. We begin with a study on precipitation and temperature in the Brazilian North region (where part of the Amazon rain forest is located), using point-referenced data models. Lastly, we explore survival analysis, presenting a data set on the survival of leukemia cancer that has been already explored in the literature. We use two approaches to fit spatial survival data set using the proposed module.

Finally, Chapter 7 presents the main discussions, highlights the contributions so far and the directions for future research.

Chapter 2

Basics of spatial data

This chapter is dedicated to the presentation of basic aspects related to spatial statistics. The purpose of this chapter, divided into three sections, is to establish the theoretical background for this work. We begin with the definition of spatial data and the concepts regarding this field of study in Section 2.1. Section 2.2 covers concepts related to point-referenced data that shall be needed in later chapters. Ultimately, Section 2.3 is concerned with areal data and the definition of the conditionally autoregressive model.

2.1 Spatial data

Spatial data refers to data being geographically referenced, often presented as maps. As stated in Banerjee et al. (2004), spatial data sets are classified into one of the following types:

- Point-referenced data: the response $Y(\mathbf{s})$ is a random value at a location $\mathbf{s} \in \mathbb{R}^r$, where $\mathbf{s}$ varies continuously over D, a fixed subset of $\mathbb{R}^r$ that contains an r-dimensional rectangle of positive volume;
- Areal data: here, D is a fixed subset of $\mathbb{R}^r$ that is partitioned into a finite number of areal units (of regular or irregular shape) with defined boundaries;
- Point-pattern data: in this case, D is itself random; its index set gives the locations of random events that form a spatial point pattern. Such type of data are often

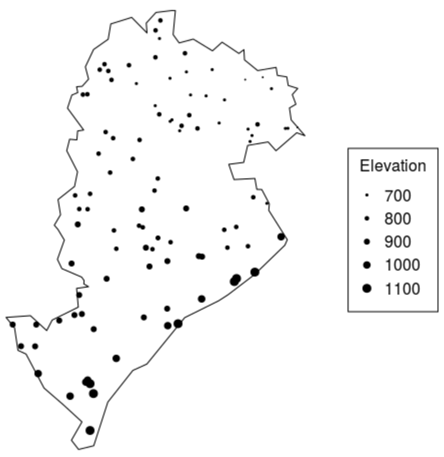


Figure 2.1: Circle plot of the surface elevation data in the city of Belo Horizonte. Circles are plotted with centers at the location of the observation and radii resembling the magnitude of the elevations (in meters).

of interest in studies of event clustering, where points tend to be spatially close to other points.

In this work, we will focus our attention into point-referenced and areal data models. Details about point-pattern data can be found in Diggle (2003).

Figure 2.1 offers an example of point-referenced data set, presenting the elevation measurements of 100 points within the Brazilian city of Belo Horizonte. This data set was obtained via `R` (see R Core Team, 2017) by randomly selecting the locations in this city and retrieving their respective elevation measurements using the `googleway` package (see Cooley, 2017). Clearly, the central and north regions present the lowest magnitude of elevations, whilst the south border has the highest altitudes. In this case, it is reasonable to expect that observed near points will present similar values. In this type of data, one might be interested in performing predictions for points with no information about their elevation within the study area.

An illustrative case of areal data is presented in Figure 2.2. In this map we have information about the number of places considered either national, state or municipal heritage site in the municipalities of the Minas Gerais state in Brazil. The data, originally from IEPHA (Historical and Artistic Heritage Institute of the Minas Gerais State), is available in Instituto Pristino (2013). The figure uses the red scale to classify values

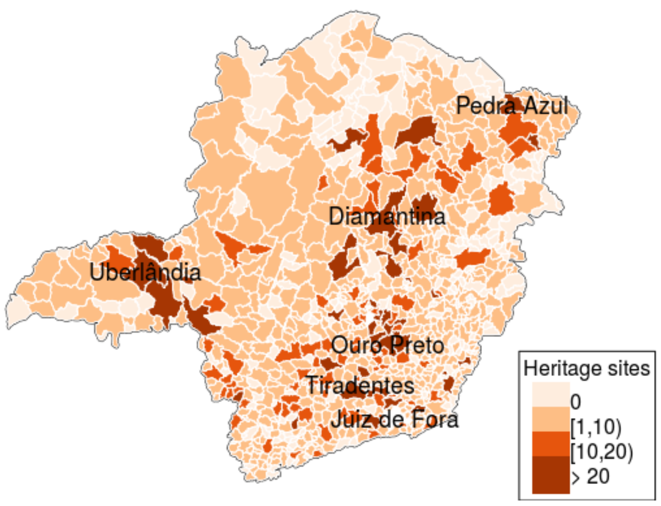


Figure 2.2: Number of registered heritage sites per municipality in the Brazilian state of Minas Gerais in 2013.

into a few classes, according to the number of heritage sites in each city. From the central-north to the south axis is located the Royal Road (Estrada Real, in Portuguese), a touristic route that comprises cities built in mid 17th century by Portuguese gold explorers (including the cities of Diamantina, Ouro Preto, Tiradentes, and Juiz de Fora, highlighted in the map). The remaining regions with high concentration of heritage sites (cities of Pedra Azul, in the northeast, and Uberlandia and its neighbors, in the west) basically consist of historic buildings registered as municipal heritage sites.

2.2 Point-referenced data

Let $\boldsymbol{s} \in \boldsymbol{R}^r$ be a generic data location in the r-dimensional Euclidean space and suppose that the data $Y(\mathbf{s})$ at location $\mathbf{s}$ is a random value; r is usually 2 or 3 for most applications (Banerjee et al., 2004, p. 21). Let $\mathbf{s}$ vary over the index set $D \subset \mathbb{R}^r$ so as to generate the multivariate random process

$$\{\boldsymbol{Y}(\boldsymbol{s}) : \boldsymbol{s} \in D\}. \tag{2.1}$$

We assume that this process has mean $\mathbb{E}[Y(\boldsymbol{s})]$ and that the variance of $Y(\boldsymbol{s})$ exists for all $\boldsymbol{s} \in D$. The notation presented here is based on the stochastic model defined in Cressie

(1993). According to Banerjee et al. (2004), the process in Expression (2.1) is said to be *Gaussian* if $\boldsymbol{Y} = (Y(\boldsymbol{s_1}), \ldots, Y(\boldsymbol{s_n}))^\top$ has a multivariate normal distribution for any set of sites $\{\boldsymbol{s_i}, i = 1, \ldots, n\}$, $n \geq 1$. In addition, we say that this process is *strictly stationary* if, for any set of n sites $\{\boldsymbol{s_1}, \ldots, \boldsymbol{s_n}\}$, $n \geq 1$, the distribution of $(Y(\boldsymbol{s_1}), \ldots, Y(\boldsymbol{s_n}))$ is the same as that of $(Y(\boldsymbol{s_1} + \boldsymbol{h}), \ldots, Y(\boldsymbol{s_n} + \boldsymbol{h}))$.

In the case of the process having a constant mean, i.e. $\mathbb{E}(\boldsymbol{s}) \equiv \mu$, and $\mathbb{C}\text{ov}(Y(\boldsymbol{s}), Y(\boldsymbol{s} + \boldsymbol{h})) = C(\boldsymbol{h})$, for all $\boldsymbol{h} \in \mathbb{R}^r$ such that $\boldsymbol{s}$ and $\boldsymbol{s} + \boldsymbol{h}$ lie within D, we say that the process has *second-order stationarity* (or *weak stationarity*). This condition implies that the covariance relationship between the values of the process can be described by a covariance function $C(\boldsymbol{h})$, a function that depends only on the separation vector $\boldsymbol{h}$.

2.2.1 Covariance functions

Let $\boldsymbol{h} \in \mathbb{R}^r$ be a separation vector, i.e. a vector containing information regarding the distance between the sites $\boldsymbol{s}$ and $\boldsymbol{s} + \boldsymbol{h}$. Assuming that $\mathbb{E}[Y(\boldsymbol{s} + \boldsymbol{h}) - Y(\boldsymbol{s})] = 0$, consider

$$\mathbb{E}[Y(\boldsymbol{s} + \boldsymbol{h}) - Y(\boldsymbol{s})]^2 = \mathbb{V}\text{ar}(Y(\boldsymbol{s} + \boldsymbol{h}) - Y(\boldsymbol{s})) = 2\gamma(\boldsymbol{h}). \tag{2.2}$$

The quantity $2\gamma(\boldsymbol{h})$ is called the *variogram*, whereas $\gamma(\boldsymbol{h})$ is known as *semivariogram*. Equation (2.2) only makes sense if the left-hand side depends only on $\boldsymbol{h}$ and not on a specific choice of $\boldsymbol{s}$. A spatial process is said to be *intrinsically stationary* if it has a constant mean and the variance of the differences of Y at pairs of locations only depends on the separation vector $\boldsymbol{h}$. Assume that $\mathbb{C}\text{ov}(Y(\boldsymbol{s}), Y(\boldsymbol{s}+\boldsymbol{h})) = C(\boldsymbol{h})$, i.e. the covariance function depends only on the separation vector $\boldsymbol{h}$. According to Banerjee et al. (2004, ch. 2), the variogram and the covariance function are associated as follows:

$$\begin{aligned}
2\gamma(\boldsymbol{h}) &= \mathbb{V}\mathrm{ar}(Y(\boldsymbol{s}+\boldsymbol{h}) - Y(\boldsymbol{s})) \\
&= \mathbb{V}\mathrm{ar}(Y(\boldsymbol{s}+\boldsymbol{h})) + \mathbb{V}\mathrm{ar}(Y(\boldsymbol{s})) - 2\mathbb{C}\mathrm{ov}(Y(\boldsymbol{s}+\boldsymbol{h}), Y(\boldsymbol{s})) \\
&= \mathbb{C}\mathrm{ov}(Y(\boldsymbol{s}+\boldsymbol{h}), Y(\boldsymbol{s}+\boldsymbol{h})) + \mathbb{C}\mathrm{ov}(Y(\boldsymbol{s}), Y(\boldsymbol{s})) - 2\mathbb{C}\mathrm{ov}(Y(\boldsymbol{s}+\boldsymbol{h}), Y(\boldsymbol{s})) \\
&= 2[C(\boldsymbol{0}) - C(\boldsymbol{h})].
\end{aligned}$$

Therefore, we can write

$$\gamma(\boldsymbol{h}) = C(\boldsymbol{0}) - C(\boldsymbol{h}). \tag{2.3}$$

As stated in Banerjee et al. (2004, p. 23), if the semivariogram function $\gamma(\boldsymbol{h})$ depends upon the separation vector only through its length $\|\boldsymbol{h}\|$, then the process is said to be *isotropic*. In this case, $\gamma(\boldsymbol{h})$ is a real-valued function with a scalar argument $\|\boldsymbol{h}\|$. These kind of processes are simpler to interpret and assumed in many applications.

A number of parametric forms are available as candidates for the semivariogram. In general, from Equation (2.3) we can recover C from γ, but some semivariogram do not determine a covariance function C, since the limit $\lim_{\|\boldsymbol{h}\|\to\infty}\gamma(\boldsymbol{h})$ may not exist. We present bellow the notation and formulation of twelve different covariance functions associated with some important semivariograms available in the literature. Consider $t = \|\boldsymbol{h}\|$ and the following covariance functions (valid in all dimensions in the Euclidean space defined for the locations coordinates, unless stated otherwise):

1. Exponential:

$$C(t) = \begin{cases} \sigma^2 \exp\{-\phi t\} & \text{if } t > 0, \\ \tau^2 + \sigma^2 & \text{otherwise.} \end{cases}$$

   The parameter σ^2 is called the *partial sill*, ϕ is the *decay parameter* ($1/\phi$ is the *range parameter*), and τ^2 is the *nugget effect*.

2. Gaussian:

$$C(t) = \begin{cases} \sigma^2 \exp\{-\phi^2 t^2\} & \text{if } t > 0, \\ \tau^2 + \sigma^2 & \text{otherwise.} \end{cases}$$

3. Powered exponential:

$$C(t) = \begin{cases} \sigma^2 \exp\{-(\phi t)^p\} & \text{if } t > 0, \\ \tau^2 + \sigma^2 & \text{otherwise.} \end{cases}$$

The powered exponential covariance function presented above is valid if the shape parameter $p \in (0, 2]$. As can be seen, the exponential and the Gaussian covariance functions are particular cases of the powered exponential function.

4. Rational quadratic:

$$C(t) = \begin{cases} \sigma^2 \left(1 - \frac{t^2}{\phi + t^2}\right) & \text{if } t > 0, \\ \tau^2 + \sigma^2 & \text{otherwise.} \end{cases}$$

5. Wave:

$$C(t) = \begin{cases} \sigma^2 \frac{\sin(\phi t)}{\phi t} & \text{if } t > 0, \\ \tau^2 + \sigma^2 & \text{otherwise.} \end{cases}$$

The wave covariance function is valid in $\mathbb{R}^n$ under some conditions for the parameters, see Yaglom (1987). Note that this is not a monotonically increasing function; thus the Wave covariance structure allows us modeling situations where we suspect that an increase in the distance between two points might be associated with a covariance that increases, decreases and increases again, as shown in Figure 2.3.

6. Matérn:

$$C(t) = \begin{cases} \frac{\sigma^2}{2^{\nu-1}\Gamma(\nu)}(2\sqrt{\nu} t\phi)^\nu K_\nu(2\sqrt{\nu} t\phi) & \text{if } t > 0, \\ \tau^2 + \sigma^2 & \text{otherwise.} \end{cases}$$

The function $\Gamma(\cdot)$ is the usual gamma function and K_ν is the modified Bessel function of order ν. The parameter ν defines the shape of the covariance function; for details, see Abramowitz and Stegun (1965). This covariance function is widely used in the literature; as an example, Sun et al. (2015) model the spatial covariance structure of rain rates with the Matérn function using point-referenced meteoro-

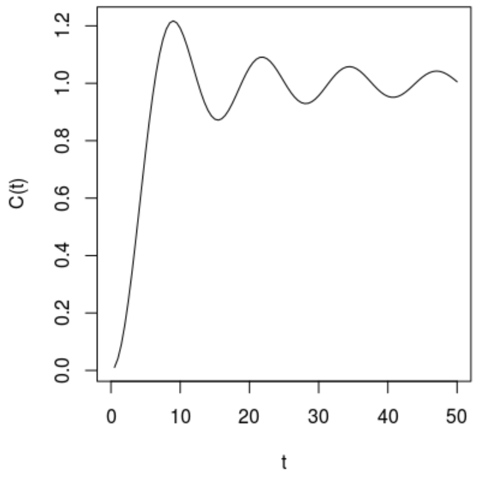


Figure 2.3: Wave covariance function with $\phi = 0.5$, $\tau^2 = 0$ and $\sigma^2 = 1$.

logical gauges.

7. Cauchy:

$$C(t) = \begin{cases} \sigma^2\big[1 + (\phi t)^2\big]^{-\kappa} & \text{if } t > 0, \\ \tau^2 + \sigma^2 & \text{otherwise.} \end{cases}$$

Here, κ is a smoothness parameter.

8. Generalized Cauchy:

$$C(t) = \begin{cases} \sigma^2\big[1 + (\phi t)^{\kappa_2}\big]^{-\kappa_1/\kappa_2} & \text{if } t > 0, \\ \tau^2 + \sigma^2 & \text{otherwise.} \end{cases}$$

This covariance function is valid in $\mathbb{R}^r$ when $\kappa_1 > 0$ and $0 < \kappa_2 \leq 2$. In this model, κ_1 controls the dependence at large distances, whereas κ_2 is the shape parameter. Note that the Cauchy covariance is a particular case of the generalized Cauchy function, when $\kappa_2 = 2$ and $\kappa_1 = 2\kappa$.

9. Spherical:
$$C(t) = \begin{cases} 0 & \text{if } t \geq 1/\phi, \\ \sigma^2\big[1 - \frac{3}{2}\phi t + \frac{1}{2}(\phi t)^3\big] & \text{if } 0 < t \leq 1/\phi, \\ \tau^2 + \sigma^2 & \text{otherwise.} \end{cases}$$
This covariance function is valid in $r = 1, 2$ or 3.

10. Circular: Let $\theta = \min\{\phi t, 1\}$ and $g(t) = 2\pi^{-1}\big(\theta\sqrt{1-\theta^2} + \sin^{-1}\sqrt{\theta}\big)$. The circular covariance function (valid in $\mathbb{R}^2$) is given by
$$C(t) = \begin{cases} \sigma^2\big[1 - g(t)\big] & \text{if } t < \phi, \\ \tau^2 + \sigma^2 & t = 0, \\ 0 & \text{otherwise.} \end{cases}$$

11. Cubic:
$$C(t) = \begin{cases} \sigma^2\big(1 - \big[7(\phi t)^2 - 8.75(\phi t)^3 + 3.5(\phi t)^5 - 0.75(\phi t)^7\big]\big) & \text{if } t < \phi, \\ \tau^2 + \sigma^2 \quad t = 0, \\ 0 \quad \text{otherwise.} \end{cases}$$
This covariance function is valid in $\mathbb{R}^3$.

12. Gneiting:
$$C(t) = \begin{cases} \sigma^2\big[1 + 8st + 25s^2t^2 + 32s^3t^3)(1 - st)^8\big] & \text{if } 0 < t < 1/s, \\ \tau^2 + \sigma^2 \quad t = 0, \\ 0 \quad \text{otherwise.} \end{cases}$$
Here, $s = 0.301187465825$. For further details see documentation of the R package `RandomFields`, Schlather et al. (2015).

The reader may find other configurations for the covariance functions previously showed. We are using those discussed in Banerjee et al. (2004) and Ribeiro Jr and Diggle (2016). In those functions, the value of $\tau^2 + \sigma^2$ is the variance in a point of the space and the parameter ϕ is related to the point above which the spatial variance can be neglected, this is known as the *range*.

2.2.2 Empirical variogram

Having a large selection of models for the covariance structure can make the user wonder how to choose one of them for a given data set. The covariance function can be chosen by plotting the *empirical semivariogram*, an estimate of the semivariogram, and comparing it to the theoretical shapes available from the choices listed above. According to Banerjee et al. (2004, p. 29, 30) the empirical semivariogram is defined as follows:

$$\hat{\gamma}(t) = \frac{1}{2|N(t_k)|} \sum_{(\mathbf{s_i}, \mathbf{s_j}) \in N(t_k)} \left[Y(\mathbf{s_i}) - Y(\mathbf{s_j})\right]^2,$$

where we divide the t-space into intervals $I_1 = (0, t_1), I_2 = (t_1, t_2), \ldots, I_K = (t_{K-1}, t_K)$, for some grid $0 < t_1 < \ldots < t_K$. Consider that $N(t_k) = \{(\mathbf{s_i}, \mathbf{s_j}) : \|\mathbf{s_i} - \mathbf{s_j}\| \in I_k\}$, for $k = 1, \ldots, K$, and $|N(t_k)|$ is the number of pairs of points in interval I_k.

We are interested in using the previously listed covariance functions in the Bayesian context with the support and resources from `JAGS`. Our main goal is to include all mentioned covariance functions in our `JAGS` module to simplify the point-referenced data analysis in a Bayesian model to be implemented via `JAGS`.

2.3 Areal data

We present now a modeling approach that is commonly applied to data collected for areal units. In general, areal data represent irregular geographic regions, such as municipalities. However, regular grids of cells can also be modeled (e.g. image pixels). Within this context, we are interested in investigating whether there is any spatial pattern in the study region and, if so, how strong it is. By spatial pattern we mean that areal units close to each other will take more similar values than those for units far from each other. This is known as the first law in Geography (see Tobler, 1970). Occasionally, this pattern can be visually identified, however we must determine its intensity. In the case of independent measurements, we expect to see no pattern at all.

In order to introduce spatial association, we define a neighborhood structure based on the arrangement of the blocks in a map through the proximity matrix.

Definition 2.1 (Proximity matrix)**.** Given the vector of measurements $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_n)$ associated with areal units $1, \ldots, n$, the proximity matrix $\boldsymbol{W}$ is such that entries w_{ij} spatially connect units i and j in some way.

Possibilities for $\boldsymbol{W}$ involve binary choices ($w_{ij} = 1$ if i and j share common border). As another option, w_{ij} could indicate a function of the distance between units (a decreasing function of distance between i and j). Alternatively, we can represent the distance in a binary fashion, setting $w_{ij} = 1$ if units i and j lie within a specific distance apart. The previous suggestions imply that $\boldsymbol{W}$ would be symmetric. In fact, we may consider $\widetilde{\boldsymbol{W}}$, standardizing the proximity matrix by $\sum_j w_{ij} = w_{i+}$. In this case, $\widetilde{\boldsymbol{W}}\mathbf{1} = \mathbf{1}$, which means that $\widetilde{\boldsymbol{W}}$ is row stochastic. The entries in $\boldsymbol{W}$ can be seen as weights: the farther unit j is from i the lower the weight in entry w_{ij}. According to Banerjee et al. (2004), $\boldsymbol{W}$ provides a mean for introducing spatial structure into statistical models.

2.3.1 Measures of spatial association

With the spatial structure defined, we can determine whether there is spatial association. Banerjee et al. (2004, p. 71) mention Moran's I and Geary's C statistics to evaluate the intensity of spatial association among areal units. The Moran's I is defined as

$$I = \frac{n \sum_i \sum_j w_{ij}(\theta_i - \overline{\theta})(\theta_j - \overline{\theta})}{\left(\sum_{i \neq j} w_{ij} \right) \sum_i (\theta_i - \overline{\theta})^2}.$$

I is not strictly supported on the $[-1, 1]$ interval and, under the null model (θ_i are i.i.d), I is asymptotically normally distributed with mean $\frac{-1}{n-1}$. Geary's C is defined as

$$C = \frac{(n-1)\sum_i \sum_j w_{ij}(\theta_i - \theta_j)^2}{2\big(\sum_{i \neq j} w_{ij}\big)\sum_i (\theta_i - \bar{\theta})^2}.$$

C is non-negative and low values (between 0 and 1) indicate positive spatial association.

2.3.2 Markov random fields

As a result of the Bayes' theorem, given the joint distribution $p(\boldsymbol{\theta})$, the full conditional distributions $p(\theta_i \mid \theta_{j \neq i})$, $i = 1, \ldots, n$ can be uniquely determined. One may wonder about the converse: does $p(\theta_i \mid \theta_{j \neq i})$ determines $p(\boldsymbol{\theta})$? Besag showed that the result from Brook's Lemma (see Banerjee et al., 2004, p. 76) allows us to retrieve the unique joint distribution determined by the full conditionals, as follows:

$$\begin{aligned} p(\boldsymbol{\theta}) = \frac{p(\theta_1 \mid \theta_2, \ldots, \theta_n)}{p(\theta_{10} \mid \theta_2, \ldots, \theta_n)} \times \frac{p(\theta_2 \mid \theta_{10}, \theta_3, \ldots, \theta_n)}{p(\theta_{20} \mid \theta_{10}, \theta_3 \ldots, \theta_n)} \times \\ \cdots \times \frac{p(\theta_n \mid \theta_{10}, \ldots, \theta_{n-1,0})}{p(\theta_{n0} \mid \theta_{10}, \ldots, \theta_{n-1,0})} \times p(\theta_{10}, \ldots, \theta_{n0}). \end{aligned} \tag{2.4}$$

This equality holds if the set of full conditional distributions determine a unique and valid joint distribution (in this case, we say the full conditionals are *compatible*) and each conditional is proper (with integrable density function). Apart from the constant term $\boldsymbol{\theta_0} = p(\theta_{10}, \ldots, \theta_{n0})$ in the right side of (2.4), in the case this distribution is proper, the joint distribution can be determined up to a proportionality constant.

As stated in Banerjee et al. (2004), when the number of regions in the study area is large, it is simpler to work with the full conditional distributions. Moreover, it is reasonable to consider that the full conditional distribution for θ_i would depend only upon the neighbors of region i. Let δ_i denote the set of neighbors of unit area i and consider

$$p(\theta_i \mid \theta_j, j \neq i) = p(\theta_i \mid \theta_j, j \in \delta_i). \tag{2.5}$$

This sort of specification for the full conditional distributions, when compatible, is referred to as a *Markov random field* (MRF). The following definitions are essential for understanding the remaining concepts of this section.

Definition 2.2 (Clique)**.** A clique is a set of units such that each element in the set is a neighbor of every other element in the set.

The proximity matrix can be depicted as a graph and thus a clique represents a set of nodes M on the graph such that each pair of indices (i, j), with both i and j in M, represents an edge on the graph. In a study area with n spatial units, we can have cliques of size $1, \dots, n$.

Definition 2.3 (Potential function)**.** A potential of order k is a function of k arguments that is exchangeable in its arguments.

The arguments of the potential function would be the values taken by the variables associated with the cells for a clique of size k. Examples include $\theta_i\theta_j$ and $(\theta_i - \theta_j)^2$ if i and j are a clique of size 2.

Definition 2.4 (Gibbs distribution)**.** $p(\theta_1, \dots, \theta_n)$ is a Gibbs distribution if it takes the form

$$p(\theta_1, \dots, \theta_n) \propto \exp\left\{\gamma \sum_k \sum_{\boldsymbol{\alpha} \in \mathcal{M}_k} \phi^{(k)}(\theta_{\alpha_1}, \theta_{\alpha_2} \dots, \theta_{\alpha_k})\right\},$$

where $\phi^{(k)}$ is a potential of order k, $\mathcal{M}_k$ is the collection of all subsets of size k of the set $\{1, \dots, n\}$, $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_k)^\top$ is a subset of this collection and $\gamma > 0$ is a scale parameter.

According to Banerjee et al. (2004, p. 78), the Hammersley-Clifford Theorem demonstrates that if $p(\theta_i \,|\, \theta_{j \in \delta_i})$ determines a unique $p(\theta_1, \dots, \theta_n)$, i.e we have a MRF, then this joint distribution is a Gibbs distribution. In addition, Geman and Geman (1984) provide the converse of this theorem. That is, determine a MRF from the Gibbs distribution. As a result, we can use the full conditionals to sample from the Gibbs distribution.

A common choice for the joint distribution is the pairwise difference that takes the form

$$p(\theta_1, \dots, \theta_n) \propto \left\{ -\frac{1}{2\tau^2} \sum_{i,j} (\theta_i - \theta_j)^2 I(i \sim j) \right\},$$

where $I(i \sim j)$ indicates whether i and j are neighbors. Note that this is a Gibbs distribution on potentials of order 1 and 2. Moreover,

$$p(\theta_i \mid \theta_j, j \neq i) = p(\theta_i \mid \theta_{j\in\delta_i}) = N\left(\sum_{j\in\delta_i} \theta_i/m_i, \tau^2/m_i\right), \tag{2.6}$$

with m_i being the number of neighbors of region i. The distribution in (2.6) is a MRF and shows that the mean of θ_i is an average of its neighbors.

2.3.3 Conditionally autoregressive (CAR) model

It was initially proposed by Besag (1974) with two approaches: the Gaussian and the non-Gaussian cases. Here, we will focus our attention only on the former case, also referred to as *autonormal*. This model is computationally convenient for Gibbs sampling and other Markov chain Monte Carlo (MCMC) and is applied to introduce spatial random effects in a second stage of a hierarchical specification.

Considering the case when the number of unit areas is large, our aim is to write the full conditionals instead of the joint distribution. Again, intuitively, it is reasonable to think that $p(\theta_i \,|\, \theta_{j\neq i})$, the full conditional distribution for θ_i of region i, will depend only upon its neighbors. Suppose we set, for $i = 1, ..., n$,

$$\theta_i|\theta_{j\neq i} \sim N\Big[\sum_j b_{ij}\theta_j, \tau_i^2\Big].$$

As stated in Banerjee et al. (2004, p. 79), the full conditionals above are compatible and through Brook's lemma we can write the joint distribution

$$p(\theta_1, \dots, \theta_n) \;\propto\; \exp\bigg\{ -\frac{1}{2}\boldsymbol{\theta}^\top D^{-1}(I-B)\boldsymbol{\theta}\bigg\}, \tag{2.7}$$

where $B = \{b_{ij}\}$ and $D = \mathrm{diag}(\tau_1^2, \dots, \tau_n^2)$. Expression (2.7) resembles a joint multivariate normal distribution for $\boldsymbol{\theta}$ with probability density given by

$$f(\theta_1, \dots, \theta_n) = \frac{1}{\sqrt{2\pi|\boldsymbol{\Sigma}_\theta|}} \exp\bigg\{ -\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}_\theta^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\bigg\},$$

with null mean vector $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}_\theta$ would be the covariance matrix given by inverting $D^{-1}(I-B)$. But first, we need to ensure that $\boldsymbol{\Sigma}_\theta$ is symmetric, i.e. ${}^{b_{ij}}/_{\tau_i^2} = {}^{b_{ji}}/_{\tau_j^2}$, for all i, j. Recall the proximity matrix in Definition 2.1. By setting $b_{ij} = {}^{w_{ij}}/_{w_{i+}}$ and $\tau_i^2 = {}^{\tau^2}/_{w_{i+}}$ we guarantee the symmetry of $\boldsymbol{\Sigma}_\theta$ and thus Expression (2.7) becomes

$$p(\theta_1, \ldots, \theta_n) \; \propto \; \exp\left\{ -\frac{1}{2\tau^2}\boldsymbol{\theta}^\top(\boldsymbol{D}_w - \boldsymbol{W})\boldsymbol{\theta} \right\}, \tag{2.8}$$

where $\boldsymbol{D}_w = \text{diag}(w_{1+}, \ldots, w_{n+})$. However, in Expression (2.8), $\boldsymbol{D}_w - \boldsymbol{W}$ is singular, since each of its rows sum up to zero and thus the distribution in (2.8) is improper. This aspect can be solved by redefining $\boldsymbol{\Sigma}_\theta^{-1} = D_w - \rho\boldsymbol{W}$ and choosing a value ρ to make the indicated $\boldsymbol{\Sigma}_\theta^{-1}$ nonsingular. Banerjee et al. (2004) state the conditions to choose this parameter: $\rho \in \left({}^1/_{\lambda_{(1)}}, {}^1/_{\lambda_{(n)}}\right)$, where $\lambda_{(1)} < \lambda_{(2)} < \cdots < \lambda_{(n)}$ are the ordered eigenvalues of $\boldsymbol{D}_w^{-1/2}\boldsymbol{W}\boldsymbol{D}_w^{-1/2}$. Setting $\rho = 0$ leads to θ_i being independent normally distributed random variables with null mean and variance ${}^{\tau^2}/_{w_{i+}}$, and the full conditional $p(\theta_i|\theta_j, j \neq i)$ becomes $N(0, {}^{\tau^2}/_{w_{i+}})$.

We are interested in the CAR model presented here in the Bayesian context with the support and resources from `JAGS`, delivering a simple method to use the CAR distribution in `JAGS`, reducing the complicated computations related to the covariance matrix in this model.

Chapter 3

Extending JAGS

Bayesian inference is an important approach in statistics, in which the Bayes' rule is applied to update the probability of an event as data information becomes available. This type of inference became very popular in the past decades with the advent of modern computers capable of executing computationally demanding algorithms. This chapter discusses the use of the Bayesian inference in spatial analyses through the softwares of the `BUGS` family and `JAGS`. This chapter is divided into four sections. Section 3.1 introduces the `BUGS` project and highlights the main differences between `JAGS` and the members of the `BUGS` family. Section 3.2 demonstrates how a hierarchical Bayesian model can be fit using `JAGS`. Section 3.3 illustrates steps required to build a custom module in `JAGS`. Ultimately, Section 3.4 presents our extension to `JAGS`, which features all the covariance functions and the CAR distribution defined in the previous chapter.

3.1 The `BUGS` project

Bayesian inference is an alternative to the frequentist inference. We must specify the model with the prior distributions expressing our knowledge about the parameters before observing the data. In addition, in many situations the Gibbs sampling algorithm is used to sample from unknown joint posterior distributions. In this case, we shall compute the posterior full conditionals, which can be very challenging for users who are not used to the mathematical computations related to the Bayes' rule. Once these posterior full conditionals are determined, one can apply the Markov chain Monte Carlo (MCMC) algorithm to draw a posterior sample for each parameter. Again, the implementation of the Gibbs sampler algorithm is not trivial and thus not attractive for most users. The `BUGS` project (started in 1989) and `JAGS` (initially proposed in 2003) are good alternatives to work with the Gibbs sampling because they build the whole algorithm without much effort from the user. The statistical model is specified by simply stating the prior distributions and the likelihood function. According to Lunn et al. (2009), these statistical softwares have been fundamental in raising attention to Bayesian modeling among both academic and commercial communities. Two well known members of the `BUGS` family are:

- `WinBUGS`, released in 1997, it uses a standard 'point-and-click' windows interface for controlling the analyses. It runs exclusively on the Windows platform;
- `OpenBUGS`, released in 2005, the open-source version of the core `BUGS` code with a variety of interfaces. It runs on Windows, with a similar graphical interface to `WinBUGS`, and on Linux also with a plain-text interface.

The last release of `WinBUGS` was in August 2007. The alternative `OpenBUGS` had its last release in 2014. Although `OpenBUGS` is open-source, it was written in Component Pascal, a programming language that is not widely used as `C++`. An attractive alternative to the previous options is the `C++` open-source version of the `BUGS` family, named `JAGS` (Just Another Gibbs Sampler); see Plummer (2003). This option has almost the same syntax language of the two other softwares. An important feature is the fact that it was written to run on the Unix platform; it also runs on Windows. In addition, it is modular extensible, meaning that users can implement custom functions, distributions and samplers. Similar to `OpenBUGS` and its interfaces to the `R` software through `R2OpenBUGS` (Sturtz et al., 2005) and `rbugs` (Yan and Prates, 2013), the `JAGS` software can also be used within `R`, through the packages `R2jags` (Su and Yajima, 2015) and `rjags` (Plummer, 2016). These packages allow the user to take advantage of all the resources in `R` for creating graphics and descriptive analyses using the outputs from `OpenBUGS` and `JAGS`.

Whilst `WinBUGS` and `OpenBUGS` have an add-on that provides some features for spatial statistics, as far as the authors are concerned, there is no spatial module in `JAGS`, which can be seen by some practitioners as a deficiency making `JAGS` the last choice for a spatial analysis. This fact motivated us to create a novel spatial extension to `JAGS`, a module containing key tools for the analyses of point-referenced and areal data.

3.1.1 The `GeoBUGS` module

This add-on to `WinBUGS` and `OpenBUGS` is an interface for spatial modelling, developed by a team at the Department of Epidemiology and Public Health of Imperial College at St Mary's Hospital in London. Although this spatial module is part of `OpenBUGS`, its source code is not freely available for developers.

Concerning point-referenced data, only two covariance structures are available within this package. The first one is the powered exponential (defined in Section 2.2.1). The other one is called *disc* model and is defined as

$$C(t) = \begin{cases} \frac{2\sigma^2}{p\{\cos^{-1}(t/\phi) - [(t/\phi)(1-t^2/\phi^2)]^{1/2}\}} & \text{if } t < \phi, \\ 0 & \text{if } t > \phi. \end{cases}$$

Apart from these two structures, if the user wants to work with other covariance matrices, they must build it within the `BUGS` model script, which can be seen as a difficult task, depending on their programming skills.

Regarding areal data, `GeoBUGS` implements both the proper and the improper versions of the CAR distribution. In this setting the user might be requested to provide values for the arguments of the CAR function, such as a vector listing the identification numbers of the adjacent areas for each area, a vector giving unnormalized weights associated with each pair of areas, a vector giving the number of neighbors for each area, and an scalar argument representing the precision of the distribution.

3.2 Bayesian Inference with JAGS

The central idea in this section is not to present a meaningful statistical model, instead we demonstrate how a hierarchical model is set in `JAGS`, briefly describing the commands needed to write a script for a statistical model.

As stated in Plummer (2003), the architecture that the `BUGS` language uses is based on graphs. The `BUGS` language provides its own form to specify statistical models which mimics the mathematical specification of the model in terms of parent–child relationships. The first step in the construction of a Bayesian graphical model is to describe the conditional independence relationships between the variables using a directed acyclic graph (DAG). The complex joint distribution is thus broken down into a set of simpler parent-child relationships. The fundamental object of this language is the node, which represents a variable in the model. In addition, a model script associates a graph with a set of samplers and a set of monitors. A sampler defines a method of updating a graph, such as the Gibbs sampler. A monitor records the sampled values and summarizes them (the *trace monitor* records the sampled value every n^{th} iteration). Stochastic relationships are denoted by '$\sim$' whereas deterministic relationships are denoted with a '`<-`'. Repetitive structures are represented using for-loops. Moreover, the `BUGS` language is a declarative language, which means that it does not matter in which order the various statements are made.

Consider the following simple model

$$\begin{aligned} Y_i \mid \mu, \theta_i, \sigma^2 &\sim N_n(\mu + \theta_i, \sigma^2), \\ \boldsymbol{\theta} &\sim N_n(\boldsymbol{\mu}_\theta, \boldsymbol{\Sigma}^{-1}), \\ \sigma^2 &\sim IG(a_\sigma, b_\sigma), \\ \mu &\sim N(m, v), \end{aligned} \tag{3.1}$$

where $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_n)^\top$ and $\boldsymbol{\Sigma}$ is the precision matrix. With some adjustments, the model above can be used to fit a data set with point-referenced information such as the Belo Horizonte elevation data shown in Figure 2.1 (see Section 4.3 for a detailed example).

Box 3.1: JAGS script using the exponential covariance function without the module.

```
model {
  for (i in 1:N) {
    Y[i] ~ dnorm (mu_y + theta[i], tau2)
  }
  theta ~ dmnorm (mu_theta, Sigma)
  mu_y ~ dnorm (m , v)
  tau2 ~ dgamma (a_tau, b_tau)
  sig2 <- 1 / tau2
}
```

Any `JAGS` script can be executed with the command line interface in the Unix terminal and also with its interfaces to the `R` software. The script version of the model indicated in (3.1) is shown in Box 3.1. For this model, the user is required to provide values for the parameters `mu_theta`, `Sigma`, `m`, `v`, `a_tau`, `b_tau` and the response variable `Y[1:N]`. Note that here we use the reciprocal of the gamma distribution to sample from the inverse gamma distribution. In addition, the univariate and the multivariate normal distribution, respectivelly `dnorm`$(\cdot,\cdot)$ and `dmnorm`$(\cdot,\cdot)$, are parameterized in terms of the precision.

After the model has been defined, it must be compiled. According to Plummer (2003), it is also at this stage that the provided data is combined with model script to create a graphical model. Syntax errors in the model specification and possible directed cycles will produce compilation error. Before the model is run, it must be initialized by either providing initial values for the unknown parameters or by defining a random number generator. In the subsequent step, we define the initial burn-in period for the MCMC output chain: all values sampled at this stage are discarded, since during this period we consider that the chains have not yet converged to the target distribution. Finally, we define the size of the MCMC chains and run the model script in `JAGS`. An analysis of the resulting chains, including the posterior means and highest posterior density (HPD) intervals, can be done with the support of `R` and its packages, e.g. `coda` (see Plummer et al., 2006).

3.3 Adding custom functions to JAGS

A central study taken as starting point for our work is Wabersich and Vandekerckhove (2014), which demonstrates how to create custom distributions in `JAGS` for univariate random variables. An important aspect not explored in this reference, which can be seen as one of the contributions of our work, is the fact that it is possible for the user to create a `JAGS` module that only contains functions and does not implement any sampler for a probability distribution. This feature was used to implement the many spatial functions evaluated ahead. Such functions are used to build covariance matrices to be inserted in the multivariate normal sampler already available in `JAGS`.

In this section, we describe the steps needed to build new functions in `JAGS`. It is important to remark that differently from what is described in Wabersich and Vandekerckhove (2014), the functions implemented in our proposed module does not yield a scalar value, instead they give covariance or precision matrices. As an illustration of the process of development, we will focus on the Matérn covariace function. Basic knowledge of the `C++` programming language is required for a complete understanding. Once created, we can dynamically load the custom module to extend the `JAGS` functionality.

We need to specify two `C++` classes to create a new `JAGS` function; the first one is for the module itself, let us call it `GeoJAGS` (its content is shown in Box 3.2). This file should be stored in the `src/` directory, a subfolder of the working directory for the project. Again, in this tutorial we are demonstrating how to develop a single function for `JAGS`. More functions can be added within the constructor function (line 11), all newly developed functions for `JAGS` must be inserted here by creating a new object of the corresponding class, as shown in line 12.

Box 3.2: The `GeoJAGS.cc` module definition

```
1  #include <module/Module.h>   // include JAGS module base class
2  #include <functions/cvmatern.h>
3  namespace GeoJAGS { // start defining the module namespace
4    // Module class
5    class GeoModule : public Module {
6      public:
7        GeoModule();   // constructor
8        ~GeoModule(); // destructor
9    };
10   // Constructor function
11   GeoModule::GeoModule() : Module("GeoJAGS") {
12     insert(new CVMatern); // load the new function into JAGS
13   }
14   // Destructor function
15   GeoModule::~GeoModule() {
16     std::vector<Function*> const &fvec = functions();
17     for (unsigned int i=0; i<fvec.size(); ++i) {
18       delete fvec[i]; // delete all instantiated functions objects
19     }
20   }
21 } // end namespace definition
22 GeoJAGS::GeoModule _GeoJAGS_module;
```

Box 3.3: The Matérn covariance function class header file `cvmatern.h`

```
#ifndef CVMatern_FUN_H_
#define CVMatern_FUN_H_
#include <function/ArrayFunction.h> //the base class used

using namespace jags;
namespace GeoJAGS {
  class CVMatern : public ArrayFunction {
    public :
      CVMatern(); //the constructor function
      void evaluate(double *value, std::vector<double const *> const &args,
         std::vector<std::vector<unsigned int> > const &dims) const;
      bool checkParameterDim(std::vector<std::vector<unsigned int> > const &dims) const;
      bool checkParameterValue(std::vector<double const *> const &args,
                               std::vector<std::vector<unsigned int> > const &dims) const;
      std::vector<unsigned int> dim(std::vector <std::vector<unsigned int> > const &dims,
        std::vector <double const *> const &values) const;
  };
}
#endif /* CVMatern_FUN_H_ */
```

Now we must define the Matérn covariance function, which returns a covariance matrix according to the definition of the function in Section 2.2.1. A new `JAGS` function is implemented through a new `C++` class. Since the return value of the new function is a matrix object, we set its parent class to be of the type `ArrayFunction`. Thus, the new class should contain four inherited functions, together with its own constructor:

- `evaluate`: routine to apply calculations using the input values of the covariance function and returns a vector structure containing this matrix. Because the `JAGS` programming structure in `C++` does not handle matrices, the covariance matrix should be returned in a $(1 \times n^2)$-dimensional vector;
- `checkParameterDim`: checks whether dimensions of the arguments of the covariance function are correct, in other words, if one argument should be real-valued, this function must perform this verification;
- `checkParameterValue`: checks whether each of the argument values lie in the domain of the corresponding parametric space;
- `dim`: calculates the dimension of the return value, based on the arguments. In any covariance function, the output value should be a squared matrix of order n (the number of observations), therefore, the `dim` function must return the vector (n, n).

Box 3.3 contains the file `/src/functions/cvmatern.h`, which includes the prototypes of the constructor and the four required functions previously listed.

The `evaluate` function takes three input arguments:

- The first argument (`value`) is the array of doubles which contains the result of the evaluation on the exit. In our case, this vector must contain the covariance matrix.
- The second argument (`args`) contains pointers that reference the arguments of the function.
- The argument `dims` contains the corresponding dimensions of each element of `args`.

The arguments of the remaining functions have a straightforward interpretation, just follow the ones described for the `evaluate` function. The implementations of the required functions in Box 3.3 are provided in Box 3.4. Line 6 defines the constructor function for our class. It calls the constructor for `ArrayFunction` with two input arguments. The first one is used to name the function and the second one is used to define its number of arguments. For the Matérn covariance function, it should contain five arguments: a matrix with the locations coordinates and values for σ^2, ϕ, τ^2 and ν. Note that the representation of matrices adopted in `JAGS` is made by row vectors (line 15), with the first n elements being the first column of the matrix and so forth.

In order to compile the code, build and install the module we need to create some configuration files such as the `configure.ac` in the main directory and a `Makefile.am` in every directory of the project. These files contain information on correct configuration and the necessary arguments to build the project. Full details can be found in Wabersich and Vandekerckhove (2014) at step 3 on page 18.

3.4 The `GeoJAGS` module

Considering the description of the functions for spatial data models given in Chapter 2, one can clearly see that performing all computations of covariance and precision matrices is not a trivial task, especially for those professionals not familiar with programming. In fact this is a considerable group, including researches from social sciences, medical sciences, natural sciences to name a few. From this perspective, our proposed module arises as an alternative that simplifies the work of writing `JAGS` scripts for models dealing with point-referenced data and areal data.

The `GeoJAGS` module is currently available for Mac and Linux users and can be

Box 3.4: The `cvmatern.cc` file

```cpp
#include <config.h>
#include "cvmatern.h"
#include <cmath> //basic math operations
#include <JRmath.h>
namespace GeoJAGS {
CVMatern::CVMatern() : ArrayFunction("CVMatern", 5) {}
void CVMatern::evaluate(double *value, std::vector<double const *> const &args,
     std::vector<std::vector<unsigned int> > const &dims) const{
  int n = dims[0][0], m = dims[0][1], count = 0;
  std::vector<std::vector<double> > coords(n);
  std::vector<std::vector<double> > c(n);
  double sigma2 = args[1][0], phi = args[2][0], tau2 = args[3][0], nu = args[4][0];
  double gamma, bessel, dist; //the distance from two coordinates
  for (int i = 0; i < n; i++){ c[i].resize(n); coords[i].resize(m); }
  for (int j = 0; j < m; j++){
    for (int i = 0; i < n; i++){ coords[i][j] = args[0][count]; count++; }
  }
  for (int i=0; i<n; i++) {
    for (int j=i; j<n; j++) {
      if (i==j){ c[i][j] = tau2 + sigma2; }
      else{
        dist = 0; //distance computation
        for(int k=0; k<m; k++){
          dist += (coords[i][k] - coords[j][k])*(coords[i][k] - coords[j][k]);
        }
        dist = sqrt(dist); gamma = tgamma(nu);
        bessel = bessel_k(2*sqrt(nu)*dist*phi, nu, 1);
        c[i][j] = (sigma2 / (pow(2, nu-1) * gamma)) * pow(2*sqrt(nu)*dist*phi,nu) * bessel;
        c[j][i] = c[i][j];
      }
    }
  }
  //creates a vector that contains the matrix as a return object
  for (int i = 0; i < n; ++i) {
    value[i*n + i] = c[i][i];
    for (int j = 0; j < i; ++j) {
        value[i*n + j] = c[j][i]; value[j*n + i] = c[j][i];
    }
  }
}
bool CVMatern::checkParameterDim(std::vector<std::vector<unsigned int> > const &dims) const{
  int n = dims[0][0], m = dims[0][1], dim_sigma2 = dims[1][0], dim_phi = dims[2][0],
        dim_tau2 = dims[3][0], dim_nu = dims[4][0];
  return (n > 1  && m >= 1 && dim_sigma2 == 1 && dim_phi == 1 && dim_tau2 == 1 && dim_nu == 1);
}
bool CVMatern::checkParameterValue(std::vector<double const *> const &args,
                            std::vector<std::vector<unsigned int> > const &dims) const{
  double sigma2 = args[1][0], phi = args[2][0],  tau2 = args[3][0], nu = args[4][0];
  return (sigma2 > 0 && phi > 0 && tau2 > 0 && nu > 0);
}
std::vector<unsigned int> CVMatern::dim(std::vector <std::vector<unsigned int> > const &dims,
      std::vector <double const *> const &values) const{
  std::vector<unsigned int> n = dims[0];
  dim[0] = n[0];  dim[1] = n[0];
  return dim;
}
}
```

downloaded at `geojags.sourceforge.net` together with installation instructions. An installation wizard for Windows users is still under development and will be available soon at the same address. In the next subsections, we present a general description of the functions currently available in the `GeoJAGS` module. A full comprehensive list of all functions available in the module is detailed in Appendix A.

3.4.1 Covariance functions for point-reference data models

In the previous section, we described how to implement a custom function in `JAGS`, using as an example the Matérn covariance function. All the covariance functions described in Section 2.2.1 were implemented using a similar structure.

At this time, we have available in our `GeoJAGS` module twelve functions that builds covariance structures for the point-referenced data case: Matérn, exponential, Gaussian, powered exponential, rational quadratic, wave, Cauchy, generalized Cauchy, spherical, circular, cubic, and Gneiting. These functions in the module were designed to be used along with the multivariate normal distribution, with the matrix given as the return value representing the covariance matrix of this distribution.

3.4.2 The CAR model

Regarding areal data, `GeoJAGS` provides a simple approach to use the CAR model in any `JAGS` script. Here, we sample from the joint distribution instead of using the conditionals $p\left(\theta_i \mid \theta_j, j \neq i\right)$. In fact, `JAGS` currently does not allow the implementation of such conditionals, for it is not able to cope with the cycles induced by the distribution (see example below). The following example illustrates this situation.

Example (Cycle involving the random effects)**.** For simplicity purpose, consider two neighbor regions in which we desire to model the spatial random effects vector $\boldsymbol{\theta} = (\theta_1, \theta_2)$ using the CAR model. As stated in Section 2.3.3, the full conditionals are

$$p(\theta_1 \mid \theta_2) \sim N\big(\rho\theta_2, \tau^2\big),$$
$$p(\theta_2 \mid \theta_1) \sim N\big(\rho\theta_1, \tau^2\big).$$

Thus, the full conditionals of each random effect depend on the random effect of its neighbor. During the period of development of the `GeoJAGS` module, we found out that `JAGS` does not allow this kind of dependence between the variables. This generates an error message indicating possible directed cycle involving some or all of the nodes. ■

Considering the fact the CAR distribution is basically a multivariate normal distribution with a particular precision matrix, our module offers to the users a function that performs all the computations needed to build the non-singular precision matrix $\mathbf{\Sigma}_\theta$. Therefore, when planning to use the CAR model in a `JAGS` scrip, the user is required to set the distribution of the random effects vector to be a multivariate normal distribution with precision matrix defined by the function `precMatrixCAR` from our module. Here, the parameters τ^2, ρ, and the proximity matrix should be indicated accordingly. For users migrating from `OpenBUGS` to `JAGS`, we will offer a function with parameters similar to those implemented for the CAR in `GeoBUGS` (see details in Appendix A).

Sampling from the multivariate normal distribution can be very time consuming, especially if the number of regions is large. Algorithm 3.1 demonstrates the computations performed by `JAGS` in order to get a sample of size n from the multivariate normal. In terms of execution time, the algorithm to find the eigen-decomposition has the same complexity of matrix multiplication, which is $\mathcal{O}(n^{2.376})$, according to Coppersmith and Winograd (1987). Algorithm 3.1 is executed in every iteration of the MCMC whenever a multivariate normal variable is in the model script. A plausible alternative to remediate this time consuming procedure arises from the result of the following theorem, taken from Beezer (2008).

Algorithm 3.1 Sample from the multivariate normal distribution

```
1: generate $n$ values from $\mathrm{N}(0,1)$;
2: find the eigen-decomposition of the covariance matrix;
3: for each of the $n$-samples do
4:     scale by the square-root of the corresponding eigenvalue;
5: end for
6: rotate the vector of samples by pre-multiplying the scaled vector by the orthonormal
   matrix found by the decomposition.
```

Theorem 3.1 (Eigenvalues of a scalar multiple of a matrix)**.** *Suppose W is a square matrix and λ is an eigenvalue of W. Then $\alpha\lambda$ is an eigenvalue of αW.*

Proof. Let $\boldsymbol{x} \neq \mathbf{0}$ be one eigenvector of A for λ. Then

$$\begin{aligned}
(\alpha W)\boldsymbol{x} &= \alpha\,(W\boldsymbol{x}) && \text{(scalar matrix multiplication)}\\
&= \alpha\,(\lambda\boldsymbol{x}) && \text{(definition of eigen-values and -vectors)}\\
&= (\alpha\lambda)\,\boldsymbol{x} && \text{(scalar multiplication associativity)}
\end{aligned}$$

So $\boldsymbol{x} \neq \mathbf{0}$ is an eigenvector of λW with the eigenvalue $\lambda\alpha$. □

Recall the fact that there is no manual for developers in `JAGS`. The best material available in the literature is Wabersich and Vandekerckhove (2014), which can be seen as a short tutorial on creating extensions to the `JAGS`. In fact, this paper guides the development of univariate functions and distributions only. Based on some instructions in the mentioned paper and after reading part of the `JAGS` source code, we implemented a version of the multivariate normal, which receives the eigen-decomposition as an argument (replacing the usual mean vector and covariance matrix). Details of this implementation can be found in Appendix B. In this setting, the user is required to provide the eigen-values and -vectors of the matrix $\boldsymbol{D_w} - \rho\boldsymbol{W}$, where $\boldsymbol{D_w}$, $\boldsymbol{W}$ and ρ are defined in Section 2.3.3. Internally, this function will skip the more time consuming step in Algorithm 3.1 (Line 2), due to the eigen-decomposition being given as argument. We use the result from Theorem 3.1 to get the eigen-values and -vectors of $\tau^2(\boldsymbol{D_w} - \rho\boldsymbol{W})$. In the CAR model, τ^2 is considered to be random and $\boldsymbol{D_w} - \rho\boldsymbol{W}$ is a constant matrix. Note that here, we are considering the parameter ρ to be fixed.

Table 3.1 shows the execution times of a model script used as an example (see Box 3.5). Here, we simply sample from the multivariate normal distribution using three implementations. The execution times related to different dimension sizes are shown in the columns of Table 3.1. We use the conventional multivariate normal distribution sampler (`dmnorm`), our implemented version of this distribution as well (`dmnorm2`), and the version using the result from Theorem 3.1 (`eigen`). Additionally, a graphical representation of these times are displayed in Figure 3.1. Note that when the eigen-decomposition is given,

Vector size	10	50	100	200	300	500	800
`dmnorm`	0.28	10.99	42.81	200.41	574.47	1,528.75	5,175.41
`dmnorm2`	0.27	7.39	20.87	75.39	208.83	633.25	2,130.64
`eigen`	0.10	0.63	3.04	23.27	89.42	291.87	1,297.24

Table 3.1: Comparison among execution times (in seconds) of algorithms to sample from the multivariate normal distribution: `JAGS` (`dmnorm`), our own proposed implementation (`dmnorm2`) and the implementation using the given eigen-decomposition (`eigen`). Seven different vector sizes are considered.

sampling from the multivariate normal distribution is much faster. We also remark that the `dmnorm2` version is faster to perform the linear algebra computations needed in this sampling algorithm (it runs faster than `dmnorm`). In this example, the MCMC algorithm was set to perform 1,000 iterations. This performance test was carried out in a Ubuntu 16.04 machine with Intel Core i7-3770 x8 processor and 8GB memory.

Despite the efficiency of our proposed implementation, when trying to run a full Bayesian model including the likelihood and prior distributions (see an Example in 5.2), `JAGS` interrupts the execution process and produces an error message indicating that it is "unable to find appropriate sampler". `JAGS` is not able to sample from the posterior distribution, when using our proposed implementation. Note that we have defined a sampler for this distribution (again, see Appendix B).

Exploring this issue, we decided to implement our own version of the multivariate normal distribution (with mean vector and covariance matrix as arguments), which is basically a copy of the source code used in `JAGS` for defining this distribution. The only difference being in the approach to perform the eigen-decomposition. `JAGS` uses the `LAPACK` library (see Anderson et al., 1999), a library with `Fortran 77` subroutines for solving numerical linear algebra problems. However, in our implementation, we adopted the `Armadillo` library (see Sanderson and Curtin, 2016), which according to its authors is a high quality linear algebra library for the `C++` language, aiming towards a good balance between speed and ease of use. Surprisingly the same error message appeared

Box 3.5: JAGS script to sample from our proposed implementation.

```
model {
  theta[1:n] ~ dmnormcar(eigenValues, eigenVectors, tau2)
}
```

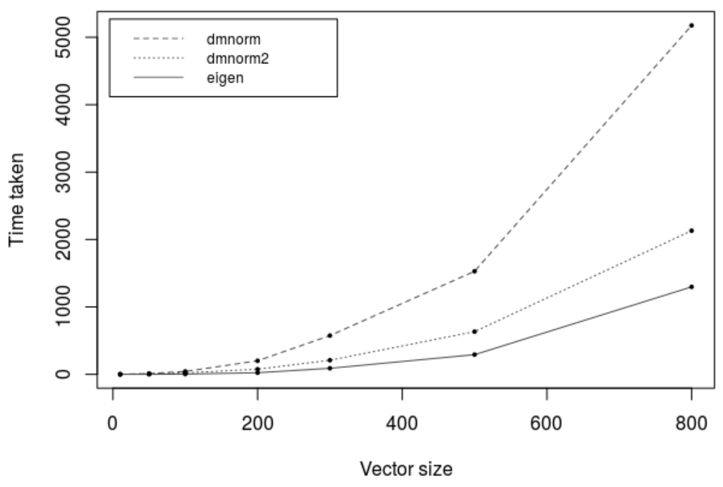


Figure 3.1: Times (in seconds) to sample from a multivariate normal distribution in `JAGS` using three approaches. The dashed line is the conventional sampler (`dmnorm`), the dotted line is our version of the conventional sampler (`dmnorm2`). The solid line is our proposed version using the eigen-decompotisision (`eigen`).

when trying to run a full Bayesian model with our implementation of this distribution. We were expecting `JAGS` to be able to sample from this distribution, as its procedures are the same as in the native multivariate normal distribution in `JAGS`.

With these results, we came to the conclusion that further work is necessary in order to define a new multivariate distribution in `JAGS` beyond developing a new `C++` class (which is the case for univariate distributions in `JAGS`). However, there is no complete manual for `JAGS` developers and no source of information about it, only the `JAGS` source code and the tutorial for implementing univariate functions and distributions in Wabersich and Vandekerckhove (2014) are available. Our main investigations and findings in this regard so far are described in Appendix B. As a future work, we plan to solve this computational issue.

Currently, our module offers a function (namely `precMatrixCAR`) to simply using the CAR model within a `JAGS` script. This function computes the precision matrix of the CAR distribution and this matrix is expected to be used as the precision matrix argument of the `dmnorm` function in `JAGS`. Although we demonstrated that this function can be inefficient, when the number of regions is small, the time taken to execute the model is reasonable. Regardless the number of regions, the posterior chains of the parameters present low levels of autocorrelation due to block sampling.

Chapter 4

Validating the covariance functions

In this chapter, we show a comparative study to validate the proposed implementation of the covariance structures in the `GeoJAGS` module, confirming that it works properly. By validation, we mean that we run a Bayesian model and compare the results with the corresponding frequentist estimates, considering a data set already explored in the literature. We start by presenting in Section 4.1 a simple data set that will serve as the base for our analysis. There is no special interest or main motivation in the application related to this data set. Some other interesting applications of spatial data models involving real data are discussed ahead in Chapter 6. In Section 4.2, we show the estimation using the frequentist approach. In Section 4.3, we confront the Bayesian estimation, using the proposed `GeoJAGS` module, with the corresponding script without the module. This is intended to show how simple and clean is the code when using the proposed module. In Section 4.4, we compare the maximum likelihood estimates with the Bayesian estimates. Finally, in Section 4.5, we demonstrate how to perform predictions at arbitrary locations with the proposed module. We also compare the results with those from the `GeoBUGS` module. The tests were developed using the freely available `R` software (R Core Team, 2017) and its packages `coda` (Plummer et al., 2006), `geoR` (Ribeiro Jr and Diggle, 2016) and `R2jags` (Su and Yajima, 2015).

4.1 Surface elevations data

The point-referenced data for this example are taken from Diggle and Ribeiro (2007). We have chosen this database since it is well known in the literature and thus allows a full comparison with published results to evaluate the performance of the `GeoJAGS` module. The data, stored into the `R` package `geoR`, contains the measurements (y_i) of the surface elevation for 52 locations ($\boldsymbol{s_i}$) within a 310-foot square. One unit of height represents 10 feet of elevation, whereas the unit of distance is 50 feet. Each $\boldsymbol{s_i}$ is a 2×1 vector containing the coordinates of the observation i. Figure 4.1 shows how the observations are distributed in the study area.

Let $S(\boldsymbol{s_i})$ denote the true elevation value at location $\boldsymbol{s_i}$. In this example, each y_i is approximately $S(\boldsymbol{s_i})$, since surface elevation can be measured with some negligible error, i.e. $S(\boldsymbol{s}) = Y(\boldsymbol{s}) + \epsilon$, with $\epsilon \approx 0$. The following model is considered in our analysis:

$$\mathbf{Y} \sim N_n[\mu, \mathbf{\Sigma}], \tag{4.1}$$

where $\mathbf{Y} = (y_1, \ldots, y_n)^\top$, $\mu = (\mu_1, \ldots, \mu_n)^\top$ and $\mathbf{\Sigma}$ is the covariance matrix that will be handled through our module. In addition,

$$\mu_i = \mu(s_i) = \beta_0 + \beta_1 s_{i1} + \beta_2 s_{i2}, \tag{4.2}$$

with s_{i1} and s_{i2} being the latitude and longitude coordinates, respectively.

Again, the parameter estimation that will be presented ahead in the next sections are based on the model proposed here for the elevation data.

4.2 Parameter estimation: the frequentist approach

The inference here is based on the maximum likelihood method to estimate β_0, β_1, β_2, and any additional parameter needed to properly define the covariance structure of the data. When using the `likfit` function from the `geoR` package, with parameter `trend="1st"` to consider the individual mean defined in (4.2), and assuming the Matérn

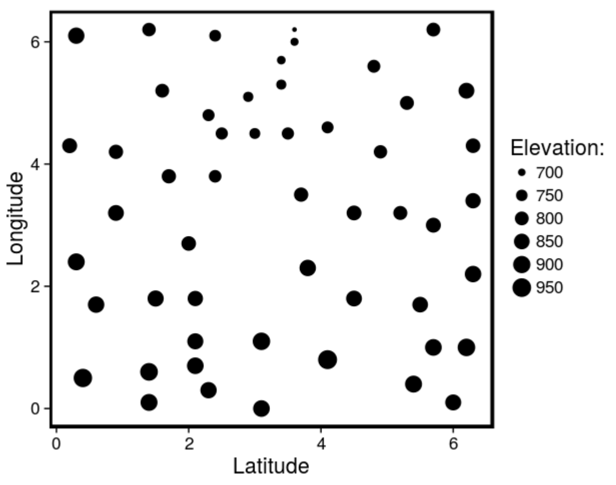


Figure 4.1: Circle plot of the surface elevation data. Circles are plotted with centers at the location of the observation and radii resembling the magnitude of the elevations (as multiples of 10 feet).

covariance function with $\kappa = 1.5$ fixed, we get the following estimates for the elevation data application:

$$\hat{\beta}_0 = 912.4865, \qquad \hat{\beta}_1 = -4.9904, \qquad \hat{\beta}_2 = -16.4640,$$

$$\hat{\tau^2} = 34.8953, \qquad \hat{\sigma^2} = 1{,}693.1329, \qquad \hat{\phi} = 0.8061.$$

As explained in Diggle and Ribeiro (2007), since `likfit` uses a numerical maximization procedure, initial values for the covariance parameters can be specified through the argument `ini`. In this example, `ini` contains initial values for σ^2 and ϕ. It is important to note that the Matérn covariance function implemented in `geoR` is not parameterized as described in Chapter 2. Instead, the package uses

$$C(t) = \begin{cases} \frac{\sigma^2}{2^{\kappa-1}\Gamma(\kappa)}\left(\frac{t}{\phi^*}\right)^{\kappa} K_{\kappa}\left(\frac{t}{\phi^*}\right) & \text{if } t > 0, \\ \tau^2 + \sigma^2 & \text{otherwise,} \end{cases}$$

where the parameter κ is equivalent to our ν and $\phi^* = 2\sqrt{\nu}\phi$.

Box 4.1 shows the `R` code considered to fit the previously described model. Similarly, we can also fit the model for other covariance functions available in the package (such as Cauchy, circular, cubic, exponential, spherical, Gneiting and wave) by simply replacing the argument `cov.model` in line 2 with the appropriate name of the covariance function.

Box 4.1: Parameter estimation using `likfit` function

```
1 > require(geoR)
2 > mod.matern = likfit(elevation, ini=c(3000, 2), trend = "1st", cov.model = "matern", kappa = 1.5)
```

4.3 Parameter estimation: the Bayesian approach

In this section, the focus is on the estimation of the hierarchical model through the Bayesian inference for the elevation data. Again, the software `JAGS` simplifies the Gibbs sampler implementation, since it requires only the specification of the prior distributions and the likelihood function. There is no need to specify the posterior full conditionals. The main obstacle in using `JAGS` for a spatial point-referenced data analysis is the construction of the covariance matrix $\mathbf{\Sigma}$.

As mentioned in Section 3.1.1, `GeoBUGS` offers to `OpenBUGS` and `WinBUGS` users two covariance structures: the powered exponential and the disc. Working with any covariance function other than these two demands that the user writes the code for this structure within the model script. In fact, this can be seen as a difficult task and also inefficient in terms of execution time. To the best of the authors' knowledge, currently neither `WinBUGS`, `OpenBUGS` or `JAGS` have implemented the Matérn covariance function and, furthermore they do not have the tools required to define this covariance structure within their model scripts (a function for the modified Bessel function is also unavailable). In this situation, we are unable to compare the estimated parameters from the `likfit` function with those from the Bayesian model fit using the Matérn covariance structure with `JAGS` or `OpenBUGS`.

4.3.1 The code with and without the module

We will now present how to use the new module and discuss the differences with respect to the script without the module. Before we start, we assume the reader have followed the steps to correctly install the `GeoJAGS` module in his or her computer; see steps given

Box 4.2: JAGS script using the exponential covariance function without the module.

```
1  model
2  {
3    Y ~ dmnorm(mu, omega.w)
4    for (i in 1:N) {
5      mu[i] <- beta[1] + beta[2]*coords[i,1] + beta[3]*coords[i,2]
6    }
7    for (i in 1:N){
8      sigma[i,i] <- tau2 + sig2
9      for (j in (i+1):N){
10       sigma[i,j] <- sig2 * exp(-phi*dist[i,j]);
11       sigma[j,i] <- sigma[i,j]
12     }
13   }
14   beta ~ dmnorm(mu.beta, omega.beta)
15   omega.w <- inverse(sigma)
16   sig2 ~ dgamma(a_sig, b_sig)
17   tau2 ~ dgamma(a_tau, b_tau)
18   phi ~ dgamma(a_phi, b_phi)
19 }
```

in Section 3.3 and in the tutorial by Wabersich and Vandekerckhove (2014, p. 20-21). In addition, the software R and the package `R2jags` are expected to be installed and ready to use.

Without our module, the `JAGS` user is restricted to work with simple covariance structures (such as the spherical, rational quadratic or the exponential). See an example considering the exponential covariance function in Box 4.2. In this case, we set gamma prior distributions for σ^2, τ^2 and ϕ, and a multivariate normal distribution for the regression coefficients in $\boldsymbol{\beta}$. The covariance matrix is stored in `omega.w` (Box 4.2, line 15). Formally, the prior specifications are:

$$\sigma^2 \sim Ga(a_\sigma, b_\sigma), \qquad \tau^2 \sim Ga(a_\tau, b_\tau), \qquad \phi \sim Ga(a_\phi, b_\phi),$$

$$\boldsymbol{\beta} \sim N_3(\boldsymbol{M_\beta},\ \boldsymbol{V_\beta}),$$

where the hyperparameters a's and b's are specified to determine vague priors. We chose $a_\sigma = a_\tau = a_\phi = 0.1$, $b_\sigma = b_\tau = b_\phi = 0.001$, which determine a gamma distribution with mean 100 and variance 10^5. These values are carefully chosen to appropriately represent the scale of the observation in the elevation data. The remaining hyperparameters values are $\boldsymbol{M_\beta} = (0,0,0)^\top$ and $\boldsymbol{V_\beta}^{-1} = \text{diag}_{3\times 3}(10^{-6})$, which is a precision matrix; these choices for the coefficients suggest no information regarding the sign of these parameters and a high prior uncertainty about them.

The Gibbs sampler is set to perform 15,000 iterations (burn-in period with 5,000

Box 4.3: Running the JAGS model in R.

```
library(R2jags); load.module(``GeoJAGS");
library(geoR); data(elevation);
cvexp = jags(data = list(`coords' = coord, `N' = 52, `Y' = elevation$data,
                         `mu.beta' = rep(0,3), `omega.beta' = diag(x=1/1000000, ncol=3, nrow=3),
                         `a_sig' = 0.1, `b_sig' = 0.001,
                         `a_tau' = 0.1, `b_tau' = 0.001,
                         `a_phi' = 0.1, `b_phi' = 0.001),
             model.file=`model.exp', n.chains = 1, n.burnin = 1000, n.iter = 2000,
             parameters.to.save = c(`beta', `tau2', `sig2', `phi'))
```

iterations and 1,000 observations forming the final posterior sample). Due to the autocorrelation of the chains, we choose lag = 10 for selecting spaced observations for the posterior sample. Moreover, the analysis is based on a single chain. Box 4.3 shows how to run the `JAGS` model inside the `R` environment.

It is important to emphasize that implementing the posterior full conditionals can be an obstacle, and thus it is not attractive for those not familiar with the Bayesian inference or not well trained in programming. In some situations, only the kernel of the posterior full conditionals is know; in this case extra sampling algorithms such as Metropolis-Hastings (see Metropolis et al. 1953; Hastings 1970), adaptive rejection sampling (see Gilks and Wild 1992; Gilks 1992) or slice sampling (see Neal 2003), among others, are needed within the Gibbs sampler. With the support of the sampling algorithms implemented in `JAGS`, our module can significantly reduce the implementation burden of the problem.

The usage of the `GeoJAGS` module is quite straightforward. Besides the fact that it is simpler, many parts of the model script in Box 4.4 look similar to the one defined in Box 4.2. Note that the commands in lines 7-13 of Box 4.2 are replaced by the one in line 9 of Box 4.4, where we set the exponential covariance structure. Here we have

Box 4.4: JAGS script using the exponential covariance structure with the module.

```
 1 model
 2 {
 3   Y ~ dmnorm(mu, omega.w)
 4   for (i in 1:N) {
 5     mu[i] <- beta[1] + beta[2]*coords[i,1] + beta[3]*coords[i,2]
 6   }
 7   beta ~ dmnorm(mu.beta, omega.beta)
 8   omega.w <- inverse(sigma)
 9   sigma <- CVPowerExp(coords, sig2, phi, tau2, 1)
10   sig2 ~ dgamma(a_sig, b_sig)
11   tau2 ~ dgamma(a_tau, b_tau)
12   phi ~ dgamma(a_phi, b_phi)
13 }
```

used the fact that the exponential covariance function is a particular case of the powered exponential function, when $p = 1$. However, GeoJAGS also offers the CVExp function, in which the user is not required to set a value for the parameter p, since it is fixed (see details in Appendix A).

Additionally, note that in Box 4.2 line 10 the user is required to provide the dist argument, which is a matrix of interlocation distances. In fact, this is not a trivial task depending on the users programming skills. When using the covariance functions from the proposed module, the user needs to provide only the list of the locations coordinates. We compared the performance of the model scripts described in Boxes 4.2 and 4.4, considering a MCMC setting with burn-in period of 1,000, lag = 5 and posterior sample size of 1,000 iterations. In the model in Box 4.2, the time taken to perform the Gibbs sampler algorithm was 116 seconds (disregarding the time taken to build the interlocation distances matrix), while, when using the GeoJAGS module to build the covariance matrix (Box 4.4), it took 114 seconds to produce the posterior chain. The machine considered in this performance test was a Ubuntu 17.04, Intel Core i7-4770 x8 processor with 16GB memory. Note that although the time difference is small, the facility the proposed module provides to its user is considerable.

4.3.2 Model fit with Matérn covariance

Now that we have built the GeoJAGS module, we can fit a Bayesian model with the Matérn covariance structure for the elevation data using the newly created CVMatern function. We also implemented the bessel_k function, that computes the modified Bessel function of the third kind, for those users willing to implement their own Matérn covariance function (see details in Appendix A).

Let us recall the model in (4.1). Similarly to the script in Box 4.4, we can set the covariance structure to any of the twelve functions currently available in the module, replacing the command in line 9 by the respective name of the covariance function. Currently, the options are: CVExp, CVGaussian, CVPowerExp, CVMatern, CVCauchy, CVGenCauchy, CVGneiting, CVRatQuad, CVCircular, CVCubic, and CVSpherical. As it can be seen, the chosen name for the function clearly indicates the type of the covariance structure being used. We can run this `JAGS` script in `R` using the structure in Box 4.3.

When using the spherical, the circular or the cubic covariance functions, we emphasize that the definitions presented in Section 2.2.1 show that the intervals where these functions are defined depend upon the parameter ϕ. The reciprocal of this parameter represents the distance at which the spatial dependence vanishes (in the cases of circular and cubic covariance structures). For the spherical covariance function, ϕ itself is seen as the distance at which the spatial dependence is negligible. Therefore, the covariance functions were implemented assuming ϕ as fixed and specified by the user, who is in charge to determine an appropriate or reasonable value for this dependence threshold. If we set a prior to ϕ, computational problems arise when attempting to compute the inverse of the covariance matrix. We suggest that when using these covariance structures, the user should plot the empirical variogram and choose a suitable value for ϕ in the application. Thus, line 12 in Box 4.4 can be removed and a value for ϕ must be passed as data to `JAGS`.

4.4 Comparing frequentist and Bayesian estimates

In this section, we develop a comparative study to verify whether the results from the Bayesian model fit using `JAGS` are in accordance with those from the frequentist analysis reported in the literature. In Box 4.1, we demonstrated how to adjust a model with the linear trend defined in Equation (4.1) using maximum likelihood method, considering the Matérn covariance function. We run the code for all the covariance functions available in the `R` package `geoR`. Some of these functions produced errors when trying to adjust the model for the elevation data. The wave covariance is one of them; here the `likfit` function reported an error message that seems to be related to matrix inversion. This error is observed even when running the `likfit` function multiple times with distinct initial values for ϕ and σ^2, as recommended by the developers. Although the documentation of the `geoR` states that the generalized Cauchy covariance function is available in the package, when attempting to use it we observe an error notification displayed in the screen informing that it is in fact not yet implemented. Further, `geoR` does not imple-

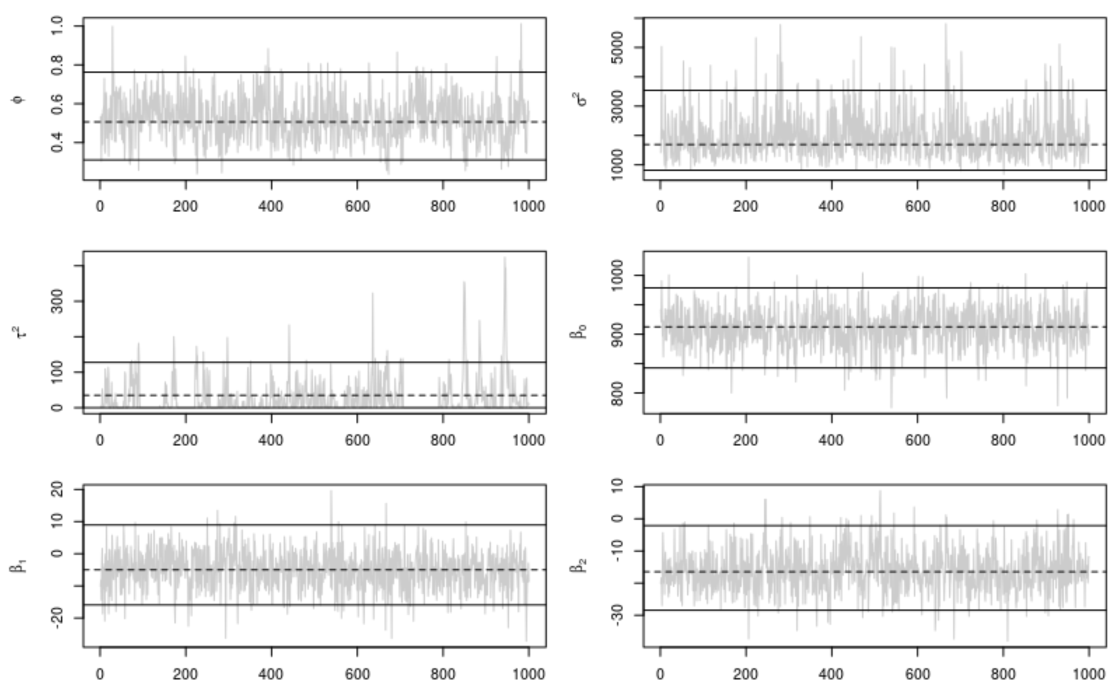


Figure 4.2: Posterior sample, 95% HPD limits (solid lines), and the maximum likelihood estimates (dashed horizontal line) of the coefficients and the parameters using the Matérn covariace function for the elevation data.

ment the rational quadratic covariance structure, therefore we could not compare with the estimates from the `GeoJAGS` module for this spatial model.

The results discussed in Figure 4.2 and Table 4.1 are related to the Matérn covariance function. The main goal of the analysis here is not to perform model selection, instead we aim to compare the estimates of the maximum likelihood method in `likfit` with those from `JAGS` (using our module). Figure 4.2 shows the chains for six parameters estimated with the Matérn model for the elevation data. In each graph, the dashed horizontal line indicates the maximum likelihood estimates. The two horizontal solid lines represent the HPD intervals limits for each parameter (with 95% credibility). In addition, the posterior means, HPD intervals, and maximum likelihood estimates are presented in Table 4.1. We considered the `coda` package to build the HPD intervals. This is an advantage of using the Bayesian approach with `JAGS` instead of the frequentist. In other words, if we want to compute confidence intervals for the maximum likelihood estimates, we must compute their limits by ourselves, since there is no function that can automatically perform these computations in `geoR`. Moreover, the `JAGS` script for this model considering the Matérn

Parameters	Maximum likelihood estimates	JAGS model using `CVMatern` function	HPD interval (95%)	
ϕ	0.506	0.512	0.279	0.741
τ^2	34.895	37.602	0.000	146.124
σ^2	1,693.133	1,971.465	844.610	3,552.477
β_0	912.487	912.050	846.895	984.655
β_1	-4.990	-4.868	-17.541	8.647
β_2	-16.464	-16.500	-29.634	-3.919

Table 4.1: Comparison of parameter estimates for the two approaches (Bayesian vs. Frequentist) using the Matérn covariance structure.

covariance function took 655 seconds to run a sample of 15,000 iterations (specifications of the machine used: Ubuntu 17.04, Intel Core i7-4770 x8 processor and 16GB memory). The results from other covariance functions are presented in Appendix C. They show similar conclusions as those for the Matérn.

Note that all the maximum likelihood estimates lie within the respective HPD interval, in other words the chains are oscillating in a region of the parameter space where the maximum likelihood estimates are located. As it can be seen, all chains indicate the visual behavior of a convergence condition and they exhibit low autocorrelation, which is expected since we are fitting a simple Gaussian model.

This result (and those in Appendix C) clearly confirms that the new module is working as expected. The remaining comparison between the frequentist and the `JAGS` model estimates and further comments related to other covariance structures are presented in Appendix C. Additionally, Figure 4.3 shows the estimates for β_0 using the maximum likelihood and the Bayesian methods. Note that the frequentist estimates for the models considering the rational quadratic, the wave and the generalized Cauchy covariance functions are not shown due to the fact that they are not available in `geoR`. Further comments can be found in Appendix C. After several tests being carried out, we can finally conclude that the covariance functions available in our `GeoJAGS` module are correctly implemented and working appropriately.

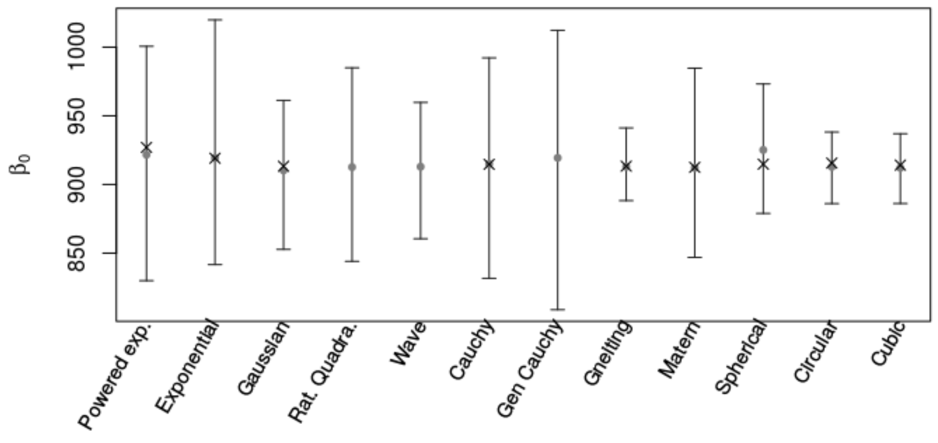


Figure 4.3: Estimates for β_0: the x dots are the maximum likelihood estimates, the solid circles are the posterior mean, and the vertical lines represent the HPD for each of the twelve covariance functions in the `GeoJAGS` module.

4.5 Spatial prediction at arbitrary locations

Spatial prediction can be performed by using the `GeoJAGS` module in a different manner that is done in `GeoBUGS`. In this section, we compare the usage of these two modules for predicting response values at arbitrary locations and also analyze the results given by them for the surface elevations data. We define a 15×15 regular grid of points at which we wish to predict surface elevation within the study area (see Figure 4.4). The hierarchical model previously presented in Section 4.3 will be adapted here to accommodate prediction in such a way that is done in `OpenBUGS`, thus we will be able to compare the predicted values from the two spatial modules. The model presented here is based on the one available in the `GeoBUGS` user manual for spatial prediction on the surface elevations data. A few adjustments were made in the prior distributions in order to make it possible to run the model with the very same specifications in `JAGS` and `OpenBUGS`.

Instead of considering the response variable (elevation measurements) following a multivariate normal distribution, we set

$$\begin{aligned} Y_i \mid \boldsymbol{W}, \nu, \beta &\sim N(\mu_i, \nu), \\ \boldsymbol{W} \mid \sigma^2, \tau^2, \phi, \kappa &\sim N\big(\mathbf{0}, \boldsymbol{\Sigma}(\sigma^2, \tau^2, \phi, \kappa)\big), \end{aligned} \tag{4.3}$$

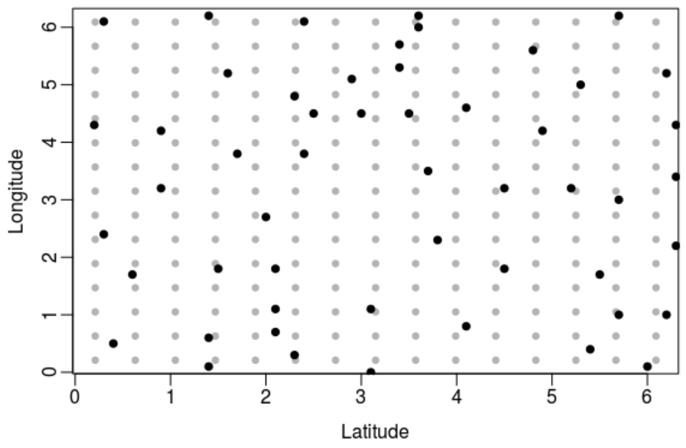


Figure 4.4: Spatial arrangement of the elevation data. Solid black circles are locations of the 52 observed measurements, solid gray circles are the 225 locations in which the response variable will be predicted.

where $\mu_i = \beta_0 + \beta_1 s_{i1} + \beta_2 s_{i2} + w_i$, s_{i1} and s_{i2} are the latitude and the longitude ($i = 1, \ldots, n$), $\boldsymbol{W} = (w_1, \ldots, w_n)^\top$ is the vector of spatial random effects, $\boldsymbol{\Sigma}(\cdot)$ is the powered exponential covariance function. Consider σ^2, τ^2, ϕ, and κ being the parameters of the covariance structure and ν is the non-spatial variance. The prior distributions are

$$\beta_1 \sim N(0, 10^5), \qquad \beta_2 \sim N(0, 10^5), \qquad \beta_j \sim N(0, 10^5),$$

$$1/\nu \sim Ga(10^{-3}, 10^{-3}), \quad 1/\sigma^2 \sim Ga(10^{-3}, 10^{-3}), \quad \phi \sim U(0.05, 20), \quad \kappa \sim U(0.05, 1.95).$$

The values for the hyperparameters were chosen according to the example available in the `GeoBUGS` manual.

Box 4.5 shows the `JAGS` script for the model in (4.3). The lines 2-5 are related to the observed data (`Nobs` elements), lines 6-9 represent the values to be predicted (`Npred` items). Note that `coords` is a (`Nobs` + `Npred`) $\times$ 2 vector with latitude and longitude coordinates for both "observed" and "to be predicted" locations. This vector is used as an argument for the `CVPowerExp` function. Recall that in the definitions of the covariance functions in Section 2.2.1, we define σ^2 as the spatial variance effect and τ^2 as the non-spatial variance effect. In the model presented in (4.3), ν plays the role of the non-spatial variance. As a result, the argument τ^2 in the covariance function `CVPowerExp` must be set to zero to avoid identifiability issues. The argument κ is the exponent term in the

Box 4.5: JAGS script for prediction.

```
model{
  for (i in 1:Nobs) {
    mu[i] <- beta0 + beta1*coords[i,1] + beta2*coords[i,2] + w[i]
    Y[i] ~ dnorm(mu[i], inv.nu)
  }
  for (j in 1:Npred) {
    mu[Nobs+j] <- beta0 + beta1*coords[Nobs+j,1] + beta2*coords[Nobs+j,2] + w[Nobs+j]
    Ypred[j] ~ dnorm(mu[Nobs+j], nugget)
  }
  w ~ dmnorm(mu.w, Omega.w)
  Sigma.w <- CVPowerExp(coords, sig2, phi, 0, kappa)
  Omega.w <- inverse(Sigma.w)
  inv.sig2 ~ dgamma(a_inv.sig2, b_inv.sig2)
  sig2 <- 1/inv.sig2
  inv.nu ~ dgamma(a_nu, b_nu)
  phi ~ dunif(a_phi, b_phi)
  kappa ~ dunif(a_kappa, b_kappa)
  beta0 ~ dnorm(mu.beta0, tau.beta0)
  beta1 ~ dnorm(mu.beta1, tau.beta1)
  beta2 ~ dnorm(mu.beta2, tau.beta2)
}
```

Box 4.6: Running the JAGS model in R.

```
library(R2jags); load.module(‘‘GeoJAGS”);
outjags = jags(data = list(‘Nobs’ = Nobs, ‘Npred’= Npred, ‘mu.w’ = rep(0, Nobs+Npred),
                             ‘coords’ = coords, ‘Y’ = elevation$data,
                             ‘a_nu’ = 0.001, ‘b_nu’ = 0.001,
                             ‘a_inv.sig2’ = 0.001, ‘b_inv.sig2’ = 0.001,
                             ‘mu.beta0’ = 0, ‘tau.beta0’ = 1/100000,
                             ‘mu.beta1’ = 0, ‘tau.beta1’ = 1/100000,
                             ‘mu.beta2’ = 0, ‘tau.beta2’ = 1/100000,
                             ‘a_phi’ = 0.05, ‘b_phi’ = 20,
                             ‘a_kappa’ = 0.05, ‘b_kappa’ = 1.95),
                 inits = list(nugget=0.001, inv.sig2=0.001, phi=0.4, kappa=1),
                 parameters.to.save = c("kappa", "beta0", "beta1", "beta2", "inv.nu", "phi", "sig2",
                                        "w", paste0(‘pred[‘, 1:Npred, ‘]’)),
                 model.file=filepath, n.chains = 1, n.burnin = 10000, n.iter = 20000, n.thin = 10)
```

powered exponential covariance structure. Finally, vector `w` stores the spatial random effects of all locations (observed and to be predicted).

Values for the remaining variables in the model script must be passed as data before running `JAGS`, as seen in Box 4.6, with commands to run the `JAGS` model inside the R environment using the `R2jags` package. Again, values for the hyperparameters were chosen according to those presented in the `GeoBUGS` manual. The analysis here is based on a single chain and the Gibbs sampler is set to perform 20,000 iterations (burn-in period with 10,000 iterations and 1,000 observations forming the final posterior sample). Due to autocorrelation of the chains, we choose lag = 10.

The `OpenBUGS` script for a version of the model in (4.3) is presented in Box 4.7. Here, the powered exponential covariance structure is introduced by the `spatial.exp` function and the prediction is performed through `spatial.pred` function, which performs joint prediction from the joint distribution and not from the full conditionals. Latitude and

Box 4.7: OpenBUGS model script for prediction.

```
model {
  for(i in 1:Nobs) {
    mu[i] <- beta0 + beta1 * coordX[i] + beta2 * coordY[i]
  }
  for(j in 1:Npred){
    mu.pred[j] <- beta0 + beta1 * coordX.pred[j] + beta2 * coordY.pred[j]
  }
  Y[1:Nobs] ~ spatial.exp(mu[], coordX[], coordY[], inv.sig2, phi, kappa)
  Y.pred[1:Npred] ~ spatial.pred(mu.pred[], coordX.pred[1:Npred], coordY.pred[1:Npred], Y[])
  inv.sig2 ~ dgamma(a_inv.sig2, b_inv.sig2)
  sigma2 <- 1/inv.sig2
  phi ~ dunif(a_phi, b_phi)
  kappa ~ dunif(a_kappa, b_kappa)
  beta0 ~ dnorm(mu.beta0, tau.beta0)
  beta1 ~ dnorm(mu.beta1, tau.beta1)
  beta2 ~ dnorm(mu.beta2, tau.beta2)
}
```

longitude coordinates for observed values are now detached from those for prediction lattice points in vectors `coordX` and `coordY`, and `coordX.pred` and `coordY.pred`, respectively. The `R` script for running this `OpenBUGS` model is quite analogous to the one in Box 4.6. Hyperparameters values are not shown in a separate box given that they are the same as those used in the `JAGS` script, thus similar to Box 4.6.

We emphasize the main difference between the two modules. The nugget effect (non-spatial variance) τ^2 cannot be included when using the `GeoBUGS` module, because this parameter is not considered in the implementation, thus all the variance effects are incorporated in one parameter. An advantage of using the proposed module is that we allow the user the right to introduce the nugget effect into the model.

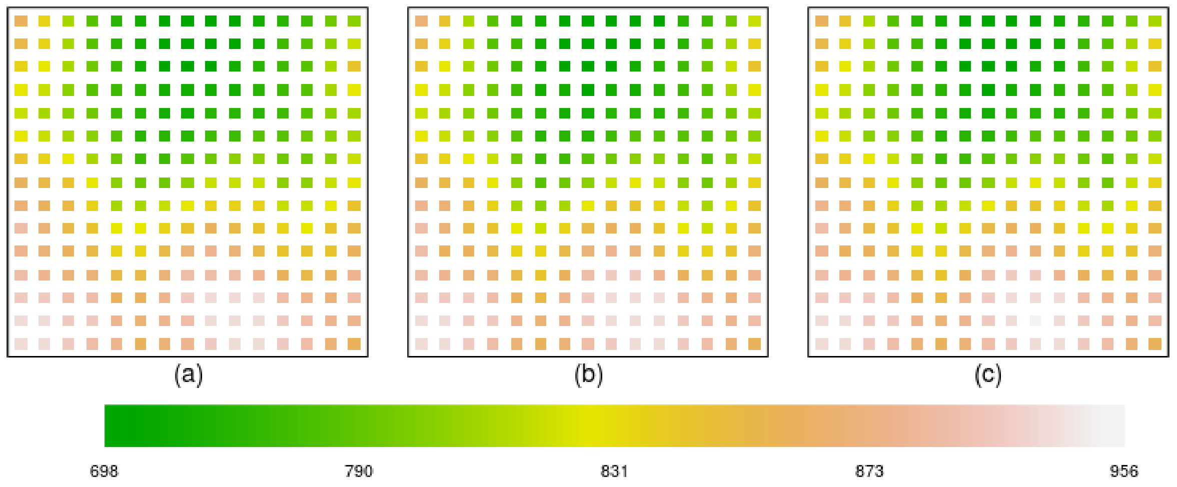


Figure 4.5: Posterior mean estimates of surface elevation prediction at 225 locations using (a) `likfit` function, (b) `GeoJAGS` module and (c) `GeoBUGS` module.

Running in a Ubuntu 17.04 machine with Intel Core i7-4770 x8 processor and 16GB memory, the `GeoJAGS` module took approximately 114 minutes to execute this script, while `GeoBUGS` took 144 minutes.

Figure 4.5 displays the posterior mean estimates for the surface elevation values predicted for the 225 locations, considering the frequentist approach and the Bayesian approach using `GeoJAGS` and `GeoBUGS` modules. Note that the three methods produced very similar estimated elevation for every location in predicting grid.

Figure 4.6 presents the interpolation over the study area using the `R` package `akima` (see Akima and Gebhardt, 2016). The `akima` package is useful for making interpolations involving irregular data over a regular study area (such as the squared region in the elevation data example). The results from the frequentist approach and those from the two spatial modules are quite similar.

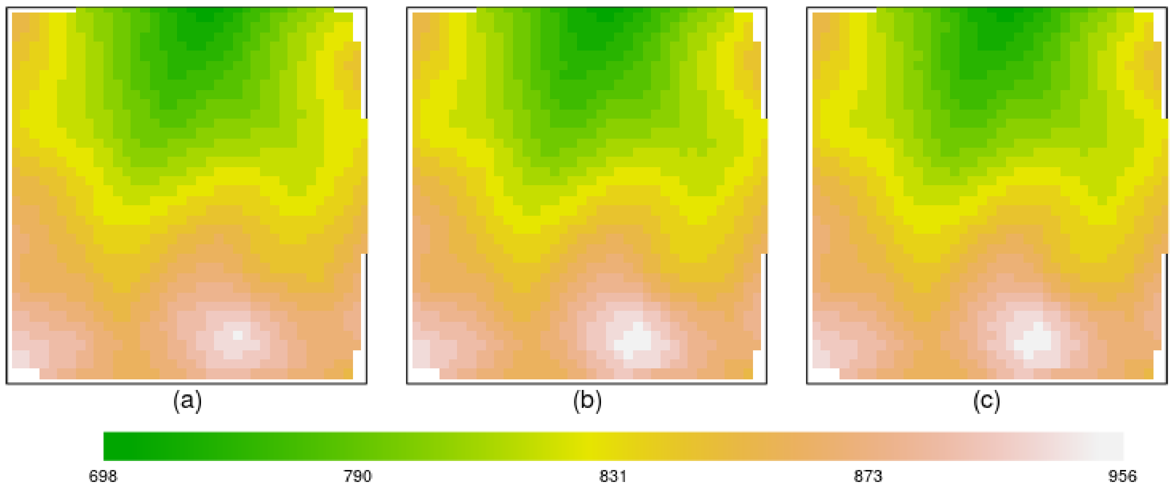


Figure 4.6: Interpolation of the 225 posterior mean estimates of surface elevation data over the study area using predictions from (a) `likfit` function, (b) `GeoJAGS` module and (c) `GeoBUGS` module.

Chapter 5

Validating the CAR distribution

In this chapter, we show a comparative study to validate the implementation of the CAR function in the `GeoJAGS` module. We begin by presenting in Section 5.1 a simple real data set for which our analysis is based. In Section 5.2, we demonstrate how to use the CAR function from our module, comparing with the implementation in `OpenBUGS`. We conclude this chapter in Section 5.3 by showing a simulation study in which we control the values for the variance parameters and also for the spatial random effects. This is done to compare the performances of `GeoJAGS` and `GeoBUGS` in terms of parameter estimation.

5.1 Gross domestic product data

The areal data for this example is the gross domestic product (GDP) per capita in the 853 municipalities of Minas Gerais state in 2013, obtained in IBGE (2018). Figure 5.1 presents the data using the green scale to classify the values into a few classes. The political boundaries in Minas Gerais state are available as polygons in the `R` package `mapsBR`, see Silva e Silva (2014). In this data set, each unit is equivalent to a thousand Brazilian reais (1 USD = 3.26 BRL in January 2018). The municipality with the minimum GDP per capita (4.22) is São João das Missões in the north of the state, whereas São Gonçalo do Rio Abaixo in the central area has the highest GDP per capita value (340.69). In addition, considering the 853 municipalities, the average GDP per capita is 15.31, with

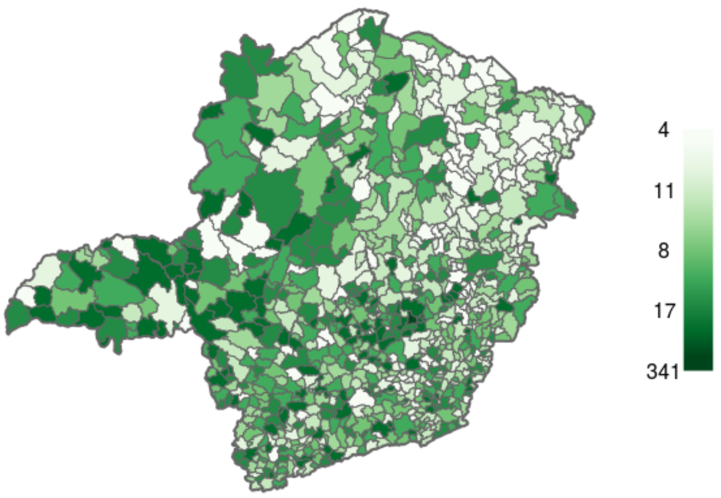


Figure 5.1: GDP per capita in the municipalities of Minas Gerais in 2013 (in thousands of Brazilian Real).

a variance of 341.34.

Although there is no evident pattern shown in Figure 5.1, an attentive eye would note the fact that when compared with the northeastern municipalities, a greater number of western regions of the map have higher GDP (darker shades of green). In this case, we might be interest in investigating spatial association between the observations. As the fundamental idea here is to validate the CAR matrix function in the `GeoJAGS` module, we will fit a rather simple hierarchical model and, afterwards, we compare the results from `GeoBUGS` with those from our module.

Let Y_i be the GDP per capita in the municipality i ($i = 1, \ldots, 853$) and let $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_{853})$ be the vector of spatial random effects associated with each region. Then, consider

$$\begin{aligned} Y_i \mid \theta_i &\sim N(\mu + \theta_i, \sigma^2), \\ \boldsymbol{\theta} &\sim CAR(\tau^2, \rho), \end{aligned} \tag{5.1}$$

where μ is a scalar representing the global GDP mean of all regions, σ^2 is the non-spatial variance effect, τ^2 is the spatial variance effect and ρ is the parameter introduced to guarantee that the precision matrix $\boldsymbol{\Sigma}^{-1}$ is nonsingular (here, we consider $\rho = 0.9$). The proximity matrix used in this model is binary, i.e. $w_{ij} = 1$ if i and j share a common

Box 5.1: OpenBUGS model script for the GDP data.

```
1  model {
2    for (i in 1:n) {
3      Y[i] ~ dnorm(mean[i], inv.sig2)
4      mean[i] <- mu + theta[i]
5    }
6    theta[1:n] ~ car.normal(adj[], weights[], num[], inv.tau2)
7    for (j in 1:sumNumNeigh){
8      weights[j] <- rho
9    }
10   inv.tau2 ~ dgamma(2.001, 1.001)
11   tau2 <- 1/inv.tau2
12
13   inv.sig2 ~ dgamma(2.001, 1.001)
14   sig2 <- 1/inv.sig2
15
16   mu ~ dnorm(0, prec.mu)
17   prec.mu <- 1/100
18 }
```

boundary.

5.2 Hierarchical model fitting

Since we are interested in a Bayesian modeling approach, we set the following prior specifications as a complement to the model in (5.1):

$$\mu \sim N(\mu_0, \nu_0^2), \qquad \sigma^2 \sim IG(a_\sigma, b_\sigma), \qquad \tau^2 \sim IG(a_\tau, b_\tau),$$

where the hyperparameters are chosen to determine vague distributions. We choose the global mean μ to be normally distributed with mean $\mu_0 = 0$ and variance $\nu_0^2 = 100$, values that suggest no information regarding the sign of this parameter and a high prior uncertainty about it. We choose $a_\sigma = a_\tau = 2.001$ and $b_\sigma = b_\tau = 1.001$, which give an inverse gamma distribution with mean 1 and variance 10^3.

The Gibbs sampler algorithm is set to perform 10,000 iterations with a burn-in period of 5,000 iterations and 5,000 observations forming the posterior sample. Since we are fitting a simple model, the autocorrelation is not expected to be severe, thus we choose lag=1. Moreover, the analysis here will be based on a single chain.

Box 5.1 shows the `OpenBUGS` script for the model in (5.1). As mentioned before, `GeoBUGS` implements the CAR distribution through the function `car.normal`, which uses a particular setting for its parameters (see line 6 of Box 5.1). The `adj` parameter is a sparse representation of the full adjacency matrix, `weights` is a vector giving unormalised

Box 5.2: JAGS model script for the GDP data.

```
1  model {
2    for (i in 1:n) {
3      Y[i] ~ dnorm(mean[i], inv.sig2)
4      mean[i] <- mu + theta[i]
5    }
6    theta[1:n] ~ dmnorm(zero, precMatrixCAR(W, rho, inv.tau2))
7
8    inv.tau2 ~ dgamma(2.001, 1.001)
9    tau2 <- 1/inv.tau2
10
11   inv.sig2 ~ dgamma(2.001, 1.001)
12   sig2 <- 1/inv.sig2
13
14   mu ~ dnorm(0, prec.mu)
15   prec.mu <- 1/100
16 }
```

weights associated with each pair of areas (this parameter plays the role of the scalar ρ defined in Section 2.3.3), `num` is a vector giving the number of neighbors for each area and `inv.tau2` is a scalar argument representing the spatial precision (inverse of spatial variance). The first three arguments of this function must be entered as data. The `sumNumNeigh` scalar value is the length of `adj`. Details about this implementation of the CAR distribution can be found in the `GeoBUGS` user's manual at `www.mrc-bsu.cam.ac.uk/wp-content/uploads/geobugs12manual.pdf`.

Box 5.2 presents the `JAGS` script for model (5.1) using the CAR matrix function (`precMatrixCAR`) from the `GeoJAGS` module, as described in Section 3.4.2. Note that, differently from the implementation in `GeoBUGS`, the parameter ρ in the `JAGS` script is a scalar value, i.e. every neighboring relationship in the study region weights the same (again, we consider $\rho = 0.9$). Line 6 shows how the CAR distribution for the spatial random effects can be used with the proposed module. As the name suggests, the `zero` parameter in the multivariate normal distribution is a null vector. The proximity matrix is stored in `W` and `inv.tau2` represents the precision. Although it is necessary to use the `precMatrixCAR` with the `dmnorm` function, the implementation is simpler than the one from `GeoBUGS`.

Again, by using the `R` packages `rbugs` and `R2jags`, we can easily run the model scripts in Boxes 5.1 and 5.2, respectively. The posterior means of μ, σ^2, τ^2 and for one of the spatial random effects are presented in Table 5.1. As there are a large number (853) of spatial random effects, we decided to show only results corresponding to the capital city of Minas Gerais, Belo Horizonte, namely θ_{BH}. The global mean GDP per capita estimated

Parameters	GeoJAGS			GeoBUGS		
	Mean (SD)	95% HPD interval		Mean (SD)	95% HDP interval	
μ	15.26 (0.62)	14.07	16.48	15.26 (0.59)	14.02	16.38
σ^2	340.16 (16.64)	308.08	373.33	303.30 (18.16)	268.90	337.90
τ^2	0.55 (0.26)	0.18	1.04	58.02 (28.13)	13.42	109.20
θ_{BH}	0.02 (0.31)	-0.58	0.66	7.10 (3.89)	-0.52	14.74

Table 5.1: Comparison of estimates for the two spatial modules. Only one spatial random effect is shown, representing Belo Horizonte. Standard deviations are shown in parentheses.

by the two modules are close to the data average 15.31. For the non-spatial variance σ^2, the estimate from `GeoJAGS` (340.16) is close to the sample variance, whereas the same does not occur for the corresponding estimate from `GeoBUGS` (303.30). There is a substantial difference between the estimated spatial variance τ^2 from `JAGS` and `OpenBUGS`. The same behavior is observed with respect to the spatial random effects θ_i (see also Figure 5.4). These aspects of difference will be regarded in the remaining of this subsection.

Figure 5.2 shows the chains for the very same parameters presented in Table 5.1. In each graph, the solid horizontal line represents the posterior mean estimate, the two horizontal dashed lines indicate the limits of the 95% HPD intervals. For comparison purposes, the vertical axis limits are fixed for each parameter in the graph, i.e. each row in Figure 5.2 has the same range in the y-axis. Note that the autocorrelation in the chains is not negligible, specially those from `GeoBUGS`. A detailed version of this figure for parameters τ^2 and θ_{BH} can be found in Appendix D (page 112).

Additionally, using the `effectiveSize` function from the `coda` package, we obtain the effective sample size (ESS) for each chain in the model. Figure 5.3 shows the ESS of the chains obtained from `GeoJAGS` (right column) and `GeoBUGS` (left column). We can see that in terms of autocorrelation the performance of the proposed module is better than the `GeoBUGS` module for almost all parameters, being comparable only for the spatial variance effect τ^2 (again, see also Figure D.1, page 112) and the global mean μ.

Figure 5.4 displays the posterior mean of the spatial random effects described in (5.1). Note that the range of the spatial random effects are considerably different for each spatial module. An explanation for this divergence lies in the fact that the non-spatial variance is greater than the spatial variance, which makes it difficult to estimate the

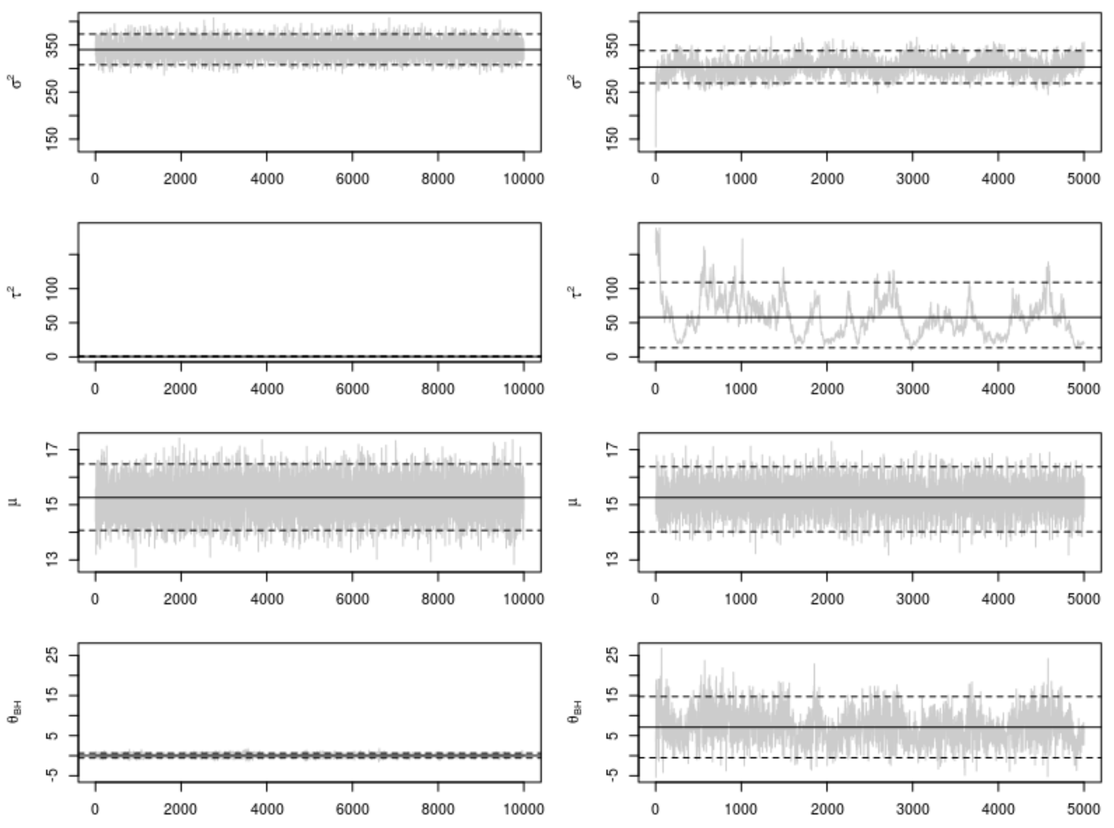


Figure 5.2: Posterior sample, 95% HPD limits (dashed lines) and the posterior mean (solid horizontal lines) of 4 parameters of the GDP per capita in Minas Gerais data set. Estimates in the left column are from `GeoJAGS` and from the right column are from `GeoBUGS`.

spatial random effects. Another reason for this difference could be due to the fact that the `GeoJAGS` module jointly samples all the spatial random effects in a single block (using the multivariate normal distribution) whilst, based on some evidence, `GeoBUGS` samples from the univariate full conditional distributions (we are not sure since it is not an open-source software and neither is mentioned in its user manual). In fact, there will be differences in the autocorrelation whether the parameters are sampled in block or individually and it affects the estimates in terms of efficiency (see Mayrink and Gamerman, 2009). Since we are dealing with a real data set and we do not know the true values of the latent random effects, we cannot state whether `GeoJAGS` or `GeoBUGS` are correctly estimating these effects. Aiming to investigate these different behaviors in the estimates from the two modules, we present a simulation study in Section 5.3.

Despite the difference in the scale of the estimated random effects θ_i from `JAGS` and

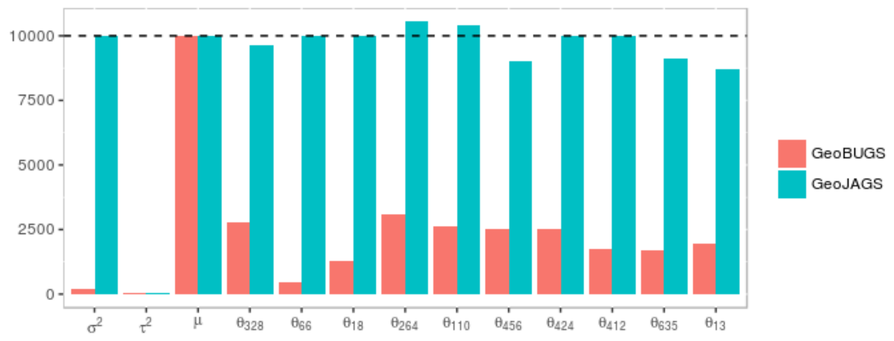
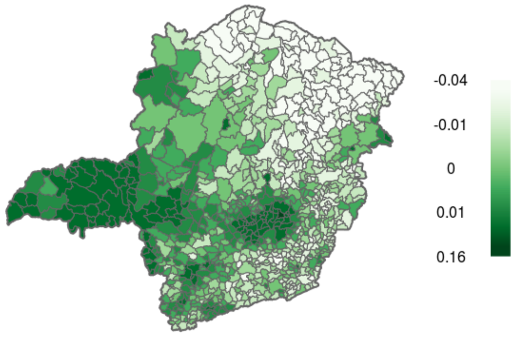


Figure 5.3: GDP data analysis. Effective sample size of posterior chains for the variance parameters (σ^2 and τ^2) and for 10 randomly selected spatial random effects (θ_i). The left bars display the ESS from `GeoBUGS` and the right bars show the ESS from `GeoJAGS`. The horizontal dashed line indicate the chain size (10,000 iterations).

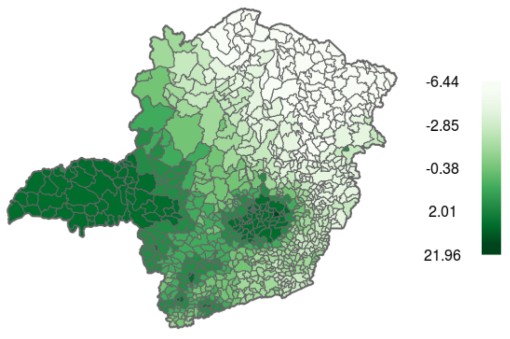
`OpenBUGS`, they clearly exhibit a higher spatial effect with impact to the response variable (GDP per capita) in some areas: the central part of Minas Gerais, where Belo Horizonte is located; the central western zone, known as Triangulo Mineiro (an area with high concentration of factories and agribusiness activities), and the southern portion of the state (having agriculture, mainly coffee production, as base of its economy). On the other hand, the northeastern region presents modest spatial random effects, as expected, for it is known as the poorest area of the state. Again, this proposed model is simple indeed and therefore may be improved by inclusion of some covariates such as the HDI (human development index), rate of citizens living in the urban area of the municipalities, education level, number of technology companies in the region, etc. By doing so the non-spatial variance is expected to reduce, leading to better estimates (see Section 5.3.2).

5.3 Simulation study

The results from the previous section motivated a simulation study to better evaluate the behavior of the the proposed `GeoJAGS` and `GeoBUGS` modules. The hierarchical model to be fitted is the same as described in Section 5.2, with the only difference being the response variable Y. Here the response is randomly generated complying with the constraints described in Section 5.3.1. In these simulation cases, we set the MCMC algorithm to perform 10,000 iterations, with a burn-in period of 5,000 and lag=1, forming a poste-



(a)



(b)

Figure 5.4: Posterior mean estimates of the random effects for the GDP data obtained from: (a) `GeoJAGS` and (b) `GeoBUGS` modules.

rior sample with size 5,000. For the computation of the effective sample sizes (explored ahead), we considered the full chain (including the burn-in period).

5.3.1 Case 1: data variance grater than spatial variance

Here we consider $\sigma^2 \gg \tau^2$, reproducing the scenario of the real data set. Let $\sigma^2 = 340$, chosen according to the magnitude of the data variance, and $\tau^2 = 0.66$, which is a value within the HPD interval from `GeoJAGS` (see Table 5.1). The neighborhood structure in this simulation study involves the 853 municipalities of Minas Gerais. Using the `R` software, we generated:

$$\boldsymbol{\theta} \sim N_{853}(\mathbf{0}, \boldsymbol{\Sigma}),$$

where $\mathbf{0}$ is a 853-dimensional null vector, $\boldsymbol{\Sigma}^{-1} = \frac{1}{\tau^2}(D_w - \rho W)$ with $\rho = 0.9$. In addition, we set $\mu = 15$ to generate the response $\boldsymbol{Y} = (Y_1, \ldots, Y_n)$, with $Y_i \sim N(\mu + \theta_i, \sigma^2)$. The generated spatial random effects lie in the interval $[-1.77, 1.52]$, and $Y_i \in [-44.76, 82.68]$.

With this synthetic data set, we run the model scripts in Box 5.1 and Box 5.2. The range of the random effect posterior means is $[-0.06, 0.07]$ from `GeoJAGS` and $[-0.22, 0.31]$ from `GeoBUGS`. Further, their respective mean squared errors are 0.211 and 0.218. The chains of some parameters in this model are presented in Appendix D (page 113). The estimated spatial random effects are shown in Figure 5.5 along with the real $\boldsymbol{\theta}$. Note that the estimates do not convey much information regarding the original random effects. As

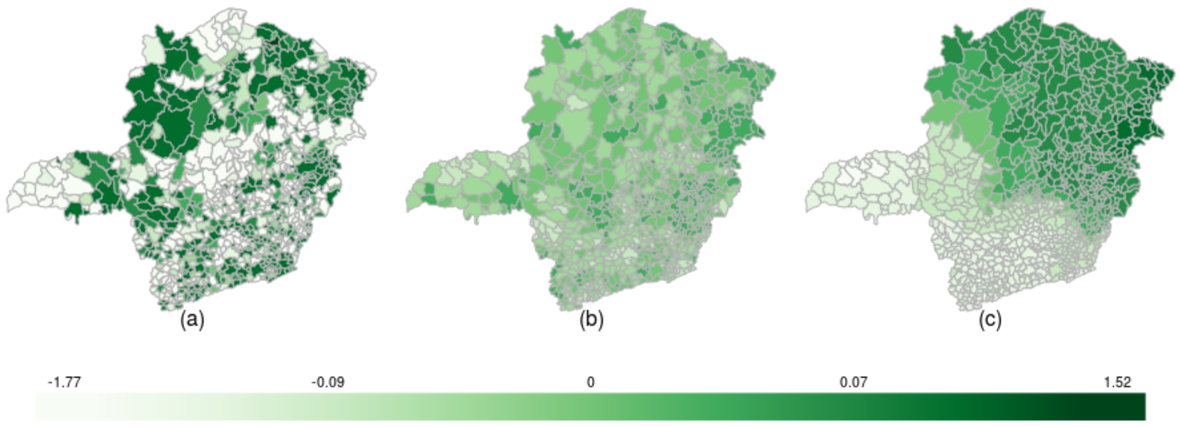


Figure 5.5: Comparison between (a) synthetic spatial random effects and corresponding posterior mean estimates from (b) `GeoJAGS` and (c) `GeoBUGS`.

mentioned before, this can be expected since the non-spatial variance effect σ^2 is rather greater than the spatial variance effect τ^2.

Table 5.2 shows the real values and the corresponding estimates from the two modules. Whilst the true value of the nonspatial variance $\sigma^2 = 340$, the estimates from both modules are approximately 351 (with similar standard deviations close to 17), however, the corresponding 95% HPD intervals contain the real value. For the remaining parameters, the performance of `GeoJAGS` and `GeoBUGS` are similar in terms of parameter estimation and HPD interval computation, with the main difference being related to the HPD interval of the random effects (only estimates for the random effect θ_{66} is shown in Table 5.2).

Figure 5.6 presents the ESS (based on `effectiveSize` function from the `coda` package) for the variance parameters and the ESS for 10 randomly selected spatial effects.

	Real	GeoJAGS			GeoBUGS		
		Mean (SD)	95% HPD interval		Mean (SD)	95% HDP interval	
μ	15.00	14.61 (0.64)	13.38	15.85	14.61 (0.64)	13.30	15.82
σ^2	340	351.85 (16.89)	318.05	383.74	351.61 (16.87)	319.10	384.50
τ^2	0.66	0.64 (0.43)	0.13	1.53	0.62 (0.38)	0.13	1.53
θ_{66}	0.01	0.01 (0.33)	-0.68	0.63	-0.03 (0.43)	-0.88	0.85

Table 5.2: Comparison of estimates for the two spatial modules for the simulation study with $\sigma^2 \gg \tau^2$. Only one spatial random effect is shown, representing Belo Horizonte. Standard deviations are shown in parentheses.

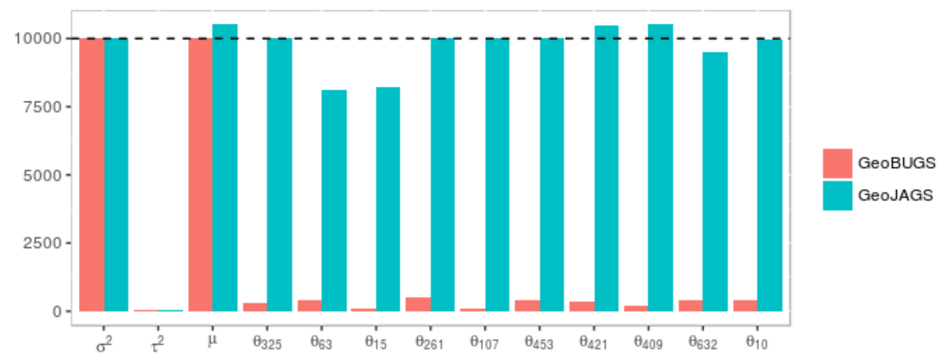


Figure 5.6: Effective sample size of the chains for the variance parameters (σ^2 and τ^2) and for 10 randomly selected spatial random effects (θ_i). The left bars display the ESS from `GeoBUGS` and the right bars the ESS from `GeoJAGS`. The horizontal dashed line indicates the size of the chain (5,000 iterations).

Observe that the ESS formulation allows values grater than the actual size of the posterior chain. Overall, the figure shows that the posterior chains from `GeoJAGS` are considerably less autocorrelated than those from the `GeoBUGS` module. Considering all parameters in the model, the average ESS of the chains from `OpenBUGS` is 415 (1st qu.: 232; 3rd qu.: 496), while from `JAGS` this quantity is 9,936 (1st qu.: 9,687; 3rd qu.: 10,019). Note that the parameter with the higher autocorrelation (and thus the lower ESS) is the spatial variance τ^2.

In addition, we explore the relative bias, defined as

$$RB = \frac{\hat{\alpha} - \alpha}{|\alpha|}, \tag{5.2}$$

for any parameter α in a model, with α being the true value and $\hat{\alpha}$ the estimated value (without the burn-in period). This ratio can also be expressed as a percentage if multiplied by 100. Since the spatial random effects θ_i are close to zero and due to the form of the relative bias equation, the RB for some parameters took large values (in the order of 10^2). The mean RB from `GeoBUGS` and `GeoJAGS` are -0.50 (1st qu.: -1.11; 3rd qu.: 1.01) and -0.01 (1st qu.: -1.00; 3rd qu.: 1.00), respectively. The range of relative bias from `OpenBUGS` is $[-259.20, 103.57]$ and from `JAGS` is $[-17.65, 14.75]$. Ultimately, the percentage of RB (in absolute value) from our proposed module that are less than those from `GeoBUGS` is 55%.

5.3.2 Case 2: data variance smaller than spatial variance

In this simulation study, we will consider that the non-spatial variance is less than the spatial variance, i.e. $\sigma^2 < \tau^2$. The data generation here follows the same specification described in Section 5.3.1, with the only difference being the values $\sigma^2 = 0.1$ and $\tau^2 = 5$. The generated random effects lie in the interval $[-4.84, 4.16]$, whereas $Y_i \in [9.89, 18.91]$.

Again, we use the softwares `JAGS` and `OpenBUGS` to run the model (scripts in Box 5.1 and 5.2). Figure 5.7 presents the spatial distribution of the true random effects along with the estimated means from `GeoJAGS` and `GeoBUGS` modules. The range of the spatial random effect posterior means is $[-4.40, 3.48]$ for `GeoJAGS` and $[-4.70, 3.64]$ for `GeoBUGS`. Furthermore, the respective mean squared errors are 0.104 and 0.145. Note that the proposed module gives estimates slightly closer to the synthetic data than `GeoBUGS`. In addition, the spatial distribution of the random effects are very similar in both spatial modules, resembling the original data.

The posterior means for μ, σ^2, τ^2 and for one spatial random effect (θ_{66}) are presented in Table 5.3. The global mean μ estimated by the two modules are close to 15 (the real value). Differently from the previous simulation case, the non-spatial variance σ^2 estimated from `GeoJAGS` is closer to the real value 0.10 when compared with the result from `GeoBUGS`. For the estimated spatial variance τ^2, only the mean estimate from our module is close to the true value; the HPD interval from `GeoBUGS` does not contain the

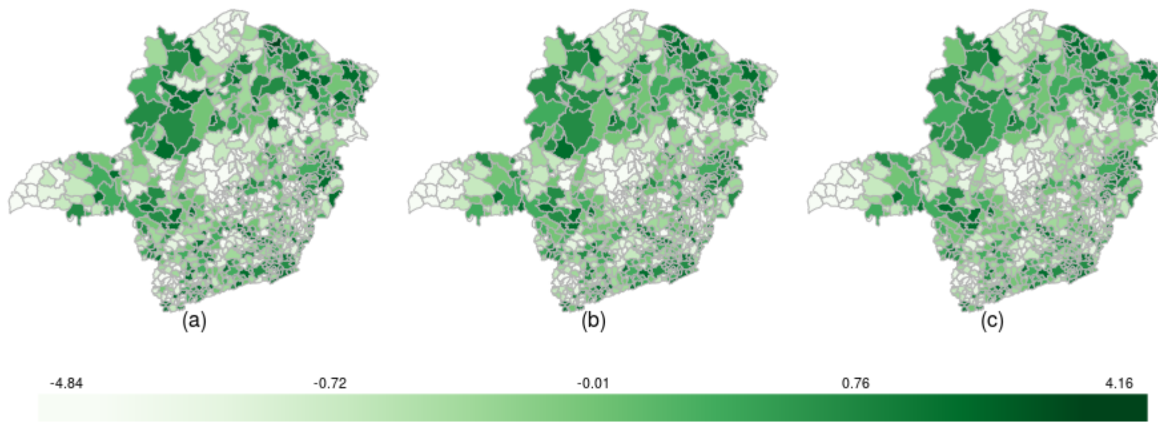


Figure 5.7: Comparison between (a) synthetic spatial random effects and corresponding posterior mean estimates from (b) `GeoJAGS` and (c) `GeoBUGS` modules.

	Real	GeoJAGS		GeoBUGS	
		Mean (SD)	95% HPD interval	Mean (SD)	95% HDP interval
μ	15.00	14.96 (0.10)	14.76 15.16	14.96 (0.02)	14.91 14.99
σ^2	0.10	0.23 (0.07)	0.11 0.36	0.35 (0.08)	0.20 0.51
τ^2	5.00	4.83 (0.43)	3.95 5.61	3.43 (0.39)	2.72 4.23
θ_{66}	0.03	0.05 (0.42)	-0.79 0.84	0.00 (0.45)	-0.83 0.94

Table 5.3: Comparison between the real values and the parameter estimates from the two spatial modules. For simplicity, only one spatial random effect is shown (θ_{66}), index chosen at random. Standard deviation is shown in parentheses.

real value of this parameter. Regarding the spatial random effect, both HDP intervals contain the true value of θ_{66}.

Figure 5.8 shows the ESS values for the variance parameters and for 10 randomly selected spatial effects (the indices here are the same chosen in Section 5.3.1). In this controlled situation, we can see that the chains from `GeoJAGS` and `GeoBUGS` are similar in effective size. Again, the parameters with the higher autocorrelation (and thus the lower ESS) are the spatial and non-spatial variances, regardless the module used to fit the data. Considering all parameters in the model, the average ESS of the chains from `OpenBUGS` is 7,459 (1$^{\text{st}}$ qu.: 5,497; 3$^{\text{rd}}$ qu.: 9,413), while from `JAGS` the average is 7,727 (1$^{\text{st}}$ qu.: 5,489; 3$^{\text{rd}}$ qu.: 10,000). Clearly, both modules present similar performance in terms of effective size of the chains, differently from the case when $\sigma^2 \gg \tau^2$, which `GeoJAGS` shown a better performance than `GeoBUGS` (see again Figure 5.6).

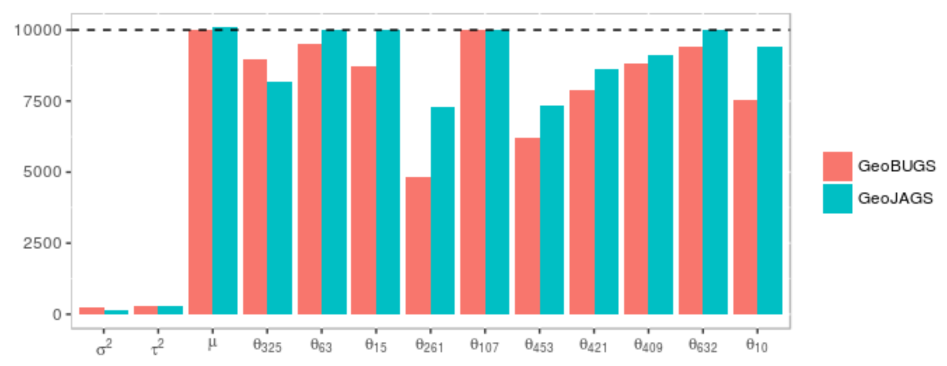


Figure 5.8: Effective sample size of the chains for the variance parameters (σ^2 and τ^2) and for 10 randomly selected spatial random effects (θ_i). The left bars display the ESS from `GeoBUGS` and the right bars the ESS from `GeoJAGS`. The horizontal dashed line indicates the size of the chain (10,000 iterations).

In addition, we compute the relative bias as defined in (5.2) for each parameter in the model. The range of relative bias from `GeoBUGS` is $[-108.97, 73.82]$ and from `GeoJAGS` is $[-85.96, 73.78]$. The mean RB when using `GeoBUGS` and `GeoJAGS` are 0.22 (1^{st} qu.: $-0.28; 3^{\text{rd}}$ qu.: 0.36) and 0.03 (1^{st} qu.: $-0.23; 3^{\text{rd}}$ qu.: 0.32), respectively. Ultimately, the percentage of RB (in absolute value) from our proposed module that are less than those from `GeoBUGS` is 67%.

5.4 Execution times

Currently, when the number of regions is large (in our simulation cases, $n = 853$), the execution time of our proposed implementation of the CAR model is larger than that of `GeoBUGS`. For the simulation case presented in Section 5.3.1, the `OpenBUGS` script run in 21.90 seconds, while in `JAGS` took 10,408.07 seconds in a Ubuntu 16.04 machine with Intel Core i7-3770 x8 processor and 8GB memory. In Section 5.3.2, the execution time from the `GeoBUGS` module was 22.13 seconds and from `GeoJAGS` was 10,996.04 seconds using the same machine. This is due to the fact that we use the multivariate normal distribution together our CAR matrix function (computing the precision matrix). The implementation of the `dmnorm` sampler performs the eigen-decomposition of the precision matrix (see Section 3.4.2) in every iterate of the MCMC algorithm. Such task is time consuming.

Despite the time taken, the chains produced by our proposed model are less autocorrelated due to blocking sampling. Again, based on some evidences, we believe that the `GeoBUGS` CAR function samples individually from all full conditional posterior distributions, which reduces both the execution time and the the effective sample size of the chains. Thus, in this case, the `JAGS` user shall consider the loss in terms of computational speed and the gain in quality of estimated, as previously shown in the simulation studies. Also, taking into account the simpler script needed to implement the CAR model in `JAGS` than that needed in `OpenBUGS`.

We are already working on improvements in the CAR implementation for the `GeoJAGS` module that can drastically reduce the execution time of the CAR model in our proposed module. Again, some details about this can be found in Section 3.4.2 and Appendix B.

Chapter 6

Using GeoJAGS with real data sets

The main motivation of this work is to simplify spatial statistics modeling in a hierarchical Bayesian approach using JAGS. In this chapter, we present two real applications from different research areas: climatology and medical science. It is important to emphasize that we are not proposing any new modeling strategy here. The idea is to show how the GeoJAGS module can be used in different applications. Section 6.1 presents climate data from the north region of Brazil, where part of the Amazon rain forest is located. The main focus will be on performing prediction on temperature and precipitation at arbitrary locations for which no information is available within the region. Finally, we conclude this chapter in Section 6.2 with an application on the survival of patients diagnosed with acute myeloid leukemia in northwest England.

6.1 A study on the Brazilian North region climate

The Amazon is a vast region that spans across eight countries (Brazil, Bolivia, Peru, Ecuador, Colombia, Venezuela, Guyana, Suriname and French Guiana). According to World Wild Life (2017), its landscape contains one in ten known species on Earth, half of the planet's remaining tropical forests, 4,100 miles of winding rivers, 2.6 million square miles in the Amazon basin. There is a clear link between the health of the Amazon and the health of the planet, since the rain forests help stabilize local and global climate. Deforestation may release significant amounts of carbon, which could have catastrophic

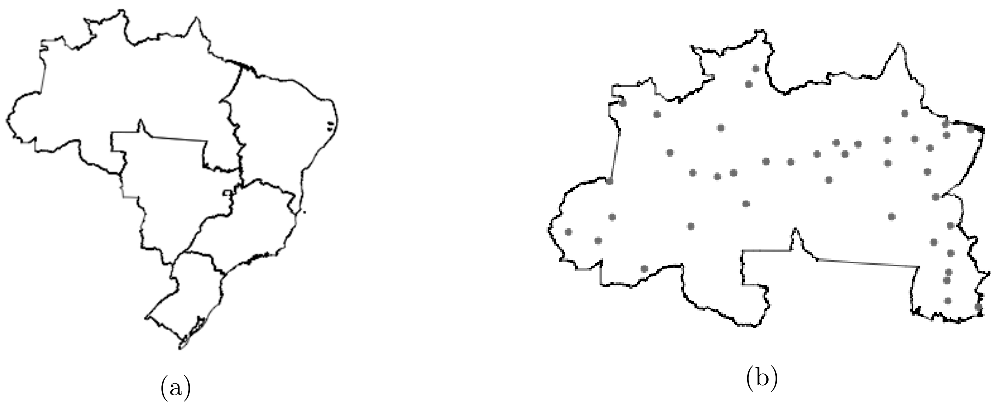


Figure 6.1: Map of Brazil with (a) its geopolitical division into five regions and (b) the North region additionally showing the location of the meteorological stations in this region.

consequences around the world.

The Brazilian Amazon is mainly located in the north region, the largest region of Brazil, corresponding to 45% of the national territory. Information regarding the weather in the country is provided by the Brazilian National Institute of Meteorology (Instituto Nacional de Metereologia, INMET), see INMET (2017). Its database, named BDMEP (acronym for meteorological database for teaching and research), stores daily meteorological measurements of the various conventional weather stations in the INMET stations network. The atmospheric variables available for consultation are: precipitation in the last 24 hours, dry-bulb temperature[1], wet-bulb temperature[2], maximum temperature, minimum temperature, relative humidity, atmospheric pressure at station level, insolation, and direction and wind speed.

We will retain our attention to temperature and precipitation measurements taken in the north region of Brazil. Section 6.1.1 presents a hierarchical spatial model for the precipitation and Section 6.1.2 investigates the temperature. Figure 6.1 presents the geopolitical division of Brazil into five regions and the north region with the marked locations of the INMET's 41 meteorological gauges. Note that the gauges are not regularly distributed throughout the space and the total number of stations is small given the area of the region (3,853,676.9km^2). Therefore, we aim to predict precipitation amounts and

[1]The dry-bulb temperature is the temperature of the air measured by a thermometer freely exposed to the air, but shielded from radiation and moisture.

[2]The wet-bulb temperature is the lowest temperature which may be achieved by evaporative cooling of a water-wetted (or even ice-covered), ventilated surface.

the average temperatures at arbitrary locations over the map.

Although the BDMEP database is available in the INMET's website, some effort is required to get and manipulate the data before fitting any statistical model. The procedure for extracting the data from the website is as follows: 1) log in into the system (after creating an account); 2) choose the atmospheric variables and the time span accordingly; 3) a map of Brazil will be displayed with all stations; 4) in that map, click on the desired station, then select the option for downloading the data and finally download the generated text file; 5) repeat this procedure to obtain the data from each station (in our case, 41 stations). In order to eliminate this tiresome work, we have used the `R` package `rvest` (see Wickham, 2016), which enables us to simulate a session in the web browser and automatically perform the numerous steps needed to download the data from all 41 meteorological stations. In addition, we manipulated the obtained text files using the `dplyr` package (see Wickham et al., 2017).

Before discussing the Bayesian models for prediction, we describe the approach used to visualize the resulting estimates. When willing to construct a smooth map of predictions, the number of locations to predict must be large, due to size of the study area. However, this fact makes the `JAGS` model very inefficient, since we specify a multivariate normal distribution for a vector parameter and, as stated before, sampling from this distribution is time consuming when the number of regions is large. Thus, we need to specify the number of points for prediction taking into account a threshold between execution

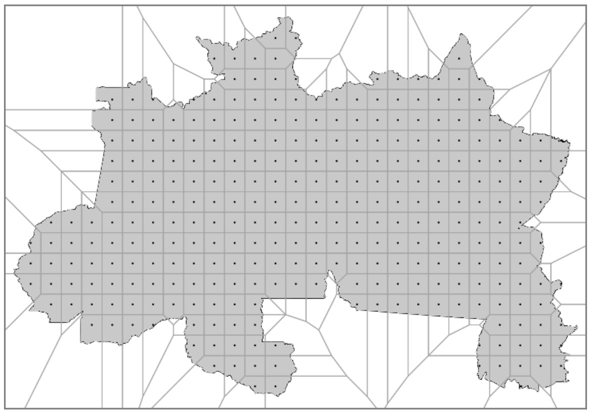


Figure 6.2: Voronoi diagram for the Brazilian north region with the grid of points for prediction of precipitation and temperature.

time and smoothness of the resulting map. In our models, we will predict precipitation amounts and average temperatures on 301 points of a regular grid within the north region. The estimates for these locations will be shown in a Voronoi diagram (Aurenhammer, 1991, see), which is a partitioning of a plane into regions based on distance to points in a specific subset of the plane. That set of points (the locations for prediction) is pre-specified. For each point $\boldsymbol{s_i}$ in that set, there is a corresponding region consisting of all points closer to $\boldsymbol{s_i}$ than to any other point in the grid. Thus, each Voronoi region will receive color related to the value predicted in that location. In `R`, we have used the functions provided by the `deldir` package (see Turner, 2017) to produce Figure 6.2.

6.1.1 Modeling precipitation

Let $Y_{obs,i} = Y(\boldsymbol{s}_{obs,i})$ be the total precipitation (in millimeters) of the station located at $\boldsymbol{s}_{obs,i}$ ($i = 1, \ldots, n = 41$). We will analyze the model in two different periods of time, namely the "rainy month" and the "driest month", March and September, respectively, of the years 2006 and 2016. As a baseline for comparison of the predictions, consider the maps provided by CPRM (2017); see Appendix D. These maps were build considering precipitation measurements between the years 1977 and 2006 and the data is provided in maps of monthly averages. Thus we decided to analyze precipitation in 2006 and after a ten year period, in 2016. We are interested in modeling the geographic distribution of rainfall in northern Brazil using the hierarchical Bayesian approach. It is important to remark that the models presented here are not spatio-temporal models. The same model is used to fit the data from different periods of time.

In total, there are 41 stations in the system, however some of them have missing values. In the years of 2006 and 2016, there are records of precipitation amount in 40 and 38 meteorological stations, respectively, ranging from 0.4mm to 757.5mm in the month of March. Considering September, the data set contains information about 40 and 37 gauges in the years of 2006 and 2016, respectively. The range of the measurements is $[1.40, 252.52]$mm.

It is clear that the response variable can only take non-negative values. In addition, the data has an asymmetric heavy-tailed distribution in the right tail. Thus we consider the following model:

$$Y_{obs,i} \mid \mu_{obs,i},\ \alpha \sim Ga(\alpha, \alpha/\mu_{obs,i}), \tag{6.1}$$

where $\mu_{obs,i} = \mu(\boldsymbol{s}_{obs,i}) = \exp\{\beta_0 + \beta_1 s_{obs,i1} + \beta_2 s_{obs,i2} + \beta_3 s_{obs,i3} + \omega_{obs,i}\}$, $s_{obs,i1}$, $s_{obs,i2}$ and $s_{obs,i3}$ are the latitude, longitude and elevation of the observed station, respectively. In addition, $\alpha > 0$ is the shape parameter of the distribution, and this specification defines the gamma-distributed variable to have mean $\mu_{obs,i}$ and variance $\mu^2_{obs_i}/\alpha$. The random effects $\omega_{obs,i}$ will be discussed ahead in this section.

To the best of our knowledge, we emphasize that currently it is not possible to fit spatial models with non-Gaussian response variables using the `GeoBUGS` module. However, by using the proposed module, we are able to execute this Gamma model through `JAGS`. This is another contribution of our work.

In the north region, the latitude ranges from S13°20′ to N4°22′, while the longitude ranges from W73°15′ to W46°50′, and the elevation is in the interval $[-14.05,\ 1{,}235.76]$m. Since the covariates are introduced in the exponential function and due to their magnitude, we adjusted their values by dividing latitude, longitude and elevation[3] by 100. This adjustment was made after observing computational issues and some coefficient being estimated close to zero. The real impact of this modification in the estimation of the β_i coefficients is only seen in the scale of the estimated values, which will have their values multiplied by 100.

We will perform prediction on a regular grid that contains $m = 301$ points throughout the north region. They all have their geographical coordinates and elevation measurements available. These points were obtained using the `R` package `sp` (see Pebesma and Bivand, 2005). The function `spsample` regularly samples locations within a polygon (the shape of the study region), given the desired number of points, in our case we consider 300. This function yields a grid having approximately the pre-specified number of location, that is the reason we have 301 points instead of exactly 300. For each location,

[3]We are aware of the high mountains in the further north part of the study region, specially those in the Roraima state. In this case, we considered applying the logarithmic function to the elevation measurements in order to reduce their scale. We fit the data considering this new scale (log). However, the resulting estimates did not show a considerable difference if compared to the estimates given from the model which considers the elevation dhivided by 100.

we collected its elevation measurement from Google's database by using the `googleway` package (see Cooley, 2017).

The aim is to predict the response at these locations given the observed data $Y_1, \ldots, Y_n$. Let $Y = (Y_{obs,1}, \ldots, Y_{obs,n})^\top$ be the n-vector containing the observed values, with $Y_{obs,i}$ being as defined in (6.1), and $Y_{pre} = (Y_{pre,1}, \ldots, Y_{pre,m})^\top$ be the m-vector with the responses to be predicted. We assume:

$$\begin{aligned} Y_{pre,i} \mid \omega_{pre,i}, \boldsymbol{\beta} &\sim Ga(\alpha, \alpha/\mu_{pre,i}), \quad i = 1, \ldots, m \\ \boldsymbol{\Omega} = (\boldsymbol{\Omega}_{obs}, \boldsymbol{\Omega}_{pre}) &\sim N_{n+m}(\mathbf{0}, \boldsymbol{\Sigma}(\boldsymbol{\theta})) \end{aligned} \tag{6.2}$$

where $\mu_{pre,i} = \exp\{\beta_0 + \beta_1 s_{pre,i1} + \beta_2 s_{pre,i2} + \beta_3 s_{pre,i3} + \omega_{pre,i}\}$, $\boldsymbol{\beta} = (\beta_0, \beta_1, \beta_2, \beta_3)^\top$ is the vector of regression coefficients, $\boldsymbol{\Omega} = (\omega_{obs,1}, \ldots, \omega_{obs,n}, \omega_{pre,1}, \ldots, \omega_{pre,m})^\top$ is a vector containing both observed and unobserved spatial random effects, and $\boldsymbol{\Sigma}(\boldsymbol{\theta})$ is the covariance matrix. In this model, we will adopt the Gaussian covariance function, thus $\boldsymbol{\theta} = (\sigma^2, \tau^2, \phi)$. The reason for such choice of covariance structure is due the fact that it is smooth and perhaps the most popular. Similar response values are expected in nearby locations when using a smooth covariance function. In addition, the prior specifications are:

$$\alpha \sim Ga(a_\alpha, b_\alpha), \qquad \phi \sim Ga(a_\phi, b_\phi), \qquad \sigma^2 \sim IG(a_\sigma, b_\sigma), \qquad \boldsymbol{\beta} \sim N_4(\boldsymbol{M_\beta},\ \boldsymbol{V_\beta}).$$

We chose a vague prior for α by setting $a_\alpha = 1$, and $b_\alpha = 1/8$, which gives mean 8 and variance 64. We also set $a_\phi = b_\phi = 0.1$, which provides $\mathbb{E}(\phi) = 1$ and $\mathbb{V}\text{ar}(\phi) = 10$. For σ^2, we choose $a_\sigma = 2.1$ and $b_\sigma = 1.1$, providing an inverse-gamma distribution with mean 1 and variance 10. Finally, for the vector of coefficients we chose $\boldsymbol{M_\beta} = \mathbf{0}$ and $\boldsymbol{V_\beta} = 10\boldsymbol{I_4}$.

The `JAGS` script for this hierarchical model is in Box 6.1. Note that the prediction is performed in the same manner as described in Section 4.5. In line 16, we use the Gaussian exponential function from the `GeoJAGS` module, considering the powered exponential function with $p = 2$ (alternatively, one may use the `CVGaussian` function, an encapsulation of the Gaussian covariance structure, since it is a particular case of the powered

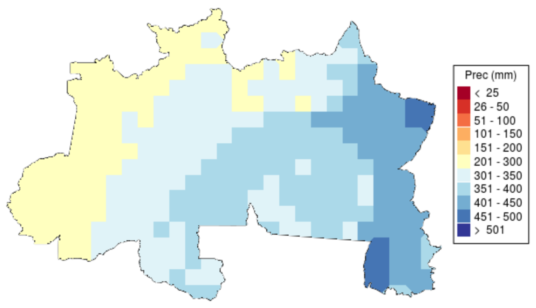
Box 6.1: JAGS script for the precipitation data.

```
model{
  for (i in 1:Nobs) {
    Y[i] ~ dgamma(alpha, eta[i])
    mu[i] <- exp(beta[1] + beta[2]*coords[i,1] + beta[3]*coords[i,2] + beta[4]*alt[i] + w[i])
    eta[i] <- alpha/mu[i]
  }
  for (i in 1:Npred) {
    Y.pred[i] ~ dgamma(alpha, eta[Nobs+i])
    mu[Nobs+i] <- exp(beta[1] + beta[2]*coords[Nobs+i,1] + beta[3]*coords[Nobs+i,2]
                      + beta[4]*alt[Nobs+i] + w[Nobs+i])
    eta[Nobs+i] <- alpha/mu[Nobs+i]
  }
  alpha ~ dgamma(a_alp, b_alp)
  w[1:(Nobs+Npred)] ~ dmnorm(mu.w, Omega.w)
  beta ~ dmnorm(mu.beta, Omega.beta)
  Sigma <- CVPowerExp(coords, sig2, phi, tau2, 2)
  Omega.w <- inverse(Sigma)
  inv.sig2 ~ dgamma(a_inv.sig2, b_inv.sig2)
  sig2 <- 1/inv.sig2
  phi ~ dgamma(a_phi, b_phi)
}
```

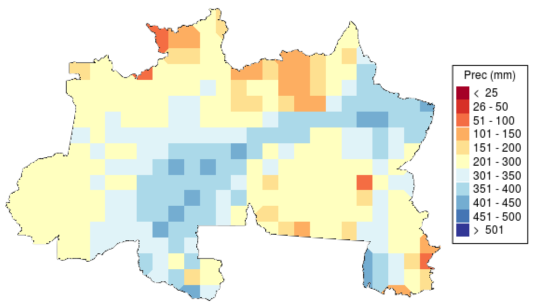
exponential function). Recall that the parameter `tau2` is the non-spatial variance effect of the model, but the shape parameter α (`alpha` in the model script) is also responsible for accommodating the non-spatial variance, therefore `tau2` is set to be zero. We set the burn-in period with 100,000 iterations and lag=50 providing a chain with 1,000 values.

The posterior mean of the four studied months are shown in Figure 6.3, using the Voronoi diagram with the 301 predicting locations. In the first row, we show the estimates for the historically rainier month, March 2006 and 2016, (Figure 6.3 (a) and (b), respectively). In the former, the direction from central to east region presents the pattern of increasing precipitation amounts, whereas the northernmost and further west presents the least precipitation estimates. This result is in accordance with the map shown in Figure D.3a (page 114), a map that presents the monthly mean precipitation amount considering measurements taken between the years 1977 and 2006. In March 2016, the highest estimated values seems to follow the Madeira river waterway. This is one of the biggest tributaries of the Amazon river, accounting for about 15% of the water in the basin (see Amazon Waters, 2018). Again, the southeast area also has higher estimates, although the values are lower than those from March 2006. The second row presents estimates for September, the historically driest month in that region for years 2006 and 2016.

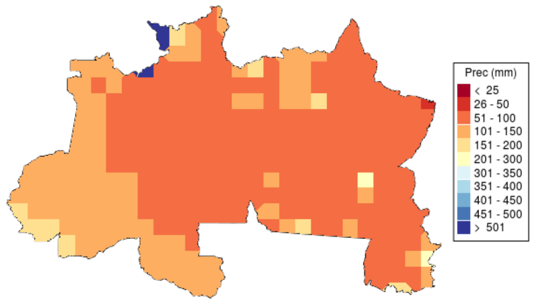
In the 2006–07 period, the Earth experienced a weak El Niño event, whereas in the period between 2015 and 2016, another El Niño event was registered, being considered



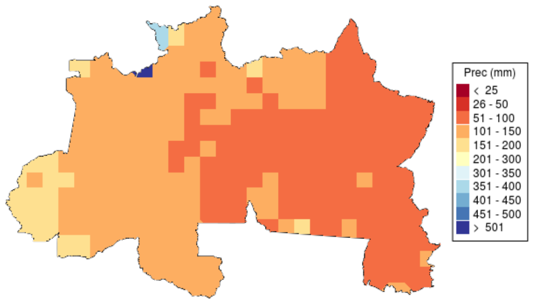
(a)



(b)



(c)



(d)

Figure 6.3: Voronoi diagram of the posterior mean estimates for the monthly precipitation (in millimeters) in the north region: (a) March 2006, (b) March 2016, (c) September 2006, and (d) September 2016.

a very strong climate event, according to National Oceanic and Atmospheric Administration, see NOAA (2018). The impact of this event varies across the world. When this event happens, there is a reduction in the amount of rain fall in some areas of the Amazon forest, especially those located further to the north and also to the east. This fact can be observed in the first row of Figure 6.3: considering the estimates, March 2016 was drier than March 2006. Meanwhile, the month of September showed a similar pattern in both years. Overall, the two maps match the pattern displayed in Figure D.3b (page 114), with a driest portion being located in the central-eastern region.

6.1.2 Modeling temperature

Now we consider the measurements of the mean temperature in the 41 meteorologic stations. We believe there is association between temperature and pluviometric mea-

Box 6.2: JAGS script for the temperature data.

```
model{
  for (i in 1:Nobs) {
    Y[i] ~ dnorm(mu[i], prec.y)
    mu[i] <- beta[1] + beta[2]*coords[i,1] + beta[3]*coords[i,2] + beta[4]*alt[i] + w[i]
  }
  for (i in 1:Npred) {
    Y.pred[i] ~ dnorm(mu[Nobs+i], prec.y)
    mu[Nobs+i] <- beta[1] + beta[2]*coords[Nobs+i,1] + beta[3]*coords[Nobs+i,2] + beta[4]*alt[Nobs+i]
          + w[Nobs+i]
  }
  w[1:(Nobs+Npred)] ~ dmnorm(mu.w, Omega.w)
  beta ~ dmnorm(mu.beta, Omega.beta)
  Sigma <- CVPowerExp(coords, sig2, phi, tau2, 2)
  Omega.w <- inverse(Sigma)
  inv.sig2 ~ dgamma(a_inv.sig2, b_inv.sig2)
  sig2 <- 1/inv.sig2
  phi ~ dgamma(a_phi, b_phi)
  prec.y ~ dgamma(a_y, b_y)
  nu2 <- 1/prec.y
}
```

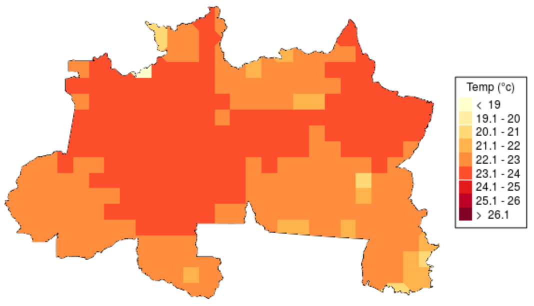
surements explored in the previous section. Thus, we also analyze the model in two different periods of time, namely the "rainy month" and the "driest month" of the years 2006 and 2016.

Missing values are also registered for this data set. There are less missing values for the minimum average temperature, therefore we decided to model this variable instead of the maximum average temperature. In fact, the behavior of these two variables is the same, with the difference between them being their magnitude. In March, there are records on minimum average temperature in 40 and 37 meteorological stations, respectively, ranging from 19.37°C to 26.41°C in the years of 2006 and 2016. For September, the data set contains information about 41 and 36 gauges in 2006 and 2016, respectively. The range of the measurements is $[19.93, 26.93]$°C.

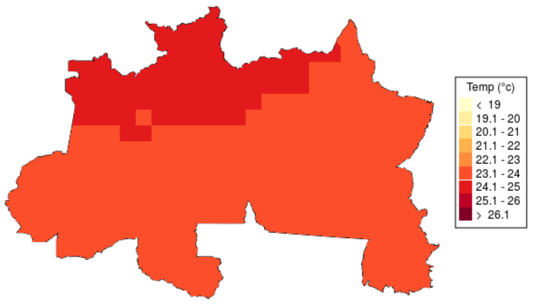
The histogram of the observed data suggests a symmetric distribution for the response in the model. In addition, given the absence of constraints on the parameter space of the response variable, we set temperature to be normally distributed, unlike the model of precipitation. Let

$$Y_{obs,i} \mid \mu_{obs,i} \sim N(\mu_{obs,i}, \nu^2), \tag{6.3}$$

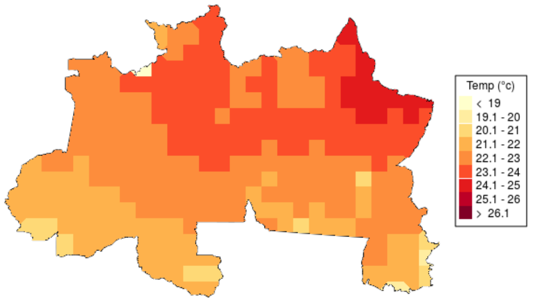
where $\mu_{obs,i} = \beta_0 + \beta_1 s_{obs,i1} + \beta_2 s_{obs,i2} + \beta_3 s_{obs,i3} + w_{obs,i}$, with $s_{obs,i1}$, $s_{obs,i2}$ and $s_{obs,i3}$ being the latitude, longitude and the elevation of the station $s_{obs,i}$, respectively. Differently from the model for precipitation, this model can also be executed using `GeoBUGS`, however, the results presented here are from the `GeoJAGS` module.



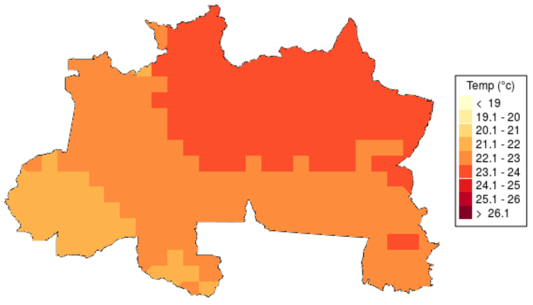
(a)



(b)



(c)



(d)

Figure 6.4: Voronoi diagram with posterior mean estimates for monthly precipitation (in Celsius degrees) in North region for (a) March 2006, (b) March 2016, (c) September 2006 and (d) September 2016.

The locations for predicting the response variable are the same $m = 301$ used in the previous section. We aim to predict the response (minimum average rainfall amounts) at these locations given the observed data $Y_1, \ldots, Y_n$. The specification of the prediction vector is the same as in (6.2), except for the fact that the response variable is normally distributed, as can be seen in the corresponding `JAGS` script in Box 6.2. The prior specifications for the common parameters are the same as in the previous section and we set $1/\nu^2 \sim G(a_\nu, b_\nu)$ with $a_\nu = b_\nu = 0.1$ (we are adopting this prior for precision, see Box 6.2, line 17). The MCMC configuration is the same as in the previous section: burn-in period of 100,000 iterates, thinning rate of 50 and final chain size of 1,000.

The Voronoi diagrams with the 301 locations and their corresponding predicted minimum temperature values are exhibited in Figure 6.4. In the first row, panels (a) and (b), we show the estimates for the historically rainier month, March 2006 and March 2016. In the former, the central north part and the northeastern region presents the highest predicted values. In the latter, the prediction was smoothed throughout the region with higher estimates than the former. Regarding September 2006, in the second row, the estimated minimum temperature follows a similar pattern as shown in March 2006. September 2016 presents lower estimates for the farthest west area and an increase in the estimated minimum temperature can be noted as we move from the southwest to the northeast, in general presenting higher estimates for the minimum average temperature.

As mentioned in Section 6.1.1, 2016 was a year with a very strong El Niño event. The impact of the event is a decreasing in precipitation, and thus, it is expected to present higher temperatures. This pattern is observed in Figure 6.4: in general, the year 2016 was warmer and drier than 2006.

6.2 Leukemia survival data

Several fields of statistics can make use of spatial modeling and survival analysis is one of them. Survival analysis aims to investigate the expected duration of time until an event of interest happens, such as failure in mechanical systems or death in biological organisms, see Lawless (2011). It is widely used in medical applications to model the survival prognosis of patients with a potentially fatal medical condition. A peculiarity of survival data is the presence of incomplete observations, called censoring, and a crucial issue of this field is building models that accommodate censoring. Here we will consider the right-censoring case (the most commom censoring in practice), when some time-to-event outcomes are not observed, but are known to be greater than the observed times. With the `GeoJAGS` module, the user is not restricted to model data with this type of censoring, since we are modeling the (latent) spatial random effects separately from the response variable.

In survival analysis, random effects of different types can be introduced in the model by a *frailty* term. It is often of interest to consider that dependence between locations in space play an important role to explain the time response. In this case, we can include spatial random effects in the survival model through the frailties.

Spatial random effects included in survival models are usually intended to represent different spatially arranged clusters, such as geographical regions. In this case, we believe that random effects corresponding to regions close to each other might be comparable in magnitude. Let t_{ij} be the time to death or censoring for individual j in cluster i, $(j = 1, \ldots, n_i,\ i = 1, \ldots, D)$ and let $\boldsymbol{x}_{ij}$ be the vector of covariates for individual j in cluster i.

In a proportional hazards model, the unique effect of a unit increase in a covariate is multiplicative with respect to the hazard rate. The usual assumption of proportional hazards enables models of the form

$$h(t_{ij}; \boldsymbol{x}_{ij}) = h_0(t_{ij}) \exp\left(\boldsymbol{\beta}^\top \boldsymbol{x}_{ij}\right),$$

where $h_0(\cdot)$ is the baseline hazard, which can be parametrically or non-parametrically modeled, and $\boldsymbol{\beta}$ is the vector or regression coefficientes. Following Banerjee et al. (2004), when using the frailty setting, the proportional hazards model above is extended to

$$h(t_{ij}; \boldsymbol{x}_{ij}) = h_0(t_{ij}) \exp\left(\boldsymbol{\beta}^\top \boldsymbol{x}_{ij} + W_i\right), \tag{6.4}$$

where W_i is the cluster-specific frailty term. Note that the W_i $(i = 1, \ldots, D)$ are not observed, but a latent variable included in the model to explain an underlying spatial variation. The simple specification for the random effects is $W_i \overset{iid}{\sim} N(0, \sigma^2)$, however we consider the case which the random effects are not independent.

Let us consider the records on the survival of acute myeloid leukemia in $n = 1{,}043$ patients in northwest England, recorded between 1982 and 1998. This data set is explored in Henderson et al. (2002) and Zhou and Hanson (2017). These records are named as `LeukSurv` in the `spBayesSurv` package (see Zhou et al., 2017), a R package for implementing Bayesian spatial survival models. It is of interest to investigate possible spatial variation in survival after accounting for known covariates, including age, sex, white blood cell count (wbc) at diagnosis (truncated at 500 units, with 1 unit $=50 \times 10^9/L$) and the Townsend score (tpi) for which higher values indicate less affluent areas, with a range from -7 to 10. The median survival time is just over 6 months. As both exact residential locations of all patients and their administrative districts are available, we can fit two models using the geostatistical and areal model approaches.

In the literature, a variety of parametric forms are used for the baseline hazard h_0, such as gamma, lognormal and Weibull. As shown in Banerjee et al. (2004, p. 302), adopting the Weibull baseline hazard function in (6.4) produces

$$h(t_{ij}; \boldsymbol{x}_{ij}) = \nu t_{ij}^{\nu-1} \exp\big(\boldsymbol{\beta}^\top \boldsymbol{x}_{ij} + W_i\big).$$

Further, setting priors distributions for ρ, $\boldsymbol{\beta}$ and a distribution for the vector of random effects $\boldsymbol{W} = (W_1, \ldots, W_D)^\top$ completes the Bayesian hierarchical model specification. The joint posterior distribution of interest is

$$p(\boldsymbol{\beta}, \boldsymbol{W}, \nu, \boldsymbol{\theta} | \boldsymbol{t}, \boldsymbol{x}, \boldsymbol{\gamma}) \propto L(\boldsymbol{\beta}, \boldsymbol{W}, \nu, \boldsymbol{\theta}; \boldsymbol{t}, \boldsymbol{x}, \boldsymbol{\gamma})\, p(\boldsymbol{W}|\boldsymbol{\theta})\, p(\boldsymbol{\beta})\, p(\nu)\, p(\boldsymbol{\theta}), \tag{6.5}$$

where the first term in the right-hand side is the Weibull likelihood, the second one is the joint distribution of the random effects, $\boldsymbol{\theta}$ is the vector indexing the distribution of the random effects, and the remaining terms are the prior distributions. For this application our choices of priors are based on Henderson et al. (2002) and Zhou et al. (2017), setting a vague independent Gaussian specification for $\boldsymbol{\beta}$ (details ahead in the text). In addition, we will consider for $\boldsymbol{W}$ two different approaches: distance-based and neighborhood-based spatial models. In (6.5), let $\boldsymbol{t} = \{t_{ij}\}$, $\boldsymbol{x} = \{x_{ij}\}$ and $\boldsymbol{\gamma} = \{\gamma_{ij}\}$ denote the collections of times to death, covariate vectors and death indicators (e.g. censored=0, dead=1), respectively, for the subjects in all clusters.

Distance-based spatial model

In this model setting, we consider the following joint distribution for the frailties:

$$\mathbf{W} | \boldsymbol{\Sigma}(\boldsymbol{\theta}) \sim N_D\big[\mathbf{0}, \boldsymbol{\Sigma}(\boldsymbol{\theta})\big],$$

where $\boldsymbol{\theta} = (\sigma^2, \phi, p)$ and $\boldsymbol{\Sigma}$ is the powered-exponential covariance matrix, as suggested in Henderson et al. (2002, p. 967). Again, as stated in Section 2.2.1, σ^2 is the spatial variance effect and $1/\phi$ is the range parameter (the distance at which the covariance

Box 6.3: JAGS script for the distance-based model.

```
model{
  for (i in 1:n) {
        is.censored[i] ~ dinterval(t[i], c[i])
        t[i] ~ dweib(rho.h0, lambda[i])
        lambda[i] <- exp(beta[1] + beta[2]* age[i] + beta[3] * sex[i]
                  + beta[4] * wbc[i] + beta[5] * tpi[i] + w[district[i]])
  }

  w[1:D] ~ dmnorm(mu.w, Omega.w)
  Omega.w <- inverse(Sigma)
  Sigma <- CVPowerExp(coords, sig2.w, phi.w, 0, 1.5)
  inv.sig2 ~ dgamma(2.01, 1.01)
  sig2.w <- 1/inv.sig2
  inv.phi ~ dgamma(2.01, 1.01)
  phi.w <- 1/inv.phi

  beta ~ dmnorm(mu.beta, Prec.beta)
  rho.h0 ~ dgamma(0.01, 0.01)
}
```

between two points vanishes).

The script in Box 6.3 presents the `JAGS` model for this data set using the covariance function from our module. We describe the parameters in the model in the order they appear in the script. Censoring is handled in `JAGS` using the `dinterval(t[i], c[i])` distribution and the right-censored survival model is set as follows: when the failure time is uncensored then t_i is observed and the censoring indicator `is.censored[i]` is zero (this vector must be passed as data in the `JAGS` script); when the failure time is right-censored (i.e. we only know that $t_i > c_i$) then t_i is unobserved (a missing value) and the censoring indicator `is.censored[i]` is one.

The observed times are set to follow a Weibull distribution with shape ν and rate λ_i. As the covariate values are part of the term in the exponential function, some scale adjustments were made to avoid numerical problems. These modifications were made considering the scale and the shape of the covariates values: the age (in years) was divided by 100; the tpi, that lies in the interval $[-6.09, 9.55]$, was divided by 10; and for the wbc, we considered $\log(\text{wbc}+1)$, due to its range (from 0 to 500) and its asymmetry (median: 7.90, mean: 38.65). In addition, the observed/censored times (in days) were divided by 30 (transformed to months), which will only modify the scale parameter ρ of the Weibull.

In this model, the individuals within the same district shares common frailty terms. The powered exponential covariance function is assumed with $p = 1.5$, as suggested in Henderson et al. (2002). Note that this model does not include the non-spatial variance τ^2, therefore, the fourth argument in the `CvPowerExp` corresponding to this parameter

Box 6.4: JAGS script for the neighborhood-based model.

```
model {
  for (i in 1:n) {
      is.censored[i] ~ dinterval(t[i], c[i])
      t[i] ~ dweib(rho.h0, lambda[i])
      lambda[i] <- exp(beta[1] + beta[2] * age[i] + beta[3] * sex[i]
                 + beta[4] * wbc[i] + beta[5] * tpi[i] + w[district[i]])
  }
  w[1:D] ~ dmnorm(mu.w, precMatrixCAR(AdjMat, rho.car, inv.sig2))
  inv.sig2 ~ dgamma(2.01, 1.01)
  sig2 <- 1/inv.sig2
  rho.car <- 0.999

  beta ~ dmnorm(mu.beta, Prec.beta)
  rho.h0 ~ dgamma(0.01, 0.01)
}
```

is set to be 0. The parameters σ^2 and ϕ follow an inverse-gamma, with mean 1 and variance 100. The vector of coefficients is multivariate-normally distributed with mean $\mathbf{0}$ and covariance matrix $100\boldsymbol{I_5}$. Ultimately, the Weibull's shape parameter follows a gamma distribution with mean 1 and variance 100.

Neighborhood-based spatial model

In this case, we consider the CAR distribution for the frailty terms using a binary proximity matrix to indicate the first-order neighbors of each region. Box 6.4 shows the model script using the `GeoJAGS`. we set the parameter $\rho = 0.999$, since Zhou et al. (2017) suggest the improper CAR distribution and the proposed implementation of `GeoJAGS` is based on the proper CAR. In addition, $\sigma^2 \sim IG(2.01, 1.01)$, which gives an inverse-gamma distribution with mean 1 and variance 100. The specifications for the remaining parameters are the same as defined in the distance-based model.

The posterior estimates

In this application we consider the following MCMC configuration: burn-in = 200,000 iterations, lag = 10 and final chain size of 1,000. Here, we do not aim to perform model selection, instead our goal is to present results using our module, and show how it simplifies the modeling of spatial survival data in `JAGS`.

Together with the posterior mean of the coefficients from `GeoJAGS`, we present in Table 6.1 the results of a similar neighborhood-based CAR model from `spBayesSurv`. It is important to remark that `spBayesSurv` implements the transformed Bernstein poly-

Covariates		Neighborhood-based model		Distance-based model (`JAGS`)
		`spBayesSurv`	`GeoJAGS`	
Age	β_1	0.048 (0.003)	0.032 (1.368)	0.032 (0.991)
Sex	β_2	0.124 (0.111)	0.062 (0.217)	0.070 (0.221)
WBC	β_3	0.004 (0.001)	0.069 (0.070)	0.068 (0.068)
Townsend	β_4	0.054 (0.016)	0.024 (0.022)	0.022 (0.021)
Spatial variance		0.269 (0.105)	0.213 (1.650)	0.981 (0.714)

Table 6.1: Comparison of the posterior means for the coefficients and the spatial variances from `GeoJAGS` and `spBayesSurv`. The last column shows the distance-based model considering a Gaussian covariance structure using `GeoJAGS`. Standard deviations in parentheses.

nomial (BP) prior for the baseline hazard function, in which we can set the centering distribution to be the Weibull, loglogistic or lognormal. As we aim the comparison between the mean estimates, we set the Weibull as the centering distribution for the BP. For the CAR model, the posterior mean coefficients and variances are similar using both softwares, except for the sex coefficient β_2, being estimated by `spBayesSurv` as approximately the double of the estimates from `GeoJAGS`. The `spBayesSurv` does not implement distance-based models, thefore we cannot compare with the corresponding results from `GeoJAGS`, even though the estimates of the neighborhood-based case are similar to those from the CAR modeling. Note that the model with greater spatial variance estimate is the distance-based model. Standard deviations from `GeoJAGS` are slightly greater in both distance- and neighborhood-based models than those from the `R` package.

Figure 6.5 shows the estimated mean district frailties for the neighborhood-based model using `spBayesSurv` and `GeoJAGS`. The spatial arrangement of frailties are similar, showing a higher frailty effect with impact to the response variable in the northern and in the central part of the study area, indicating possible clusters in these regions.

Finally, considering the MCMC specifications described above, these models were executed in a Ubuntu 16.04 machine with Intel Core i7-3770 x8 processor and 8GB memory. The time taken to run the CAR model and the distance-based model using `GeoJAGS` was 16.57 minutes and 36.77 minutes, respectively, whereas using the `spBayesSurv` package took 512.90 minutes. Note that the computational times are considerably different, but there are differences in the compared models, as stated before.

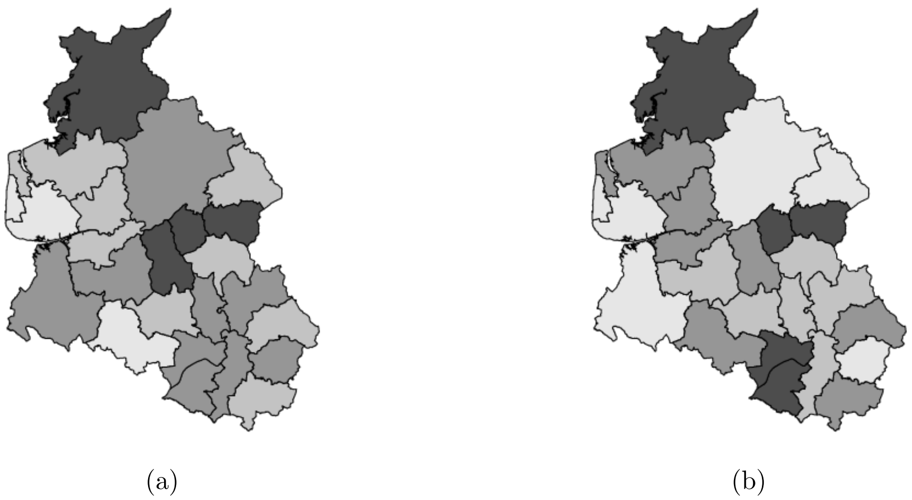


Figure 6.5: Posterior means of frailties for the neighborhood-based model from (a) spBayesSurv package and (b) `GeoJAGS` module.

Chapter 7

Conclusion and future work

The study presented in this document was motivated mainly by Wabersich and Vandekerckhove (2014), which can be seen as a short tutorial on how to implement a new JAGS module with univariate functions and distributions. The central topic explored in our research is spatial modeling and we have focused our attention on point-referenced and areal data. There is no spatial module in JAGS to build and handle the (sometimes complicated) covariance functions required in geostatistical models and the covariance matrix required in the CAR model. In fact, there is no spatial module in `JAGS` to work with any spatial data; including areal data and point-pattern data. The main goal of our study was to implement a `JAGS` module for a spatial analysis. Again, it is important to highlight the fact that the `BUGS` family of softwares (`WinBUGS`, `OpenBUGS`) and `JAGS` consist of attractive tools to work with the MCMC Gibbs Sampling widely used to fit a Bayesian model. The main advantage of them is the fact that only the likelihood and the prior distributions are necessary when defining a model script (no need to implement the posterior full conditionals). This represents a major step to attract researches interested in the Bayesian analysis, but not willing to learn or spend time with calculations and the full implementation of the MCMC algorithm. As mentioned above, the lack of a spatial module in `JAGS` makes this option less attractive than `OpenBUGS` and `WinBUGS`, since the two latter have the `GeoBUGS` module.

At this point of our research, we have implemented a `JAGS` module with twelve covariance functions to be used for a point-referenced data analysis. Some of these functions are complex and would demand a great effort to be implemented in a `JAGS` script; for example the Matérn class involving the modified Bessel function. In addition, the proposed module offers an approach to use the CAR distribution for areal data analysis, computing the precision matrix for the user. During the development period, we have realized that `JAGS` modules can be created only with functions, i.e. there is no need to add a routine to sample from certain distribution in order to build a valid module. This can be considered a contribution of our work since the tutorial in Wabersich and Vandekerckhove (2014) does not mention this aspect.

The covariance functions in the proposed module were tested using a simple geostatistical data set found in the literature (the elevation data). We have compared the estimates via `JAGS` with those from `likfit` function of the package `geoR`; note that `likfit` is based on the frequentist inference through maximum likelihood estimation. The idea here is that the frequentist results are well established in the literature, thus we have used them as a reference to check if the new module is working properly. As expected, all results from `JAGS` are close to the corresponding values from `likfit` (confirming the accuracy of the implementation). Furthermore, using the same elevation data set we presented an application involving predictions at arbitrary locations. The results obtained by `GeoJAGS` module were compared with those from `GeoBUGS` module and the frequentist approach using the `likfit` function, indicating a similar performance in terms of predictions by the two modules.

Our proposed CAR matrix function was tested using the gross domestic product data in the municipalities of Minas Gerais state in Brazil. We have compared the estimates from `JAGS` with those from `OpenBUGS`. Overall, in terms of effective sample size, the chains from `GeoJAGS` had a better performance than those from `GeoBUGS`. The random effects estimated by the two modules were considerably different in magnitude. Since it was a real scenario, the true values of the latent parameters are unknown. This fact motivated us to develop two simulation cases in order to compare the estimates provided by the two spatial modules. We came to a conclusion that although `GeoJAGS` takes a longer time to perform the computations than `GeoBUGS`, the former gave better estimates than the latter. The chains from the proposed module were less autocorrelated than those from

GeoBUGS in the case which the data variance is large than the spatial variance. When the spatial variance was set to be greater than the non-spatial variance, both modules shown a similar performance in terms of ESS.

Once the proposed module was validated, we illustrated the usage of GeoJAGS in two data sets from different research areas (climatology and medical science). Here, we have not proposed any new model, the main aim is to show how the GeoJAGS module can be used to simplify the task of modeling spatially correlated data. In addition, we showed that with the proposed module, the user is able to fit spatial data coming from other distribution than the Gaussian (as the Gamma, in Section 6.1.1).

Overall, our module arises as a compelling alternative for implementing a Bayesian hierarchical model. There is no module that provides a wide range of covariance functions for point-referenced data in neither BUGS or JAGS. In the comparison study presented in Section 4.5, the GeoJAGS module had a better performance than OpenBUGS in terms of execution time. A considerable contribution of our work is the implementation of the Matérn covariance structure, allowing the users to take advantage of the properties of this function in their Bayesian modeling. Regarding areal data, we provide a simple approach to introduce the CAR distribution in any JAGS script. In our implementation of the CAR model, we used the block sampling method (along with the multivariate normal distribution), which leads to less autocorrelated MCMC chains when compared to those sampled from the full conditional distributions. Thus, when it comes to effective sample size, our tests indicated that the proposed module produces posterior chains that are less autocorrelated than those from GeoBUGS.

7.1 Future work

At this point, we have finished the implementation of the point-referenced data tools and the CAR model for areal data. A straightforward extension of this module is the development of procedures for point-pattern data, such as modeling intensity functions for the inhomogeneous Poisson process. These functions assume that the intensity function varies spatially.

Another aspects that can be explored are the spatio-temporal models, which could supplement `GeoJAGS` module to be useful in a even wider range of applications. In this case, computational complexity will rise, since we must take into account the spatial correlation and the temporal correlation. These implementations would be useful for the meteorological data shown in Section 6.1.

Currently, we are working on some important improvements on the performance of the CAR distribution in our module. We plan to develop a new `JAGS` distribution for the proper CAR model. It will be based on the results from the *Eigenvalues of a scalar multiple of a matrix* theorem (see Theorem 3.1) which will drastically reduce the execution time of hierarchical models that include the CAR distribution from `GeoJAGS` (considering the results shown in Section 3.4.2). It is important to remark that performing these implementations consists of a challenging task since there is no tutorial on this regard, with the only resource being the `JAGS` source code and the contributions in Wabersich and Vandekerckhove (2014).

Appendices

Appendix A

Complete list of functions in `GeoJAGS`

In this appendix, we present the description of all functions currently available in the `GeoJAGS` module, their input arguments and their return values. The proposed module is available for download in `www.geojags.sourceforge.net`.

`CVCauchy(coords, sigma2, phi, tau2, kappa)`	
	Computes the Cauchy covariance matrix.
`coords`	a $n \times r$ matrix where each row has the r-dimensional coordinates of the n data locations ($r \geq 1$).
`sigma2`	the partial sill (spatial variance).
`phi`	the reciprocal of the range parameter.
`tau2`	the nugget effect (non-spatial variance).
`kappa`	the smoothness parameter.
Value	a $n \times n$ matrix with entries following the Cauchy covariance function.

`CVGenCauchy(coords, sigma2, phi, tau2, kappa1, kappa2):`	
	Computes the generalized Cauchy covariance matrix.
`coords`	a $n \times r$ matrix where each row has the r-dimensional coordinates of the n data locations ($r \geq 1$).
`sigma2`	the partial sill (spatial variance).
`phi`	the reciprocal of the range parameter.
`tau2`	the nugget effect (non-spatial variance).
`kappa1`	controls the dependence at large distances.
`kappa2`	the shape paratemer.
Value	a $n \times n$ matrix with entries following the generalised Cauchy covariance function.

`CVCircular(coords, sigma2, phi, tau2):`

	Computes the circular covariance matrix.
`coords`	a $n \times r$ matrix where each row has the r-dimensional coordinates of the n data locations ($r \geq 1$).
`sigma2`	the partial sill (spatial variance).
`phi`	the reciprocal of the range parameter.
`tau2`	the nugget effect (non-spatial variance).
Value	a $n \times n$ matrix with entries following the circular covariance function.

`CVCubic(coords, sigma2, phi, tau2):`

	Computes the cubic covariance matrix.
`coords`	a $n \times r$ matrix where each row has the r-dimensional coordinates of the n data locations ($r \geq 1$).
`sigma2`	the partial sill (spatial variance).
`phi`	the reciprocal of the range parameter.
`tau2`	the nugget effect (non-spatial variance).
Value	a $n \times n$ matrix with entries following the cubic covariance function.

`CVExp(coords, sigma2, phi, tau2):`

	Computes the exponential covariance matrix.
`coords`	a $n \times r$ matrix where each row has the r-dimensional coordinates of the n data locations ($r \geq 1$).
`sigma2`	the partial sill (spatial variance).
`phi`	the reciprocal of the range parameter.
`tau2`	the nugget effect (non-spatial variance).
Value	a $n \times n$ matrix with entries following the powered exponential covariance function.

`CVGaussian(coords, sigma2, phi, tau2):`

	Computes the Gaussian covariance matrix.
`coords`	a $n \times r$ matrix where each row has the r-dimensional coordinates of the n data locations ($r \geq 1$).
`sigma2`	the partial sill (spatial variance).
`phi`	the reciprocal of the range parameter.
`tau2`	the nugget effect (non-spatial variance).
Value	a $n \times n$ matrix with entries following the powered exponential covariance function.

`CVGneiting(coords, sigma2, tau2)`:

	Computes the Gneiting covariance matrix. The corresponding covariance function only depends on the distance between two points. Differently from `geoR`, in our implementation we added the possibility of introducing the nugget effect (τ^2). Thus in the diagonal, it depends on σ^2 and τ^2. For further details see documentation of the function `RMgneiting` in the R package `RandomFields` (Schlather et al., 2015). According to this reference: *It is an alternative to the Gaussian model since its graph is visually hardly distinguishable from the graph of the Gaussian model, but possesses neither the mathematical and nor the numerical disadvantages of the Gaussian model.*
`coords`	a $n \times r$ matrix where each row has the r-dimensional coordinates of the n data locations ($r \geq 1$).
`sigma2`	the partial sill (spatial variance).
`tau2`	the nugget effect (non-spatial variance).
Value	a $n \times n$ matrix with entries following the Gneiting covariance function.

`CVMatern(coords, sigma2, phi, tau2, nu)`:

	Computes the Matérn covariance matrix.
`coords`	a $n \times r$ matrix where each row has the r-dimensional coordinates of the n data locations ($r \geq 1$).
`sigma2`	the partial sill (spatial variance).
`phi`	the reciprocal of the range parameter.
`tau2`	the nugget effect (non-spatial variance).
`nu`	the shape parameter.
Value	a $n \times n$ matrix with entries following the Matérn covariance function.

`CVPowerExp(coords, sigma2, phi, tau2, p)`:

	Computes the powered exponential covariance matrix.
`coords`	a $n \times r$ matrix where each row has the r-dimensional coordinates of the n data locations ($r \geq 1$).
`sigma2`	the partial sill (spatial variance).
`phi`	the reciprocal of the range parameter.
`tau2`	the nugget effect (non-spatial variance).
`p`	the shape parameter. When $p = 1$ it gives the exponential function, whereas $p = 2$ gives the Gaussian covariance function
Value	a $n \times n$ matrix with entries following the powered exponential covariance function.

`CVRatQuad(coords, sigma2, phi, tau2)`:

	Computes the rational quadratic covariance matrix.
`coords`	a $n \times r$ matrix where each row has the r-dimensional coordinates of the n data locations ($r \geq 1$).
`sigma2`	the partial sill (spatial variance).
`phi`	the reciprocal of the range parameter.
`tau2`	the nugget effect (non-spatial variance).
Value	a $n \times n$ matrix with entries following the rational quadratic covariance function.

`CVSpherical(coords, sigma2, phi, tau2)`:

	Computes the spherical covariance matrix.
`coords`	a $n \times r$ matrix where each row has the r-dimensional coordinates of the n data locations ($r \geq 1$).
`sigma2`	the partial sill (spatial variance).
`phi`	the reciprocal of the range parameter.
`tau2`	the nugget effect (non-spatial variance).
Value	a $n \times n$ matrix with entries following the spherical covariance function.

`CVWave(coords, sigma2, phi, tau2)`:

	Computes the wave covariance matrix.
`coords`	a $n \times r$ matrix where each row has the r-dimensional coordinates of the n data locations ($r \geq 1$).
`sigma2`	the partial sill (spatial variance).
`phi`	the reciprocal of the range parameter.
`tau2`	the nugget effect (non-spatial variance).
Value	a $n \times n$ matrix with entries following the wave covariance function.

`precMatrixCAR(W, rho, tau2)`:

	Computes the precision matrix for the CAR distribution.
`W`	a n-dimensional squared matrix given the proximity structure.
`rho`	the parameter to solve the impropriety of the distribution .
`tau2`	the spatial precision (inverse of variance).
Value	a $n \times n$ matrix of the form $\frac{1}{\tau^2}(\boldsymbol{D}_w - \rho \boldsymbol{W})$.

`bessel_k(x, nu, expo)`:

	Calculates modified Bessel functions of the third kind, which is useful for users willing to implement their own version of the Matérn covariance function instead of using our proposed one.
`x`	argument for which K's or exponentially scaled K's (K*EXP(X)) are to be calculated.
`nu`	the order of the Bessel function.
`expo`	must be equals 1 if unscaled K's are to be calculated and equals 2 if exponentially scaled K's are to be calculated.
Value	an scalar with the value of the modified Bessel function.

Appendix B

Adding a multivariate distribution to JAGS

In this appendix we discuss the steps needed to build new multivariate distributions in JAGS using as an illustrative example of the process the proposed version of the multivariate normal distribution mentioned in Section 3.4.2. This implementation will take advantage of the result from Theorem 3.1 by using the eigen-decomposition of the matrix $\boldsymbol{D}_w - \rho\boldsymbol{W}$ to obtain a sample from the multivariate normal distribution with precision matrix $\frac{1}{\tau^2}(\boldsymbol{D}_w - \rho\boldsymbol{W})$.

As stated in Section 3.3, a new distribution for JAGS is implemented through a new C++ class. In the `src/` directory of our module, add a subdirectory `distributions/` and place in it the files of the new distribution (See Boxes B.1 and B.2). For the multivariate case, this new class must inherit from the `ArrayDist` class, which is useful whenever a distribution takes values in a matrix or a array or has parameters that are array-valued (the `ArrayDist.h` file is located at `/src/include/distributions` within the JAGS directory folder). Any class that inherits from `ArrayDist` must implement its virtual functions, namely:

- `logDensity:` a routine that returns the log probability density;
- `randomSample:` draws a random sample from the distribution;
- `typicalValue:` returns a typical value from the distribution (the meaning of this will depend on the distribution, but it will normally be a mean, median or mode);
- `checkParameterDim:` checks that dimensions of the parameters are correct;
- `checkParameterValue:` Checks that the values of the parameters are consistent with the distribution;

Box B.1: The DMNormCAR class header file `dmnormcar.h`

```
#ifndef DMNORMCAR_H_
#define DMNORMCAR_H_
#include <distribution/ArrayDist.h>
#include <armadillo>
using namespace jags;
namespace GeoJAGS {

class DMNORMCAR : public ArrayDist {
    public:
      DMNORMCAR();
       double logDensity(double const *x, unsigned int length, PDFType type,
                         std::vector<double const *> const &parameters,
                         std::vector<std::vector<unsigned int> > const &dims,
                         double const *lower, double const *upper)  const;
       void randomSample(double *x, unsigned int length,
                         std::vector<double const *> const &parameters,
                         std::vector<std::vector<unsigned int> > const &dims,
                         double const *lower, double const *upper, RNG *rng) const;
       void typicalValue(double *x, unsigned int length,
                         std::vector<double const *> const &parameters,
                         std::vector<std::vector<unsigned int> > const &dims,
                         double const *lower, double const *upper) const;
       bool checkParameterDim(std::vector<std::vector<unsigned int> > const &dims) const;
       bool checkParameterValue(std::vector<double const *> const &parameters,
                                std::vector<std::vector<unsigned int> > const &dims) const;
       std::vector<unsigned int>  dim(std::vector<std::vector<unsigned int> > const &dims) const;
       static void randomsample(double *x, double const *eigenValues, double const *eigenVectors,
                                double tau2, bool prec, int nrow, RNG *rng);
       void support(double *lower, double *upper, unsigned int length,
                         std::vector<double const *> const &parameters,
                         std::vector<std::vector<unsigned int> > const &dims) const;
       bool isSupportFixed(std::vector<bool> const &fixmask) const;
    };
}
#endif /* DMNORMCAR_H_ */
```

- `dim`: Calculates what the dimension of the distribution should be, based on the dimensions of its parameters;
- `support`: returns the support of an unbounded distribution;
- `isSupportFixed`: indicates whether the support of the distribution is fixed;

Box B.1 contains the file `/src/distributions/dmnormcar.h`, which includes the prototypes of the constructor and the required functions previously listed. The `randomsample` function takes seven input arguments:

- `x`: array to which the sample values are written;
- `length`: size of the array x;
- `parameters`: parameters for the distribution. Each element is a pointer to the start of an array containing the parameters. The size of the array should correspond to the dims parameter. In our case, the parameters will be the eigen-decomposition and the spatial variance effect;
- `dims`: dimensions of the parameters. It will contain the size of the vector of eigen-values and the dimension of the eigen-vectors matrix;

- `lower`: pointer to array containing the lower boundary of the distribution. This should be of size `length` or may be NULL if there is no lower boundary;
- `upper`: same as above, but considering the upper boundary of the distribution;
- `rng`: pseudo-random number generator to use.

Since some of the functions have similar arguments and considering the descriptions above, the arguments of the remaining functions have a straightforward interpretation. Further, a few comments on each argument can be found in the `ArrayDist.h` file. Box B.2 displays implementations of the essential functions in the class. The constructor is defined in line 1, it calls its parents' constructor function with two arguments. The first sets the name of the new distribution (`dmnormcar`) and the second is used to define the number of arguments (our proposed distribution has 3 arguments: the eigen-values, eigen-vectors and the spatial variance effect). In line 2 we have the implementation of the `logdensity` function. As the name suggests, it returns the log probability density. Note that here we have used the `armadillo` library to perform the required linear algebra computations (lines 8-12). Thus, this package must be installed in the computer ahead of compiling the code in Box B.2. The `randomsample` function in line 35 implements the core part of the new distribution. Our procedure here follows Algorithm 3.1, except for computing the eigen-decomposition, since it is part of the arguments of this implementation.

After the distribution class is ready, we need to add it to the `GeoJAGS.cc` module by loading it with the constructor and deleting it with the destructor, as in Box 3.2. See Box B.3 for the lines to add. In addition, an appropriate `Makefile.am` must be added to the newly created `src/distributions/` folder. Details about this file are found in Wabersich and Vandekerckhove (2014, step 3.3, p. 19). Ultimately, steps for building and installing the module are the same as described at the end of Section 3.3.

The steps described above define a new multivariate distribution to JAGS. However when attempting to sample from this distribution on a hierarchical Bayesian model, JAGS is unable to find an appropriate sampler for this distribution. Our investigations so far led us to create a new JAGS sampler for this newly implemented distribution, based on the complex implementation of the sampler for the multivariate normal distribution which uses the Metropolis algorithm. Since there is no instructions available for developing a new sampler structure for JAGS, our implementation still needs some adjustment in order to make it work properly. Once this sampler is ready to use, we believe (based on the experiments about execution time shown in Section 3.4.2) in the decreasing of the time taken to sample from our proposed version of the multivariate normal distribution and therefore, the CAR distribution will be more efficient in terms of execution time.

Box B.2: Some functions in the DMNormCAR `dmnormcar.cc` file

```
DMNORMCAR::DMNORMCAR() : ArrayDist("dmnormcar", 3) {}
double DMNORMCAR::logDensity(double const *x, unsigned int m, PDFType type,
         vector<double const *> const &parameters, vector<vector<unsigned int> > const &dims,
         double const *lower, double const *upper) const {
  double const * eigenValues = parameters[0];
  double const * eigenVectors = parameters[1];
  double const tau2 = parameters[2][0];
  vec diag = vec(m);
  for (int i=0; i<m; i++){ diag[i] = eigenValues[i]; }
  mat P(eigenVectors, m, m);
  mat D = tau2 * arma::diagmat(diag);
  mat Pinv = inv(P);
  mat T = P * D * Pinv;
  double loglik = 0;
  vector<double> delta(m);
  for (unsigned int i = 0; i < m; ++i) {
      delta[i] = x[i];
      loglik -= (delta[i] * T(i, i) * delta[i])/2;
      for (unsigned int j = 0; j < i; ++j) {
          loglik -= (delta[i] * T(i, j) * delta[j]);
     }
  }
  switch(type) {
     case PDF_PRIOR:
          break;
     case PDF_LIKELIHOOD:
          loglik += log(arma::det(T))/2;
          break;
     case PDF_FULL:
          loglik += log(arma::det(T))/2 - m * M_LN_SQRT_2PI;
          break;
   }
    return loglik;
}
void DMNORMCAR::randomSample(double *x, unsigned int m, vector<double const *> const &parameters,
         vector<vector<unsigned int> > const &dims, double const *lower, double const *upper,
         RNG *rng) const {
    double const * eigenValues = parameters[0]; double const * eigenVectors = parameters[1];
    double const tau2 = parameters[2][0];
    randomsample(x, eigenValues, eigenVectors, tau2, true, m, rng);
}
void DMNORMCAR::randomsample(double *x, double const *eigenValues, double const *eigenVectors,
                             double tau2, bool prec, int nrow, RNG *rng) {
    double * w = new double[nrow]; int N = nrow*nrow;
  /* Generate independent random normal variates, scaled by the eigen values */
  if (prec)
    for (int i = 0; i < nrow; ++i)
        w[i] = rnorm(0, 1/sqrt(eigenValues[i]), rng);
  else
    for (int i = 0; i < nrow; ++i)
          w[i] = rnorm(0, sqrt(eigenValues[i]), rng);
  for (int i = 0; i < nrow; ++i) { /* Now transform them to dependant variates */
    x[i] = 0;
    for (int j = 0; j < nrow; ++j) {
      x[i] += tau2 * eigenVectors[i + j * nrow] * eigenValues[j];
    }
  } delete [] w;
}
```

We are currently working in two different fronts. The first aims to develop a new sampler that can generate values following the proposed implementation of the proper CAR model. In the second, we attempt to create a version of the CAR distribution that samples from the posterior full conditionals $p(\theta_i \mid \theta_j, j \neq i)$.

Box B.3: The code to add to the module class file (`src/GeoJAGS.cc`).

```
// add this to the constructor function
// to instantiate a distribution object when the module is loaded
    insert(new DMNORMCAR);

// add this to the destructor function
// to remove the instantiated distribution object when the module is unloaded
    std::vector<Distribution*> const &dvec = distributions();
    for (unsigned int i=0; i<dvec.size(); ++i) {
        delete dvec[i]; // delete all instantiated distributions objects
    }
```

Appendix C

Additional results for the validation study of covariance functions

In this appendix, we present the results for the remaining covariance functions not explored in Chapter 4. Consider the same MCMC configuration and the same prior distributions indicated for the Matérn covariance analysis. Again, the study here compares the Bayesian (JAGS with the `GeoJAGS` module) and the frequentist (`likfit` in `geoR`) estimates (model selection is not developed). Running each covariance structure is quite simple. Revise Box 4.3 (page 52) to see the general structure of the code used within R to run the JAGS model; the only change is applied to line 8, replacing the `model.file` argument, which should be the name of the JAGS model file to be executed. Similarly to the model in Box 4.4, we only need to replace the function in line 9 by the corresponding covariance function, with one of the twelve currently available options in `GeoJAGS`.

Powered exponential

Parameters	Maximum likelihood estimates	JAGS using `CVPoweredExp` function	HPD interval (95%)	
ϕ	0.044	0.194	0.007	0.583
τ^2	0.000	11.485	0.000	68.205
σ^2	3,409.387	2,364.217	973.995	4,379.111
β_0	926.985	921.640	829.942	1,000.609
β_1	-5.037	-4.719	-15.597	5.488
β_2	-17.196	-17.829	-28.679	-7.996

Table C.1: Bayesian and frequentist estimates assuming the Gaussian model described in Section 4.3.1, considering the powered exponential covariance. Let $p = 0.5$.

Exponential

Parameters	Maximum likelihood estimates	JAGS using `CVPoweredExp` function	HPD interval (95%)	
ϕ	0.402	0.362	0.092	0.684
τ^2	0.000	5.452	0.000	25.158
σ^2	1,731.799	2,268.663	893.538	4,166.565
β_0	919.103	919.3173	841.646	1,019.822
β_1	-5.583	-5.475	-20.864	7.288
β_2	-15.515	-14.983	-29.603	-1.568

Table C.2: Bayesian and frequentist estimates assuming the Gaussian model described in Section 4.3.1, considering the powered exponential covariance. Let $p = 1$, which determines the Exponential covariance.

Gaussian

Parameters	Maximum likelihood estimates	JAGS using `CVPoweredExp` function	HPD interval (95%)	
ϕ	0.725	0.703	0.470	0.941
τ^2	86.189	125.221	20.881	291.279
σ^2	1,450.216	1,588.140	715.325	2,667.433
β_0	913.266	910.206	852.720	961.228
β_1	-4.812	-4.389	-14.254	6.193
β_2	-18.271	-18.082	-28.994	-7.133

Table C.3: Bayesian and frequentist estimates assuming the Gaussian model described in Section 4.3.1, considering the Gaussian covariance (powered exponential with $p = 2$).

Rational quadratic

Parameters	Maximum likelihood estimates	JAGS using `CVRatQuad` function	HPD interval (95%)	
ϕ	-	2.459	0.659	5.043
τ^2	-	78.169	0.006	229.777
σ^2	-	2.469	901.594	3,289.230
β_0	-	912.616	843.996	984.954
β_1	-	-4.839	-17.019	8.968
β_2	-	-16.297	-27.992	-2.868

Table C.4: Bayesian estimates assuming the Gaussian model described in Section 4.3.1, considering the Rational Quadratic covariance function. The frequentist results are not presented since `likfit` (`geoR`) does not have this case.

Wave

Parameters	Maximum likelihood estimates	JAGS using `CVWave` function	HPD interval (95%)	
ϕ	-	1.423	0.744	2.252
τ^2	-	361.668	140.491	639.567
σ^2	-	1,519.512	367.957	3,012.710
β_0	-	912.906	860.505	959.730
β_1	-	-4.703	-15.897	4.621
β_2	-	-18.618	-29.484	-5.798

Table C.5: Bayesian estimates assuming the Gaussian model described in Section 4.3.1, considering the Wave covariance. `likfit` fails to fit this case, even when choosing different initial values for ϕ and σ^2.

Cauchy

Parameters	Maximum likelihood estimates	JAGS using `CVCauchy` function	HPD interval (95%)	
ϕ	1.239	1.045	0.520	1.607
τ^2	57.423	45.110	0.000	159.172
σ^2	2,640.767	2,333.670	885.927	4,318.037
β_0	914.656	914.875	831.632	992.189
β_1	-5.553	-5.112	-18.209	-9.267
β_2	-14.975	-16.146	-29.174	-1.597

Table C.6: Bayesian and frequentist estimates assuming the Gaussian model described in Section 4.3.1, considering the Cauchy covariance.

Generalized Cauchy

Parameters	Maximum likelihood estimates	JAGS using `CVGenCauchy` function	HPD interval (95%)	
ϕ	-	0.991	0.445	1.630
τ^2	-	4.480	0.000	22.584
σ^2	-	3,213.359	1,374.884	5,239.042
β_0	-	919.363	808.821	1,012.235
β_1	-	-5.320	-18.721	8.885
β_2	-	-15.121	-30.162	3.560

Table C.7: Bayesian estimates assuming the Gaussian model described in Section 4.3.1, considering the Generalized Cauchy covariance. The frequentist estimates are not presented since this option is not implemented in `likfit` (`geoR`).

Gneiting

Parameters	Maximum likelihood estimates	JAGS using `CVGneiting` function	HPD interval (95%)	
ϕ	1.337	-	-	-
τ^2	79.828	1,308.138	826.034	1,908.299
σ^2	1,422.834	37.971	0.000	300.235
β_0	913.333	912.917	888.223	941.178
β_1	-4.721	-1.654	-7.388	3.675
β_2	-18.476	-25.052	-30.133	-20.342

Table C.8: Bayesian and frequentist estimates assuming the Gaussian model described in Section 4.3.1, considering the Gneiting covariance.

Note that the maximum likelihood estimates differ significantly from the posterior means of σ^2, ϕ and τ^2. These parameters were implemented in `CVGneiting` but they are not specified in the `geoR` documentation. Even though their outputs are obtained through `likfit`.

The definition of the next covariance functions (spherical, circular and cubic; presented in Section 2.2.1) shows that the intervals where they are defined depend on the parameter ϕ. As explained in Section 4.3, the user is supposed to choose an appropriate value for ϕ (fixed, it will not be estimated). This is the reason why the next three tables do not show estimates for ϕ for the Bayesian estimates.

Spherical

Parameters	Maximum likelihood estimates	JAGS using `CVSpherical` function	HPD interval (95%)	
ϕ	0.395	-	-	-
τ^2	0.00	61.385	0.004	178.114
σ^2	1,070.964	1,283.353	627.515	1,934.576
β_0	914.771	925.110	878.884	973.210
β_1	-3.926	-5.701	-14.429	4.138
β_2	-19.817	-21.563	-31.470	-13.293

Table C.9: Bayesian and frequentist estimates assuming the Gaussian model described in Section 4.3.1, considering the Spherical covariance.

Circular

Parameters	Maximum likelihood estimates	JAGS using `CVCircular` function	HPD interval (95%)	
ϕ	2.223	-	-	-
τ^2	0.000	417.512	0.001	1,445.901
σ^2	1,091.457	891.067	0.000	1,594.459
β_0	915.571	912.822	886.032	938.140
β_1	-4.172	-1.764	-6.884	3.669
β_2	-19.984	-24.585	-29.306	-19.085

Table C.10: Bayesian and frequentist estimates assuming the Gaussian model described in Section 4.3.1, considering the Circular covariance.

Cubic

Parameters	Maximum likelihood estimates	JAGS using `CVCubic` function	HPD interval (95%)	
ϕ	3.034	-	-	-
τ^2	55.027	27.597	0.000	224.768
σ^2	1,381.935	1,274.467	796.429	1,800.511
β_0	914.082	912.042	886.096	936.941
β_1	-4.695	-1.526	-7.441	3.162
β_2	-18.912	-24.487	-28.974	-19.490

Table C.11: Bayesian and frequentist estimates assuming the Gaussian model described in Section 4.3.1, considering the Cubic covariance.

Appendix D

Additional figures

In this appendix we present supplementary figures that convey extra information regarding some topics discussed in the main text.

Posterior chains of some parameters in the GDP data analysis

Consider the data set on gross domestic product (GDP) per capita in the municipalities of Minas Gerais presented in Section 5.1. For each parameter shown in Figure 5.2,

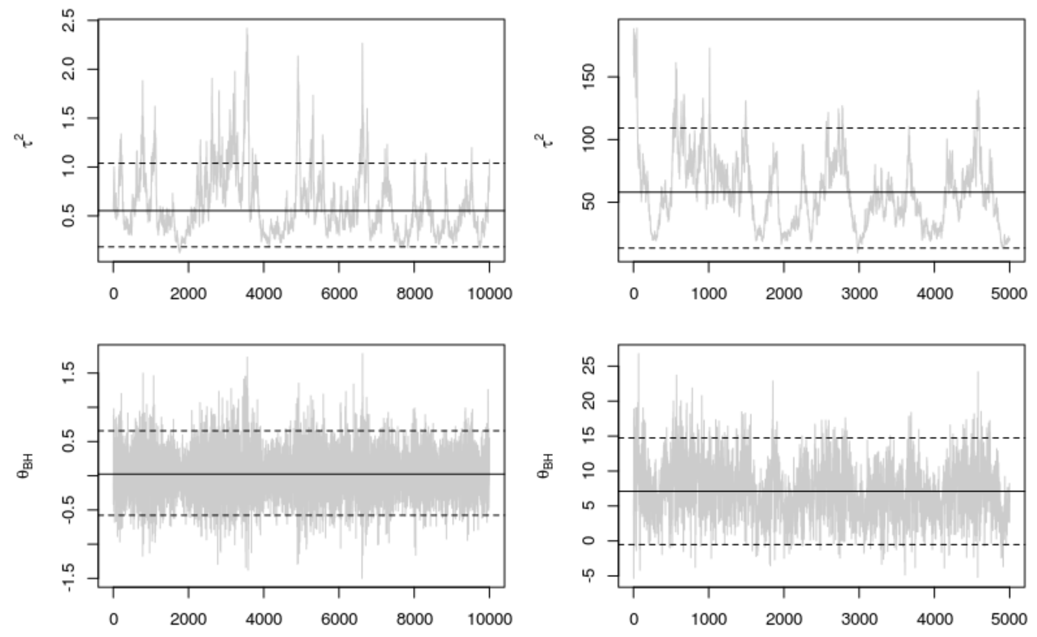


Figure D.1: Posterior sample, 95% HPD limits (dashed lines) and the posterior mean (solid horizontal line) of τ^2 and the spatial random effect related to Belo Horizonte in the GDP per capita study in Minas Gerais. Estimates in the left column are from `GeoJAGS` and those in the right column are from `GeoBUGS`.

the vertical axis are fixed, thus the magnitude of the estimates are in fact comparable. However, the range of values taken by two parameters in the JAGS model (τ^2 and θ_{BH}) are too narrow. Thus, making it impossible to note the behavior of these chains and compare with the corresponding chains from GeoBUGS. In Figure D.1 we present a detailed version of the chains for τ^2 and θ_{BH} from both GeoJAGS and GeoBUGS modules. Here, the y-axis ranges from the maximum and minimum values obtained in each chain of size 5,000. Note that the two spatial models generate high autocorrelated chains for the spatial variance parameter τ^2. The chains from GeoJAGS present a better visual behaviour of convergence than those from GeoBUGS.

Posterior chains of some parameters in the simulation analysis (case 1)

Here we consider the simulation study discussed in in Section 5.3.1, in which we set $\sigma^2 \gg \tau^2$. Figure D.2 presents the chains for μ, σ^2, τ^2 and for one of the spatial

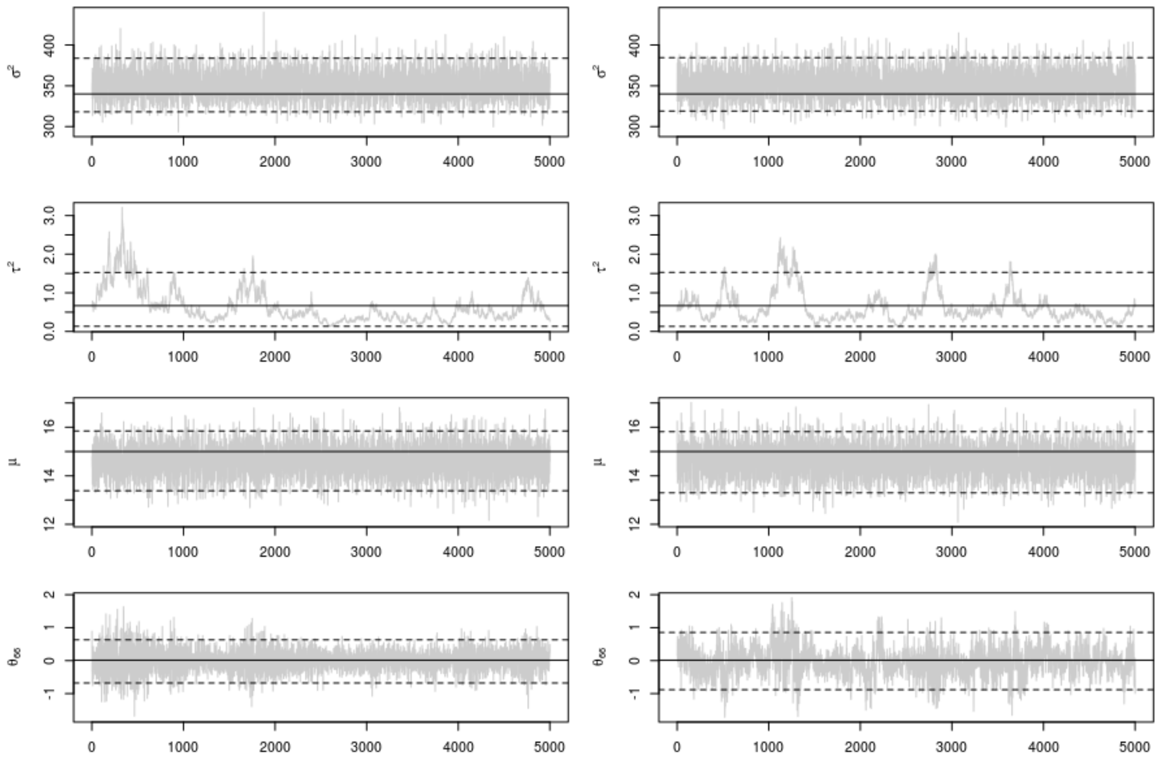


Figure D.2: Posterior sample, 95% HPD limits (dashed lines) and the real values (solid horizontal lines) of 4 parameters of the simulated data considering $\sigma^2 \gg \tau^2$. Estimates in the left column are from GeoJAGS and from the right column are from GeoBUGS.

random effects, θ_{66}. The chains in the left column are those obtained from `GeoJAGS`, whereas those in the right are from `GeoBUGS`. The true (generated) value are shown in the solid horizontal lines and the 95% HPD intervals are represented as the dashed lines. For each parameter, the HPD interval contain the true value regardless of the spatial module used. Note that the `GeoBUGS` produces chains with hight autocorrelation than `GeoJAGS`.

Pluviometric atlas of Brazil

In this section we present the data from the Geological Service of Brazil (see CPRM, 2017) which serves as a basis for comparison among the predictions made by our proposed model and the official government data presented in Section 6.1. It presents the monthly mean isohyets (each isohyet is a line on the map connecting places that presents equal rainfall amount) considering precipitation measurements taken between the years 1977 and 2006. In the maps shown in Figure D.3, we present only the average precipitation of the driest and the rainy month of the year, September and March respectively. These are the months considered in the models discussed in Section 6.1. Although these maps show data for the entire Brazilian territory, we will only consider the pattern regarding the northern region of the country.

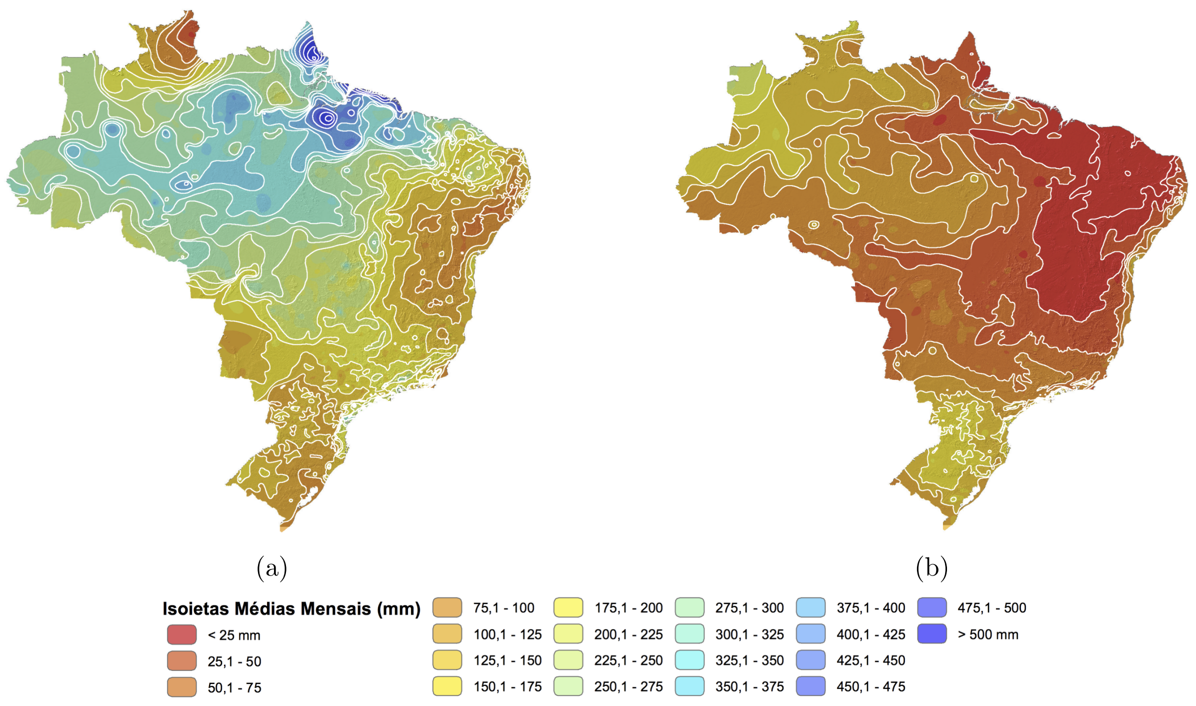


Figure D.3: Isohyets maps of monthly average precipitation (in millimeters) in Brazil for (a) March and (b) September between the years of 1977 and 2006.

References

Abramowitz, M. and Stegun, I. A. (1965), *Handbook of Mathematical Functions with Formulas, Graphs and Mathematical Tables*, Dover Publications, Inc., New York.

Addis, H., Klik, A., and Strohmeier, S. (2015), "Spatial variability of selected soil attributes under agricultural land use system in a mountainous watershed, Ethiopia," *International Journal of Geosciences*, 6, 606–613.

Akima, H. and Gebhardt, A. (2016), *akima: Interpolation of Irregularly and Regularly Spaced Data*, R package version 0.6-2.

Amazon Waters (2018), "Flows and Floods," `amazonwaters.org/waters/flows-and-floods/`, Accessed: 2018-01-02.

Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., and Sorensen, D. (1999), *LAPACK Users' Guide*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 3rd edn.

Aurenhammer, F. (1991), "Voronoi diagrams—a survey of a fundamental geometric data structure," *ACM Computing Surveys (CSUR)*, 23, 345–405.

Banerjee, S., Carlin, B. P., and Gelfand, A. E. (2004), *Hierarchical modeling and analysis for spatial data*, Chapman & Hall/CRC, Boca Raton.

Beezer, R. A. (2008), *A first course in linear algebra*, Beezer, Tacoma.

Besag, J. (1974), "Spatial Interaction and the Statistical Analysis of Lattice Systems," *Journal of the Royal Statistical Society. Series B*, 36, 192–236.

Bivand, R. S., Pebesma, E. J., Gomez-Rubio, V., and Pebesma, E. J. (2008), *Applied spatial data analysis with R*, Springer, New York.

Chiles, J.-P. and Delfiner, P. (1999), *Geostatistics: modeling spatial uncertainty*, Wiley, New York.

Cooley, D. (2017), *googleway: Accesses Google Maps APIs to Retrieve Data and Plot Maps*, R package version 2.2.0.

Coppersmith, D. and Winograd, S. (1987), "Matrix multiplication via arithmetic progressions," in *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pp. 1–6, ACM.

CPRM (2017), "Pluviometric Atlas of Brazil," `www.cprm.gov.br/publique/Hidrologia/Mapas-e-Publicacoes/Atlas-Pluviometrico-do-Brasil-1351.html`, Accessed: 2017-12-26.

Cressie, N. A. C. (1993), *Statistics for spatial data*, John Wiley & Sons, New York.

de Melo, S. N., Matias, L. F., and Andresen, M. A. (2015), "Crime concentrations and similarities in spatial crime patterns in a Brazilian context," *Applied Geography*, 62, 314–324.

Diggle, P. (2003), *Statistical Analysis of Spatial Point Patterns*, Mathematics in biology, Arnold, London.

Diggle, P. and Ribeiro, P. (2007), *Model-based Geostatistics*, Springer Series in Statistics, Springer, New York.

Gelfand, A. E. and Smith, A. F. M. (1990), "Sampling-Based Approaches to Calculating Marginal Densities," *Journal of the American Statistical Association*, 85, 398–409.

Geman, S. and Geman, D. (1984), "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6, 721–741.

Gilks, W. R. (1992), *Bayesian Statistics 4*, chap. Derivative-free adaptive rejection sampling for Gibbs sampling, pp. 641–649, Oxford University Press, Oxford.

Gilks, W. R. and Wild, P. (1992), "Adaptive rejection sampling for Gibbs sampling," *Applied Statistics*, 41, 337–348.

Gomes, V., Pitombo, C., Rocha, S., and Salgueiro, A. (2016), "Kriging Geostatistical Methods for Travel Mode Choice: A Spatial Data Analysis to Travel Demand Forecasting," *Open Journal of Statistics*, 6, 514–527.

Hastings, W. K. (1970), "Monte Carlo methods using Markov chains and their applications," *Biometrika*, 57, 97–109.

Henderson, R., Shimakura, S., and Gorst, D. (2002), "Modeling spatial variation in leukemia survival data," *Journal of the American Statistical Association*, 97, 965–972.

IBGE (2018), "Cidades e estados do Brasil," `cidades.ibge.gov.br/`, Accessed: 2018-01-02.

INMET (2017), "BDMEP - Banco de Dados Meteorológicos para Ensino e Pesquisa," `www.inmet.gov.br/portal/index.php?r=bdmep/bdmep`, Accessed: 2017-09-30.

Instituto Pristino (2013), "Relação de bens protegidos em Minas Gerais apresentados ao ICMS Patrimônio Cultural," `www.institutopristino.org.br/atlas/municipios-de-minas-gerais/baixe-os-arquivos-shp-e-kml/`.

Lawless, J. F. (2011), *Statistical models and methods for lifetime data*, vol. 362, John Wiley & Sons, Hoboken, New Jersey.

Lunn, D., Spiegelhalter, D., Thomas, A., and Best, N. (2009), "The BUGS project: evolution, critique and future directions," *Statistics in medicine*, 28, 3049–3067.

Mayrink, V. D. and Gamerman, D. (2009), "On computational aspects of Bayesian spatial models: influence of the neighboring structure in the efficiency of MCMC algorithms," *Computational Statistics*, 24, 641–669.

Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., and Teller, E. (1953), "Equation of state calculations by fast computing machines," *Journal of Chemistry and Phisics*, 21, 1087–1091.

Mohebbi, M., Wolfe, R., and Jolley, D. (2011), "A poisson regression approach for modelling spatial autocorrelation between geographically referenced observations," *BMC Medical Research Methodology*, 11, 133.

Neal, R. M. (2003), "Slice sampling," *Annals of statistics*, 31(3), 705–741.

NOAA (2018), "Historical El Nino / La Nina episodes (1950-present)," `http://origin.cpc.ncep.noaa.gov/products/analysis_monitoring/ensostuff/ONI_v5.php`, Accessed: 2018-01-28.

Pebesma, E. J. and Bivand, R. S. (2005), "Classes and methods for spatial data in R," *R News*, 5, 9–13.

Plummer, M. (2003), "JAGS: A program for analysis of Bayesian graphical models using Gibbs sampling," mcmc-jags.sourceforge.net/.

Plummer, M. (2016), *rjags: Bayesian Graphical Models using MCMC*, R package version 4-6.

Plummer, M., Best, N., Cowles, K., and Vines, K. (2006), "CODA: Convergence Diagnosis and Output Analysis for MCMC," *R News*, 6, 7–11.

R Core Team (2017), *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria.

Ribeiro Jr, P. J. and Diggle, P. J. (2016), *geoR: Analysis of Geostatistical Data*, R package version 1.7-5.2.

Sanderson, C. and Curtin, R. (2016), "Armadillo: a template-based C++ library for linear algebra," *Journal of Open Source Software*.

Schlather, M., Malinowski, A., Menck, P. J., Oesting, M., Strokorb, K., et al. (2015), "Analysis, simulation and prediction of multivariate random fields with package RandomFields," *Journal of Statistical Software*, 63, 1–25.

Shen, P. and Gong, W. (2015), "The Analysis on Evolvement of Spatial Pattern of Economy at County Level in Guangdong Province," *Open Journal of Social Sciences*, 3, 295–308.

Silva e Silva, L. G. (2014), *mapsBR: Mapas políticos do Brasil.*, R package version 1.0.

Sturtz, S., Ligges, U., and Gelman, A. (2005), "R2WinBUGS: A Package for Running WinBUGS from R," *Journal of Statistical Software*, 12, 1–16.

Su, Y.-S. and Yajima, M. (2015), *R2jags: Using R to Run 'JAGS'*, R package version 0.5-7.

Sun, Y., Bowman, K. P., Genton, M. G., and Tokay, A. (2015), "A Matérn model of the spatial covariance structure of point rain rates," *Stochastic Environmental Research and Risk Assessment*, 29, 411–416.

Tobler, W. R. (1970), "A computer movie simulating urban growth in the Detroit region," *Economic geography*, 46, 234–240.

Turner, R. (2017), *deldir: Delaunay Triangulation and Dirichlet (Voronoi) Tessellation*, R package version 0.1-14.

Wabersich, D. and Vandekerckhove, J. (2014), "Extending JAGS: A tutorial on adding custom distributions to JAGS (with a diffusion model example)," *Behavior Research Methods*, 46, 15–28.

Wickham, H. (2016), *rvest: Easily Harvest (Scrape) Web Pages*, R package version 0.3.2.

Wickham, H., Francois, R., Henry, L., and Müller, K. (2017), *dplyr: A Grammar of Data Manipulation*, R package version 0.7.4.

World Wild Life (2017), "WWF Places: Amazon," `www.worldwildlife.org/places/amazon#`, Accessed: 2017-12-24.

Yaglom, A. M. (1987), *Correlation theory of stationary and related random functions*, vol. I, Springer, New York.

Yan, J. and Prates, M. (2013), *rbugs: Fusing R and OpenBugs and Beyond*, R package version 0.5-9.

Zhou, H. and Hanson, T. (2017), "A unified framework for fitting Bayesian semiparametric models to arbitrarily censored survival data, including spatially-referenced data," *Journal of the American Statistical Association*.

Zhou, H., Hanson, T., and Zhang, J. (2017), "spBayesSurv: Fitting Bayesian Spatial Survival Models Using R," *arXiv preprint arXiv:1705.04584*.