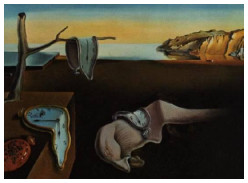

Journal of Statistical Software

MMMMMM YYYY, Volume VV, Code Snippet II. http://www.jstatsoft.org/

Pitfalls in the implementation of Bayesian hierarchical modeling of areal count data. An illustration using BYM and Leroux models.

Florian Gerber
University of Zurich

Reinhard Furrer
University of Zurich

Abstract

Several different hierarchical Bayesian models can be used for the estimation of spatial risk patterns based on spatially aggregated count data. Typically, the resulting posterior distributions of the model parameters cannot be expressed in closed forms, and MCMC approaches are required for inference. However, implementations of hierarchical Bayesian models for such areal data are error-prone. Also, different implementation methods exist, and a surprisingly large variability may develop between the methods as well as between the different MCMC runs of one method. This paper has four main goals: (1) to present a point by point annotated code of two commonly used models for areal count data, namely the BYM and the Leroux models (2) to discuss technical variations in the implementation of a formula-driven sampler and to assess the variability in the posterior results from various alternative implementations (3) to give graphical tools to compare sample(r)s which complement existing convergence diagnostics and (4) to give various practical tips for implementing samplers.

Keywords: MCMC, GMRF, R, openBUGS, geoBUGS, **spam**, **INLA**, **CARBayes**.

1. Introduction

Maps of spatially aggregated count data are often noisy, making interpretation difficult. To overcome this problem, Bayesian hierarchical models (BHMs) are frequently used to identify a smooth pattern that may be explained using underlying covariates and spatial factors.

Depending on the precise problem, different types of BHMs may be adequate. A Poisson likelihood (data layer) is commonly used for count data. The second layer, often called the process layer, links the log risk to spatially structured and unstructured components as well as potential covariates. The remaining layer(s) contain(s) the priors. The so-called Besag–York–Mollié (BYM) model (Besag *et al.* 1991; Mollié 1996) is extensively used in the case of the areal count data of rare diseases. For an overview and comparison of the BYM and other models see Waller and Carlin (2010) and Lee (2011). Yet another alternative model is the so-called Leroux model

(Leroux *et al.* 1999). For this and more alternatives also see LeSage and Pace (2009).

There is no such thing as a free lunch: ease of interpretation has to be paid by complexity of implementation. Therefore, inference in such models requires carefully adapted Markov chain Monte Carlo (MCMC) approaches or elaborated integrated nested Laplace approximation (INLA) techniques. For MCMC simulations, the sampler can be built "by hand" or in a software environment, e.g., BUGS and its derivatives, may be used. Tailored samplers are often used in an academic exercise framework or for more complicated settings. In those cases, the specifications of the "full conditional" densities, and the acceptance probabilities need to be derived.

Due to model complexity and the multitude of tuning possibilities of the approaches, both solutions are error-prone. Further, many models are robust in the sense that that (slightly) incorrect implementations may still lead to reasonable estimates such that incorrect conclusions are difficult to avoid. This recently happened in the BYM model example in Furrer and Sain (2010), where the calculation of the acceptance probability of the sampler, and, thus the results were incorrect. Similarly, the MCMC sampler of the R package **INLA** (Rue *et al.* 2009b) exhibited several issues that had to be addressed by the maintainers.

This paper has four main goals: (1) to present a point by point annotated code of the BYM model implementation (in openBUGS, R and **INLA**) and the Leroux model (in R and **CARBayes**) (2) to discuss technical variations in the implementation of a formula-given sampler and assess the variability in the posterior results from the various implementations, (3) to give graphical tools to compare samplers or even to detect incorrect samplers which complement existing convergence diagnostics and (4) to give various practical tips for implementing samplers. As an aside, the paper should reassure novices that even though the implementation of a BHM is theoretically straightforward, the road may be steep or bumpy.

This paper is not about (1) comparing or ranking individual packages or software environments, (2) or about quantitative and formal testing procedures to compare random samples.

We have opted to illustrate the approaches with a "classical" dataset to avoid any unnecessary complications. More specifically, we choose to use the oral cavity cancer data, which is available in the R packages **spam** and **INLA**. The dataset consists of death counts y_i caused by oral cavity cancer for a 5-year period (1986–1990) in the $i = 1, \ldots, 544$ districts (*Landkreise und kreisfreie Städte*) of Germany (Knorr-Held and Best 2001; Rue and Held 2005). The expected number of cases e_i was derived using demographical data that allows us to display the standardized mortality ratios y_i/e_i (Figure 1, left map). Finally, the dataset comes with a matrix $\mathbf{A}$ that defines the neighborhood stucture of the districts. We use a notation similar to Rue and Held (2005) such that, for example, bold lower and upper case letters are used for vectors and matrices, respectively.

Sections 2 and 3 discuss the BYM and Leroux models by first introducing the model and then discussing the different software implementations. The sections are concluded by comparing the MCMC sample(r)s and by presenting further remarks. Finally, Section 4 points to software options for some extensions of the models and gives some additional hints for assessing MCMC sample(r)s. The source code to reproduce the MCMC chains, Figures and Tables is provided as supplementary material.

2. Besag–York–Mollié model

In this section we give a short introduction to the BYM model, followed by implementations in openBUGS, R and **INLA**. These sections may be read independently of each other. The Sections 2.4 and 2.5 summarize the results and point to some extensions.

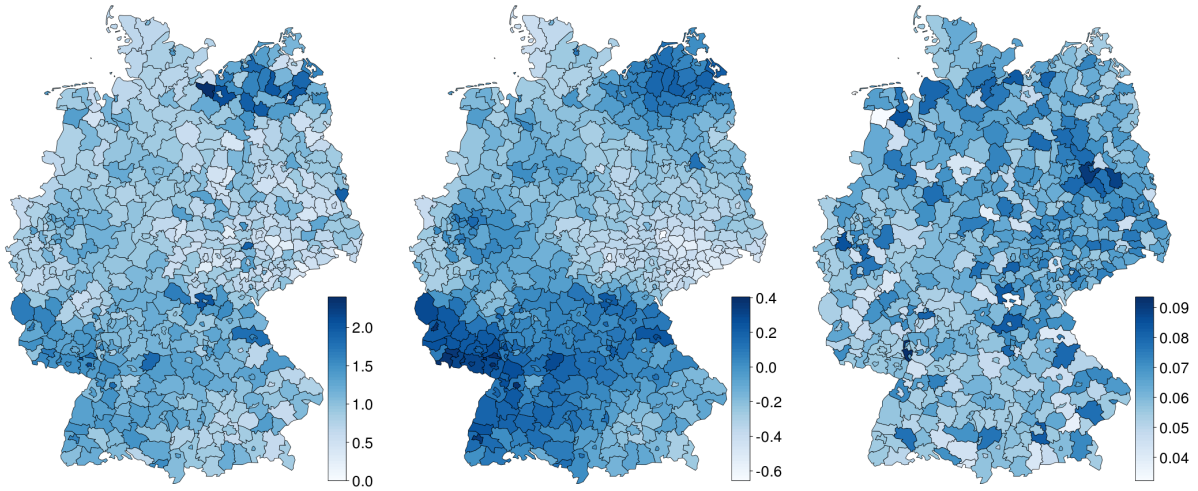

Figure 1: Standardized mortality ratios of oral cavity cancer deaths observed between 1986–1990 in Germany (left); the posterior means of the estimated relative log-risk of the BYM model (middle) and the difference between the posterior means of the log-risk of the Leroux and the BYM model (right).

To explore the spatial distribution of the relative risk, the data y_i is assumed to be conditionally independent Poisson counts with rate $e_i \exp(\eta_i)$, yielding the likelihood

$$\pi(y_i \mid \eta_i) \propto \prod_{i=1}^{n} \exp\big(y_i\eta_i - e_i \exp(\eta_i)\big) = \exp\big(\mathbf{y}^\top \boldsymbol{\eta} - \mathbf{e}^\top \exp(\boldsymbol{\eta})\big). \tag{1}$$

The log-relative risk is modeled by $\boldsymbol{\eta} = \mathbf{u} + \mathbf{v}$, where $\mathbf{v}$ is a zero mean white noise with precision κ_v, and $\mathbf{u}$ is a spatially structured component with precision κ_u. More precisely, $\mathbf{u}$ is a first order intrinsic Gaussian Markov random field (GMRF) (Rue and Held 2005, Chaper 3)

$$\pi(\mathbf{u} \mid \kappa_u) \propto \kappa_u^{\frac{n-1}{2}} \exp\Big(-\frac{\kappa_u}{2}\sum_{i\sim j}(u_i - u_j)^2\Big) = \kappa_u^{\frac{n-1}{2}} \exp\Big(-\frac{\kappa_u}{2}\mathbf{u}^\top\mathbf{R}\mathbf{u}\Big),$$

where $i \sim j$ denotes the set of all unordered pairs of neighbors, i.e., regions sharing a common border, and hence the sum over all such sets can be written using a sparse "structure" matrix $\mathbf{R}$. The resulting model is termed the *Besag–York–Mollié* model, see Besag *et al.* (1991). For inference on $\{\mathbf{u}, \mathbf{v}, \kappa_u, \kappa_v\}$ we set independent gamma priors for the precision parameters κ_u and κ_v, e.g., $\pi(\kappa_u \mid \alpha_u, \beta_u) \propto \kappa_u^{\alpha_u - 1} \exp(-\kappa_u\beta_u)$. We choose $\alpha_u = \alpha_v = 1$, $\beta_u = 0.5$ and $\beta_v = 0.01$ (and skip a carefully conducted sensitivity analysis, as this is not in the scope of this paper). This leads to the following posterior density

$$\begin{aligned}\pi(\mathbf{u}, \mathbf{v}, \kappa_u, \kappa_v \mid \mathbf{y}, \boldsymbol{\alpha}, \boldsymbol{\beta}) \propto\ & \kappa_v^{\alpha_v + \frac{n}{2} - 1} \kappa_u^{\alpha_u + \frac{n-1}{2} - 1} \\ & \times \exp\Big\{-\kappa_v\beta_v - \kappa_u\beta_u + \mathbf{y}^\top\boldsymbol{\eta} - \mathbf{e}^\top\exp(\boldsymbol{\eta}) - \frac{\kappa_u}{2}\mathbf{u}^\top\mathbf{R}\mathbf{u} - \frac{\kappa_v}{2}\mathbf{v}^\top\mathbf{v}\Big\},\end{aligned} \tag{2}$$

which is not a GMRF anymore. Since integration of this density is not feasible, we use MCMC sampling and approximation methods to estimate $\mathbf{u}, \mathbf{v}, \kappa_u$ and κ_v. In the reminder of this section we discuss different methods for this inference. Gibbs samplers implemented in openBUGS, two hand coded R implementations and the MCMC method of the R-package **INLA**, as well as an integrated nested Laplace approximations from **INLA** are presented.

2.1. openBUGS implementation

The previously described model is a BHM with four levels. The dependency structure between the random variables (nodes), their levels, and distributions are shown in Figure 2.

Note that the graph $\mathcal{G}$ is directed and acyclic. Further, each node $\nu \in \mathcal{G}$ is independent of every other node given its parent nodes $pa(\nu)$. This implies that a factorization of the full joint distribution of all nodes in $\mathcal{G}$ is proportional to $\prod_{\nu \in \mathcal{G}} \pi(\nu \mid pa(\nu))$, which is used by the openBUGS engine to generate samples from the posterior distribution (Lunn *et al.* 2013).

The openBUGS language is used to specify such models and communicate them to the openBUGS engine. Similar to the graph representation, we define the distribution of each node given its parent nodes using an R like syntax. The model description is declarative, meaning that the order of the node–definitions is irrelevant. As usual, the symbol "$\sim$" stands for "is distributed as." In order to specify the spatially structured term $\mathbf{u}$, the `car.normal()` distribution from the geoBUGS extension is used. Since the distribution of $\mathbf{u}$ is implemented with a sum-to-zero constraint, we add an additional intercept with an improper flat prior. This construct is claimed to be equivalent to a spatially structured term $\mathbf{u}$ without constraint (Lunn *et al.* 2013, p. 264). We save the following model to a text file "model.txt." Note that the code blocks corresponds to the levels likelihood, convolution-prior, and prior in Figure 2.

```
model{
    for(i in 1:N){
           Y[i] ~ dpois(landa[i])
           log(landa[i]) <- log(E[i]) + u[i] + v[i] }

    for(i in 1:N){ u[i] <- uConstr[i] + intercept }
    intercept ~ dflat()
    uConstr[1:N] ~ car.normal(adj[], weights[], num[], kappaU)
    for(k in 1:sumNumNeigh) { weights[k] <- 1 }
    for(i in 1:N){ v[i] ~ dnorm(0, kappaV) }

    kappaU ~ dgamma(1, .5); kappaV ~ dgamma(1, .01)
}
```

The observed and expected counts `Y` and `E`, as well as the neighborhood structure `adj` are saved in a separate text file "data.txt" (see supplementary material). The arguments of the `car.normal()` distribution are: a sparse adjacency matrix `adj`, the number of regions connected in each row

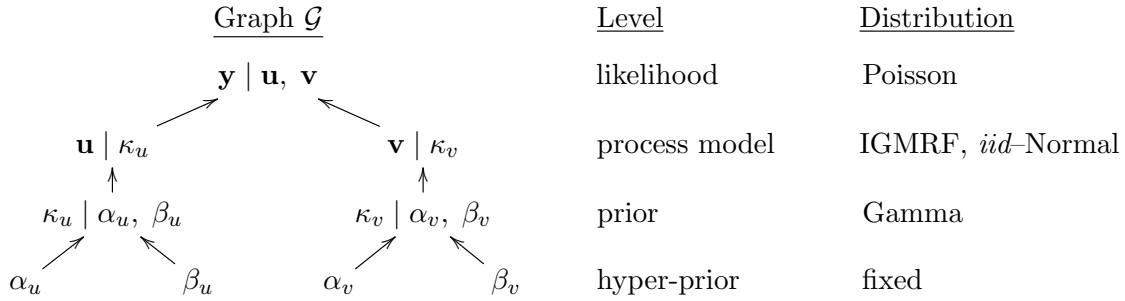

Figure 2: The variables (nodes) and their dependency structure are shown in Graph $\mathcal{G}$. The distributions and levels of the nodes in the model hierarchy are also indicated.

of the adjacency matrix `num[]` (also stored in "data.txt"), the precision of `u`, i.e., `kappaU`, and a vector of 1's in `weight`. (We do not weight the adjacency structure). For another BUGS example of a BYM model see Bivand *et al.* (2013).

The R function `bugs()` from the R-package **R2OpenBUGS** provides a convenient user interface to openBUGS (Sturtz *et al.* 2005). 300,000 samples from the posterior distribution are generated with the following call. A thinning of 20 and a burn-in of 5,000 × 20 = 100,000 is specified, resulting in 10,000 actually returned samples per variable:

```
R> require(R2OpenBUGS)
R> b <- bugs(model.file = "model.txt", data = "data.txt",
+           inits = function(){list(kappaU = 10, kappaV = 100, intercept = 1)},
+           parameters = c("kappaU", "u", "kappaV", "v"), n.iter = 15000,
+           n.burnin = 5000, n.thin = 20, n.chains = 1, bugs.seed = 2)
```

We manually set initial values for `kappaU` and `kappaV` via the argument `inits`. Further, the argument `parameters` specify variables for which samples are stored and returned to R. We only simulate one chain and set `n.chains = 1` for demonstration, but we recommend simulating several chains with different initial values that help to assess convergence. In fact, some comparisons in Section 2.4 are based on 200 different chains. openBUGS selects an appropriate sampling method for each node automatically.

Figure 3 shows diagnostic plots for the first 1,000 samples of the Markov chains for κ_u and κ_v, and Figure 1 shows the resulting posterior mean field of $\mathbf{u}$. We refer to (Knorr-Held and Best 2001) for an (epidemiological) interpretation of the results.

2.2. Two R implementations

In the following, a hand coded R implementation (R Core Team 2012) of a Gibbs sampler with Metropolis–Hastings (MH) step is presented. To simplify notations of densities, the variables $\mathbf{y}$, $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ following the conditioning sign are omitted, e.g., we write $\pi(\mathbf{u}, \boldsymbol{\eta}, \kappa_u, \kappa_v)$ instead of $\pi(\mathbf{u}, \boldsymbol{\eta}, \kappa_u, \kappa_v \mid \mathbf{y}, \boldsymbol{\alpha}, \boldsymbol{\beta})$. We start the sampling procedure by first sampling κ_u and κ_v from

$$\pi(\kappa_u \mid \mathbf{u}, \boldsymbol{\eta}) \propto \kappa_u^{\alpha_u + \frac{n-1}{2} - 1} \exp\left\{-\kappa_u \beta_u - \frac{\kappa_u}{2} \mathbf{u}^\top \mathbf{R} \mathbf{u}\right\},$$
$$\pi(\kappa_v \mid \mathbf{u}, \boldsymbol{\eta}) \propto \kappa_v^{\alpha_v + \frac{n}{2} - 1} \exp\left\{-\kappa_v \beta_v - \frac{\kappa_v}{2} \mathbf{v}^\top \mathbf{v}\right\}.$$

In a second step, the parameter $\mathbf{u}$ and $\boldsymbol{\eta}$ are updated jointly using a MH step. To do this, we rewrite $\pi(\mathbf{u}, \boldsymbol{\eta} \mid \kappa_u, \kappa_v)$, which resulted from the posterior distribution in Equation (2). Recall that $\mathbf{v}$ can be expressed as $\boldsymbol{\eta} - \mathbf{u}$.

$$\pi(\mathbf{u}, \boldsymbol{\eta} \mid \kappa_u, \kappa_v) \propto \exp\left\{\mathbf{y}^\top \boldsymbol{\eta} - \mathbf{e}^\top \exp(\boldsymbol{\eta}) - \frac{1}{2} \left(\mathbf{u}^\top, \boldsymbol{\eta}^\top\right) \begin{pmatrix} \kappa_u \mathbf{R} + \kappa_v \mathbf{I} & -\kappa_v \mathbf{I} \\ -\kappa_v \mathbf{I} & \kappa_v \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ \boldsymbol{\eta} \end{pmatrix}\right\} \quad (3)$$

The idea is to construct a proposal density for $\mathbf{u}$ and $\boldsymbol{\eta}$ that is a GMRF and thus easy to sample from. In addition, the proposal density should approximate the density in Equation (3) well to achieve a reasonable acceptance rate. We use the second-order Taylor expansion of $\mathbf{y}^\top \boldsymbol{\eta} - \mathbf{e}^\top \exp(\boldsymbol{\eta})$ around $\tilde{\boldsymbol{\eta}}$, which is $\boldsymbol{\eta}^\top \mathbf{b}(\tilde{\boldsymbol{\eta}}) - \frac{1}{2} \boldsymbol{\eta}^\top \operatorname{diag}(\mathbf{c}(\tilde{\boldsymbol{\eta}})) \boldsymbol{\eta}$, with $\mathbf{c}(\tilde{\boldsymbol{\eta}}) = \mathbf{e} \exp(\tilde{\boldsymbol{\eta}})^\top$ and $\mathbf{b}(\tilde{\boldsymbol{\eta}}) = \mathbf{y} + (\tilde{\boldsymbol{\eta}} - \mathbf{1}) \mathbf{c}(\tilde{\boldsymbol{\eta}})^\top$. This leads to the normal proposal density

$$q(\mathbf{u}^\star, \boldsymbol{\eta}^\star, \tilde{\boldsymbol{\eta}} \mid \kappa_u, \kappa_v) \propto \exp\left\{-\frac{1}{2} \left(\mathbf{u}^\top, \boldsymbol{\eta}^\top\right) \begin{pmatrix} \kappa_u \mathbf{R} + \kappa_v \mathbf{I} & -\kappa_v \mathbf{I} \\ -\kappa_v \mathbf{I} & \kappa_v \mathbf{I} + \operatorname{diag}(\mathbf{c}(\tilde{\boldsymbol{\eta}})) \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ \boldsymbol{\eta} \end{pmatrix} + \mathbf{b}(\tilde{\boldsymbol{\eta}})^\top \boldsymbol{\eta}\right\}.$$

The proposals $\mathbf{u}^\star$ and $\boldsymbol{\eta}^\star$ are then accepted with probability

$$\alpha = \min\left\{1, \frac{\pi(\mathbf{u}^\star, \boldsymbol{\eta}^\star \mid \kappa_u, \kappa_v)}{\pi(\mathbf{u}, \boldsymbol{\eta} \mid \kappa_u, \kappa_v)} \frac{q(\mathbf{u}, \boldsymbol{\eta}, \tilde{\boldsymbol{\eta}}^\star \mid \kappa_u, \kappa_v)}{q(\mathbf{u}^\star, \boldsymbol{\eta}^\star, \tilde{\boldsymbol{\eta}} \mid \kappa_u, \kappa_v)}\right\}. \tag{4}$$

As $\tilde{\boldsymbol{\eta}}$ we could take any value, but an optimized choice improves the approximation, and, hence increases the acceptance rate of the sampler. We set $\tilde{\boldsymbol{\eta}}$ to an approximation of the posterior mode. The latter is derived using two steps of the Newton–Raphson algorithm, which (under regularity conditions) converges to the posterior mode. To be more specific, we first set $\tilde{\boldsymbol{\eta}}$ to the current value of the chain and derive $\mathbf{b}(\tilde{\boldsymbol{\eta}})$ and $\mathbf{c}(\tilde{\boldsymbol{\eta}})$. Next, we set $\tilde{\boldsymbol{\eta}} = \mathbf{b}(\tilde{\boldsymbol{\eta}})/\mathbf{c}(\tilde{\boldsymbol{\eta}})$ and repeat this procedure twice. The same procedure with $\boldsymbol{\eta}^\star$ as the starting point is applied to find $\mathbf{b}(\tilde{\boldsymbol{\eta}}^\star)$ and $\mathbf{c}(\tilde{\boldsymbol{\eta}}^\star)$. This leads to a suitable acceptance rate of about 48% (Roberts and Rosenthal 2001). See Rue and Held (2005) for more details and other options to increase the accuracy of the proposal density. Note that a third Newton–Raphson iteration for finding $\tilde{\boldsymbol{\eta}}$ and $\tilde{\boldsymbol{\eta}}^\star$ does not increase the acceptance rate. With a single Newton–Raphson iteration, the chain may not converge at all.

Next, we guide the reader through the R code of the Gibbs sampler. Note the use of the R-package **spam**, which provides fast methods for sparse matrix algebra (Furrer 2013; Furrer and Sain 2010). The package contains the oral cancer data and the corresponding adjacency matrix, which we load first. `n = 544` is the number of districts.

```
R> require(spam); data(Oral); attach(Oral)
R> path <- system.file("demodata/germany.adjacency", package = "spam")
R> A <- adjacency.landkreis(path); n <- dim(A)[1]
```

We set a seed value to initialize the random number generator and the number of desired samples (300,000). Further, we define the same hyper-parameters as in the openBUGS implementation.

```
R> set.seed(2)
R> hyperA <- c(1, 1); hyperB <- c(0.5, .01)
R> totalg <- 300000
```

We build some variables to store the samples and set initial values

```
R> upost <- vpost <- array(0, c(totalg, n))
R> kpost <- array(NA, c(totalg, 2)); accept <- rep(NA, totalg)
R> upost[1,] <- vpost[1,] <- rep(.001, 544); kpost[1,] <- c(10, 100)
```

Now we construct some quantities, which are (repetitively) used during the sampling.

```
R> eta <- upost[1,] + vpost[1,]
R> C <- exp(eta) * E; diagC <- diag.spam(c(rep(0, n), C))
R> b <- c( rep(0, n), Y + (eta - 1) * C)
R> Qu <- R <- precmat.IGMRFirreglat(A); dim(Qu) <- c(2 * n, 2 * n)
R> Qv <- as.spam(rbind(cbind( diag(n), -diag(n)),
+                       cbind(-diag(n),  diag(n))))
R> Q <- kpost[1,1] * Qu + kpost[1,2] * Qv + diagC
R> struct <- chol(Q, memory = list(nnzcolindices = 6467))
R> uRuHalf <- t(upost[1,]) %*% (R %*% upost[1,]) / 2
R> vvHalf <- t(vpost[1,]) %*% vpost[1,] / 2
R> postshape <- hyperA + c(n - 1, n) / 2
```

We start the loop of the Gibbs sampler and repeat the following steps: sample κ_u and κ_v (1st block), find an optimized $\tilde{\boldsymbol{\eta}}$ using two Newton–Raphson iterations (2nd block), draw $\mathbf{u}$ and $\boldsymbol{\eta}$ from the Taylor expansion around $\tilde{\boldsymbol{\eta}}$ (3rd block), find $\boldsymbol{\eta}^\star$ and calculate the log–acceptance probability $\log(\alpha)$ (4th block), accept or reject the draw and accordingly update the parameters (5th block). Note that "⋆" in the equations corresponds to "_" in the R code.

```
R> for (i in 2:totalg) {
+       kpost[i,] <- rgamma(2, postshape, hyperB + c(uRuHalf, vvHalf))
+
+       etaTilde <- eta
+       for(index in 1:2){
+           C <- E * exp(etaTilde)
+           diagC <- diag.spam(c(rep(0, n), C))
+           b <- c(rep(0, 544), Y + (etaTilde - 1) * C)
+           Q <- kpost[i,1] * Qu + kpost[i,2] * Qv + diagC
+           etaTilde <- c(solve.spam(Q, b,
+                                    Rstruct = struct))[1:n + n]
+       }
+       C <- exp(etaTilde) * E; diagC <- diag.spam(c(rep(0, n), C))
+       b <- c( rep(0, n), Y + (etaTilde - 1) * C)
+       Q <- kpost[i,1] * Qu + kpost[i,2] * Qv + diagC
+
+       x_ <- c(rmvnorm.canonical(1, b, Q, Rstruct = struct))
+       upost[i,] <- x_[1:n]; eta_ <- x_[1:n + n]; vpost[i,] <- eta_ - upost[i,]
+       uRuHalf_ <- t(upost[i,]) %*% (R %*% upost[i,]) / 2
+       vvHalf_ <- t(vpost[i,]) %*% vpost[i,] / 2
+
+       etaTilde_ <- eta_
+       for(index in 1:2){
+         C_ <- E * exp(etaTilde_)
+         diagC_ <- diag.spam(c(rep(0, n), C_))
+         b_ <- c(rep(0, 544), Y + (etaTilde_ - 1) * C_)
+         Q_<- kpost[i,1] * Qu + kpost[i,2] * Qv + diagC_
+         etaTilde_ <- c(solve.spam(Q_, b_,
+                                   Rstruct = struct))[1:n + n]
+       }
+       C_ <- exp(etaTilde_) * E; diagC_ <- diag.spam(c(rep(0, n), C_))
+       b_ <- c( rep(0, n), Y + (etaTilde_ - 1) * C_)
+       Q_ <- kpost[i,1] * Qu + kpost[i,2] * Qv + diagC_
+       logPost_ <- sum(Y * eta_ - E * exp(eta_)) -
+           kpost[i,1] * uRuHalf_ - kpost[i, 2] * vvHalf_
+       logPost  <- sum(Y * eta - E * exp(eta)) - kpost[i,1] * uRuHalf -
+           kpost[i,2] * vvHalf
+       logApproxX_ <- - kpost[i,1] * uRuHalf_ - kpost[i,2] * vvHalf_ -
+           sum(.5 * eta_^2 * C) + sum(b * eta_)
+       logApproxX  <- - kpost[i,1] * uRuHalf  - kpost[i,2] * vvHalf -
+           sum(.5 * eta^2 * C_) + sum(b_ * eta)
+       logAlpha <- min(0, logPost_ - logPost + logApproxX - logApproxX_)
+
```

```
+      if (log(runif(1)) < logAlpha) {
+          uRuHalf <- uRuHalf_;  vvHalf <- vvHalf_
+          eta <- eta_; b <- b_; C <- C_; accept[i] <- 1
+      } else{
+        accept[i] <- 0; upost[i,] <- upost[i-1,]; vpost[i,] <- vpost[i-1,]}
+  }
```

Finally, we eliminate a burn-in of 100,000 samples and keep the posterior values of every 20th loop to obtain chains of a length of 10,000 (not shown). Diagnostic plots are shown in Figure 3.

Incorrect R implementation Furrer and Sain (2010) give an implementation of the BYM model that incorrectly calculates the acceptance probability. More specifically, on page 19, the line

```
factmp <- (postshape - 1) * (log(kstar) - log(kpost[ig-1,1]))
```

should read

```
factmp <- (postshape - 1) * (log(kstar) - log(kpost[ig-1,]))
```

Because only one element of `kpost[ig-1,]` is used (i.e., κ_v is set to κ_u), the acceptance probability is incorrect and overly high. Retrospectively, an acceptance rate of about 97% could have been an indication of an incorrectly calculated acceptance probability. The sampler uses the proposal density $q(\kappa_u^\star, \kappa_v^\star, \mathbf{u}^\star, \boldsymbol{\eta}^\star, \kappa_u, \kappa_v, \mathbf{u}, \boldsymbol{\eta})$ (jointly updating $\kappa_u, \kappa_v, \mathbf{u}, \boldsymbol{\eta}$), which is constructed on the previous value of $\boldsymbol{\eta}$ without using Newton–Raphson iterations. That leads to a very low acceptance rate, if the correct acceptance probability is used.

The sampler is available in its original version using `demo("article-jss-example2")` from **spam**. For this paper we have adjusted the burn-in, thinning, and sampling size parameters.

2.3. INLA implementation

As opposed to simulation based inference methods, the R-package **INLA** uses nested Laplace approximations to estimate model parameters (Rue *et al.* 2009a).

In order to fit the model, we first load the R-package **INLA** and the oral cancer data. `path` contains the path to the corresponding adjacency matrix. The package, documentation and examples are available on http://www.r-inla.org/.

```
R> require(INLA); data(Oral)
R> path <- system.file("demodata/germany.graph", package = "INLA")
```

Since **INLA** requires an index variable for each of the modeled components $\mathbf{u}$ and $\mathbf{v}$, we have to duplicate the index column in the data frame.

```
R> Oral.inla <- cbind(Oral, region.struct = Oral$region)
```

Next, we define the model though a formula. The functions `f()` specify the priors for $\mathbf{u}$ and $\mathbf{v}$, respectively. For $\mathbf{u}$ a regional structured prior is selected by setting `model = "besag"` and supplying a graph and a hyper-prior. We choose an unconstrained model (`constr = FALSE`), which implies that the intercept is absorbed by this random effect. Hence, the intercept is not

identifiable, and we remove it from the formula through `-1` in the first line. The *iid* random effect $\mathbf{v}$ is specified in the second `f()` function. Note that `theta` corresponds to $(\log(\kappa_u), \log(\kappa_v))^\mathsf{T}$ and, therefore, "`loggamma`" priors are specified as having the same parameters as in the previous implementations.

```
R> formula <- Y ~ -1 +
+   f(region.struct, model = "besag", graph = path,
+     hyper = list(theta = list(prior = "loggamma", param = c(1, 0.5))),
+     constr = FALSE) +
+   f(region, model = "iid",
+     hyper = list(theta = list(prior = "loggamma", param = c(1, 0.01))))
```

Internally, **INLA** reparametrizes by setting $\mathbf{x}^\mathsf{T} = (\mathbf{u}^\mathsf{T}, \boldsymbol{\eta}^\mathsf{T})$, as is commonly done (Gelfand *et al.* 1995). Finally, we fit the model.

```
R> i.out <- inla(formula, family = "poisson", data = Oral.inla, E = E,
+                verbose = TRUE)
```

An alternative is to use the MCMC method of **INLA**, called by

```
R> wd.mcmc = tempfile()
R> try(inla(formula, family = "poisson", data = Oral.inla, E = E,
+           working.directory = wd.mcmc, keep = TRUE,
+           inla.arg = "-m mcmc -N 300000 -T 20 -S .01", verbose = TRUE))
```

Currently, this MCMC method only works with the testing version of **INLA**. Additionally, the computations are carried out, but an error is returned to R. The results are nevertheless accessible in the temporary directory (path in `wd.mcmc`). We removed a burn-in of 100,000 samples and applied a thinning of 20. The results in this paper are based using the following **INLA** version:

```
R> inla.version()
INLA build date ...: Wed Feb 13 09:38:42 CET 2013
INLA hgid .........: hgid: 6d1015c52579  date: Wed Feb 13 09:28:06 2013 +0100
(output truncated)
R> sessionInfo()
R version 2.15.2 (2012-10-26)
Platform: i686-pc-linux-gnu (32-bit)
(output truncated)
```

The sampler of this particular implementation seems to work well. Diagnostic plots are shown in Figure 3. The acceptance rate was about 12%. However, the sampler in the current **INLA** testing version has a very low acceptance rate of about 3%. We hope that the issues will be addressed soon.

2.4. Comparison of the implementations

In order to keep the article at a reasonable length, we will not report convergence diagnostics of the individual samplers. Rather, we will focus on the comparison of the implementations. The assessment of "equality" of two samples is essentially the assessment, if two multivariate samples are drawn from the same distribution. Given the dimensionality of $\{\mathbf{u}, \mathbf{v}, \kappa_u, \kappa_v\}$, there is little

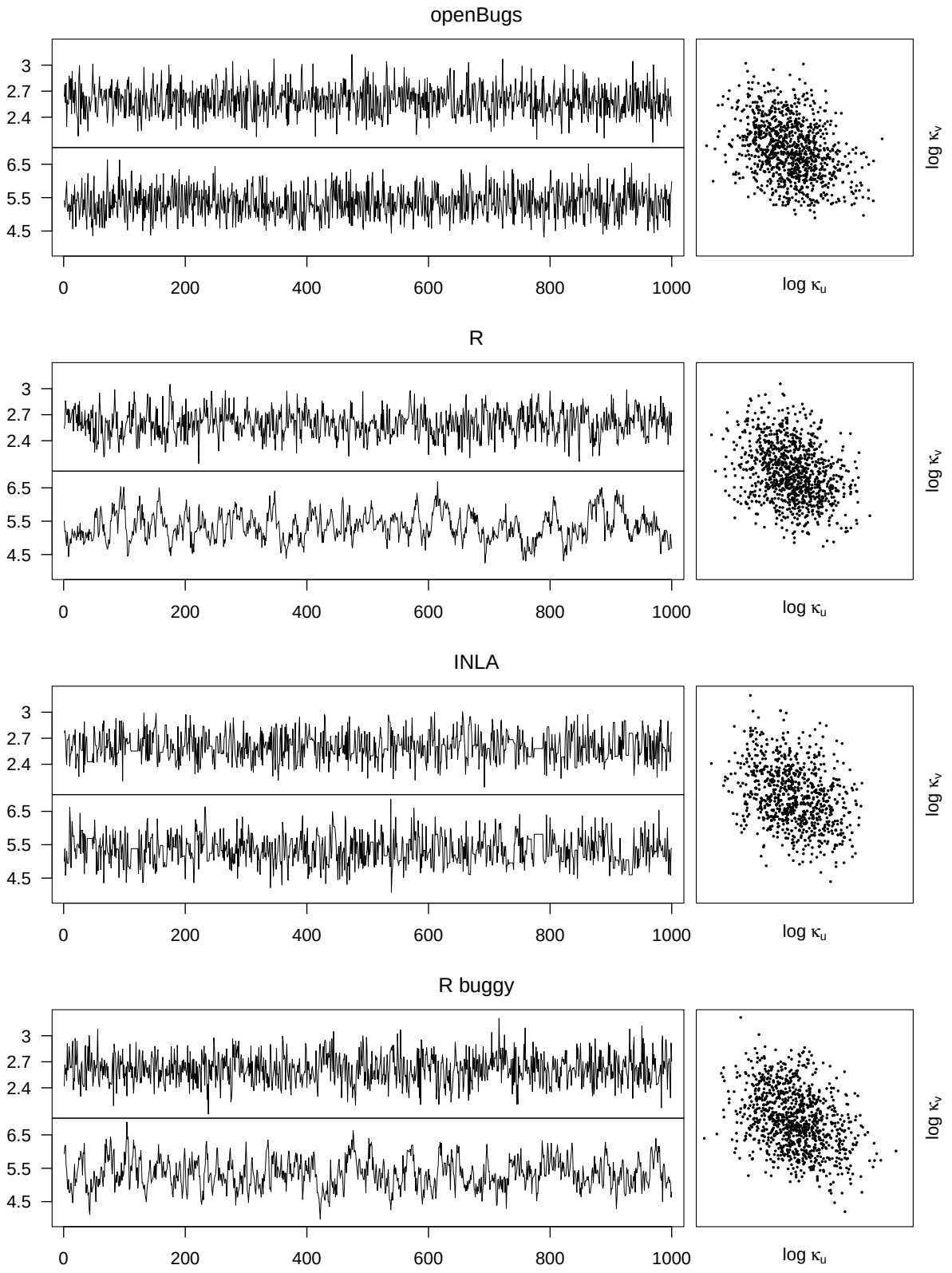

Figure 3: Diagnostic plots for the first 1,000 post burn-in samples from the openBUGS, R, **INLA** MCMC and R buggy BYM implementations are shown. Each panel consists of trace plots for $\log \kappa_u$ (upper) and $\log \kappa_v$ (lower), respectively. The mixing of $\log \kappa_u$ and $\log \kappa_v$ is shown in a scatter plot (right). The chains were already thinned with a factor of 20.

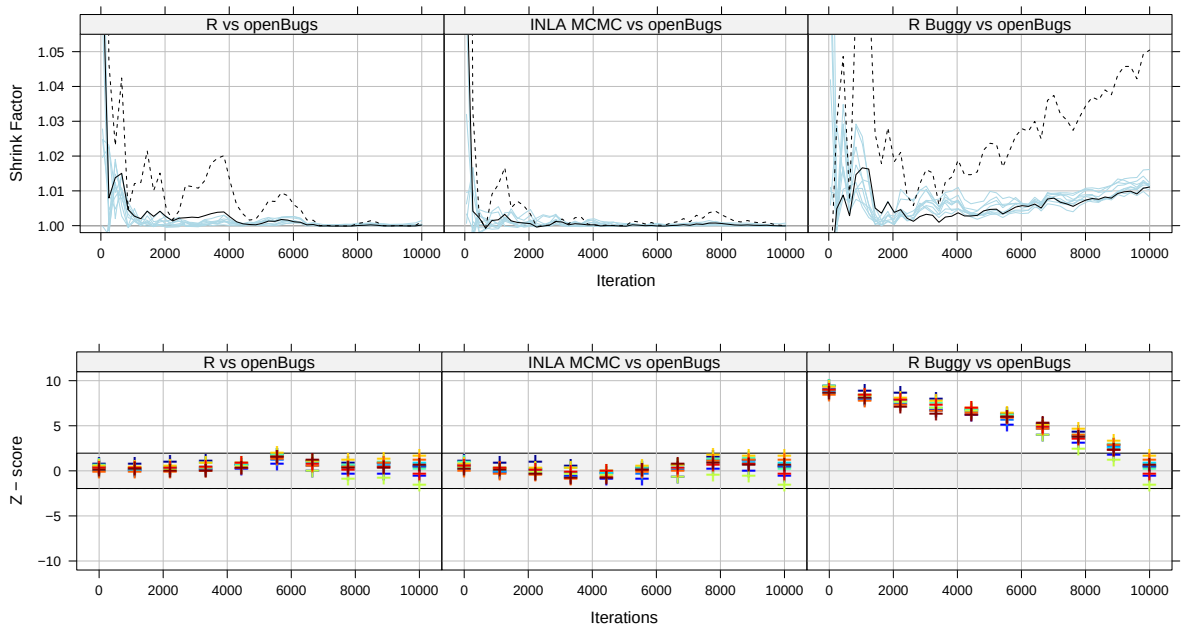

Figure 4: Comparison of the R, buggy R and INLA MCMC chains of κ_u against ten different κ_u chains from openBUGS using functions from R-package **coda**. Upper panels: Gelman plots with the median shrink factor for 10 comparisons (sold lines) and the 95% quantile for the first comparison (dashed line). Lower panels: Geweke plots with one color per comparison. Here two chains were compared by appending them to one single chain and setting `frac1` and `frac2` in the `geweke.diag()` function to 0.5.

hope that the formal tests presented in Rosenbaum (2005); Dhar *et al.* (2011) can be used. Alternatively, one can investigate individual posteriors (here κ_u, κ_v) and summary statistics of **u** and **v** (Wigley and Santer 1990; Li and Smerdon 2012).

We now illustrate a series of tools to compare two sample(r)s. Some of these are commonly used and reported here for completeness; others are new and complement the existing ones.

Figure 3 shows trace plots for $\log \kappa_u$ and $\log \kappa_v$ for the four different MCMC implementations. The trace plots of $\log \kappa_v$ for the R-implementations exhibit particularly high auto-correlations compared to those from openBUGS. This auto-correlation can be reduced by increasing the thinning value. An alternative is to jointly update all parameters of the models (κ_u, κ_v, **u**, $\boldsymbol{\eta}$). See Knorr-Held and Rue (2002) for a discussion of different (block-) update procedures in a similar setting. The scatter plots of $\log \kappa_u$ and $\log \kappa_v$ in the same Figure show no suspicious pattern.

A numerical summary of the posterior distribution of κ_u and κ_v is given in Table 1, showing little evidence of issues with the incorrect R sampler. For the posterior mean, the naive standard error (SE) and the time-series standard error (TS SE) derived with the R-package **coda** are given. With respect to the TS SE the variation in the mean estimates seems to be large.

In Figure 4, ten different chains for κ_u from openBUGS are compared against κ_u chains from the other sampler implementations. The diagnostic functions `gelman.plot()` and `geweke.diag()` from the R-package **coda** are used (Plummer *et al.* 2006). In the Gelman plots (Brooks and Gelman 1998; Gelman and Rubin 1992), each line represents the median shrink factor of a comparison of two chains. For the first comparison (black line), the 95% quantile is drawn as dashed

	Implementation	Mean	SE	TS SE	SD	2.5%	50%	97.5%
κ_u	openBUGS	13.58	0.022	0.022	2.22	9.83	13.38	18.52
	R	13.62	0.022	0.035	2.22	9.91	13.37	18.57
	R buggy	13.93	0.023	0.029	2.29	10.13	13.72	19.01
	INLA	13.62	–	–	2.20	9.77	13.46	18.38
	INLA MCMC	13.62	0.022	0.025	2.21	9.91	13.41	18.52
κ_v	openBUGS	227.1	1.05	1.05	104.8	94.05	204.55	492.80
	R	231.5	1.10	3.51	110.0	93.47	207.57	513.59
	R buggy	231.6	1.11	2.60	111.0	94.27	204.91	516.22
	INLA	234.6	–	–	115.6	93.51	207.39	532.33
	INLA MCMC	231.3	1.10	1.44	109.9	94.06	206.53	516.35

Table 1: Summary table for the estimates of κ_u and κ_v generated with openBUGS, two hand coded R implementations, **INLA** and the MCMC-method of the **INLA** package. The estimates are calculated based on 10,000 samples of one chain (generated with 300,000 MCMC iterations in total). In addition to the standard error (SE), the time-series standard error (TS SE), derived with the R-package **coda**, are given.

line. In the Geweke plot (Geweke 1992) 10 pairs of two chains were compared by appending them to one single chain and setting `frac1` and `frac2` in the corresponding R function to 0.5. Only the chains from the R buggy implementation seem to be different from the openBUGS ones. When we reduced the burn-in period from 100,000 to 5,000 samples, this effect was much less prominent.

A difficulty is often that the variability between individual chains is larger than between chains from different implementation methods. When using shorter chains this variability is even larger, and interpretation is more difficult. We recommend using empirical cumulative distribution functions (ECDFs) to compare realizations from different samplers (Figure 5, upper panels), although density estimates with the default choice of the kernel estimator and bandwidth selection of `density()` lead to an acceptable result too (Figure 5, lower left panel). When making these comparisons, we must keep in mind that the **INLA** approximations are tuned to be most accurate around the median. Hence, comparing ECDFs and tails of densities might lead to unfair comparisons. A functional boxplot approach (Sun *et al.* 2012) may help to identify outlying densities. It calculates for each curve a (modified) band depth and orders the curves from the center outwards. The introduced measure defines functional quantiles and the outlyingness of a curve. The lower right panel of Figure 5 indicates that for κ_v all non-openBUGS runs are declared as outliers.

Figure 6 shows Q-Q-plots for the precision parameter κ_u. Empirical quantiles (from an arbitrary ordered sample) from the openBUGS sampler (x-axis) and the empirical quantiles from all others (y-axis) are drawn. For a better display, we have jittered the x-axis values while preserving the order. The quantiles of the incorrect sampler are shown as a red dashed line and are clearly set off from all the other lines.

Another alternative to compare sample(r)s is to use a simple clustering algorithm of the resulting empirical densities or distributions (Figure 7). Again, the dissimilarity (here measured by the classical Euclidean distance) is much larger between the incorrect sample and all the others. However, the **INLA** approximation stands out as well. Reducing the chain length and reducing the number of chains (or similarly increasing them) has little influence on the detection capability

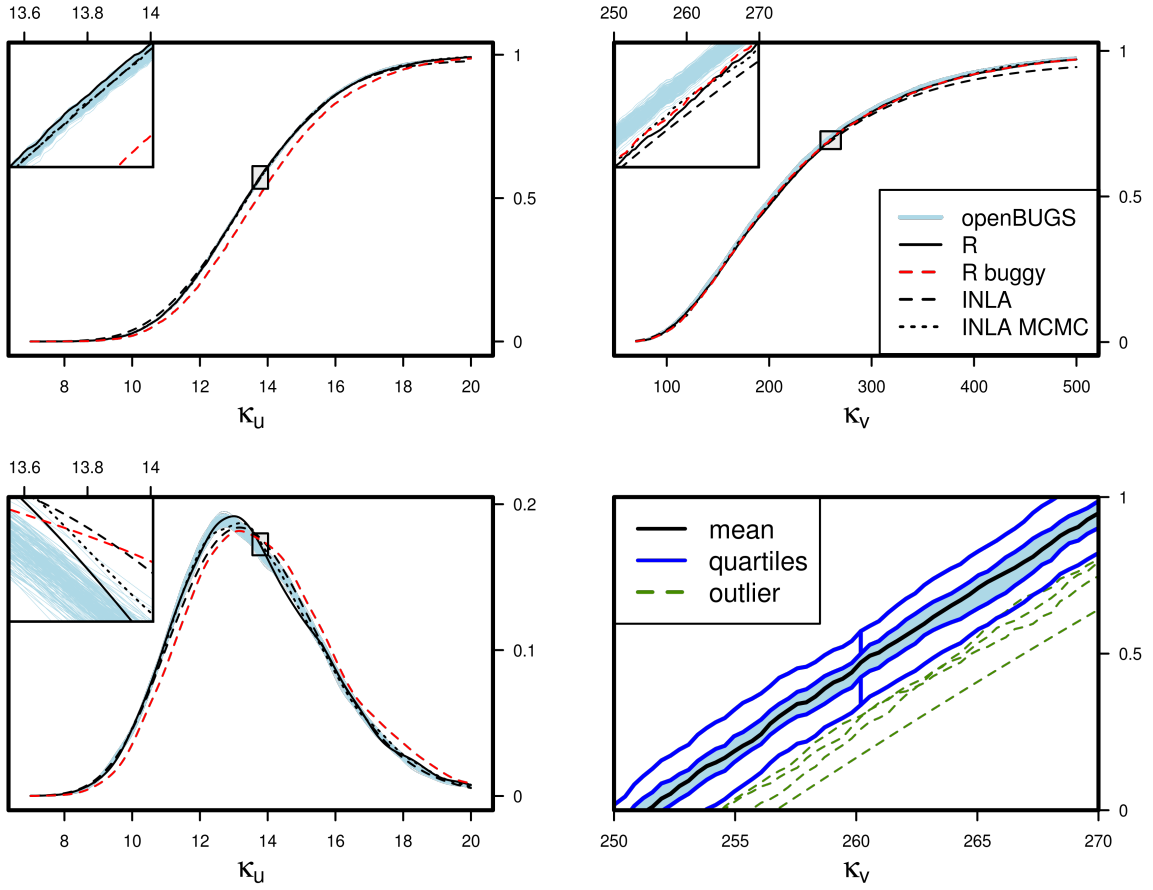

Figure 5: Comparison of the samples for κ_u (left) and κ_v (right) resulting from 200 openBUGS runs, the two R runs, and the **INLA** runs. Upper panels: empirical cumulative distribution functions (ECDFs). Bottom left panel: kernel density estimates with automatically chosen parameters (not recommended). Bottom right panel: functional boxplot where three openBUGS, the **INLA**, and the R buggy runs are marked as outliers (green lines).

of the clustering approach.

Plots of posterior mean of the spatial fields **u** and **v** (like those shown in Figure 1) are very difficult to compare, as the differences are small and are masked by the spatial patterns. Other such "simple" diagnostic plots (e.g., median, standard deviation, IQR fields or differences of such) were not helpful to us either.

A more promising tool to compare spatial fields is the scatter plot, shown in Figure 8. There the spatial components averaged over all methods, except the buggy R, (x-axis) are plotted against those from the specific method on the y-axis. The mean absolute deviation D (also shown in the figure) is smaller then 1.61×10^{-3} for the **openBUGS**, R, and **INLA** implementations and 1.76×10^{-2} for the incorrect R-implementation. The corresponding values for **v** are 4.24×10^{-1} and 1.42×10^{-1}, respectively. Overall, the estimated **u** and **v** components of all implementations seem to agree well.

A successful approach to discriminate sample(r)s and hence to identify an incorrect sampler is to take one method as a reference and plot the difference between the empirical densities or empirical distributions of reference and all the others (Figure 9). This is fast, and minor shifts or differences in scale are emphasized. A drawback is that the dependency structure of the

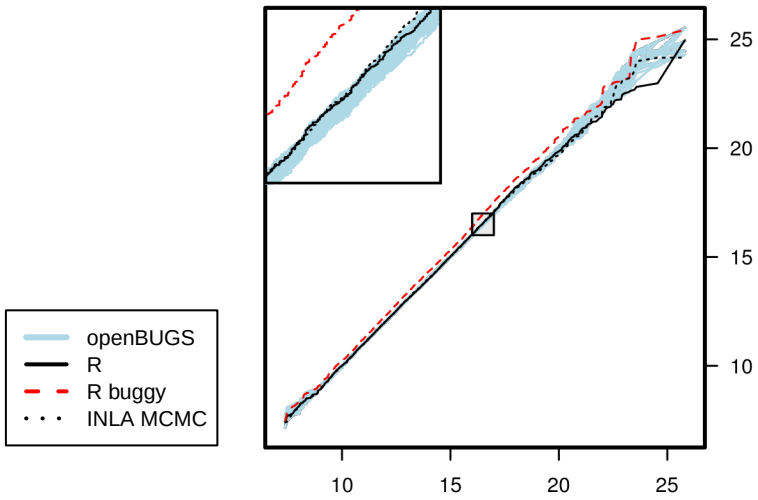

Figure 6: Q-Q-plots based on κ_u samples. The x-axis contains empirical quantiles (from an arbitrary ordered sample) from the openBUGS sampler, and the y-axis contains the empirical quantiles from all other openBUGS runs (blue lines), the R run (solid line), the incorrect (buggy) R run (red dashed line), and the **INLA** MCMC run (doted line).

samples and multiple testing issues are not taken into account. This makes the extension to a formal two-sample Kolmogorov–Smirnov test difficult.

Finally, a clustering algorithm detects again the wrong sampler by looking at (arbitrary) individual districts only. Here one chain is divided into ten subchains of a length of 1,000, and the empirical densities or distributions are calculated. For the district Sigmaringen, the result of the clustering based on these 10×4 distributions is given in Figure 10. All subchains of the incorrect sampler are nicely grouped. An advantage of this clustering approach is that (dis)similarities of several chains of possibly different implementations are summarized in one step.

2.5. Discussion, extensions, pitfalls

The R implementation in Section 2.2 is the most flexible, but it is also the most demanding to code and is obviously the most error-prone. Because of the package **spam**, sparse matrix algebra can be done via a Fortran back-end, which leads to a reasonably fast Gibbs sampler featuring 30 iterations per second on an Intel 2GHz dual-core processor (approximately an hour of computation for a chain of a length of 300,000). Thus, R implementations are reasonably fast, as long as sparse matrices are used. Personal experience shows that, in all practical cases, the time spent to speed up the calculations does not offset the time gain (see Table 1 in Furrer and Sain 2010). Even more efficient ways may involve the concept of a just-in-time compiler for R-code implemented in the R-package **compiler**, which is part of the R base packages (R Core Team 2012). Another option is to run several chains in parallel using the R-package **snowfall** (Knaus 2013). Finally, implementing (parts of) the model in C++ would reduce calculation time; for an example see Gerber (2013). The openBUGS implementation is flexible too, and if the required distributions are available, as in our case, it is much simpler to use. The sampler achieved 84 iterations per second (approximately half an hour of computation for a chain of length 300,000). In the R-package **INLA** the specification of the model is very user-friendly, but

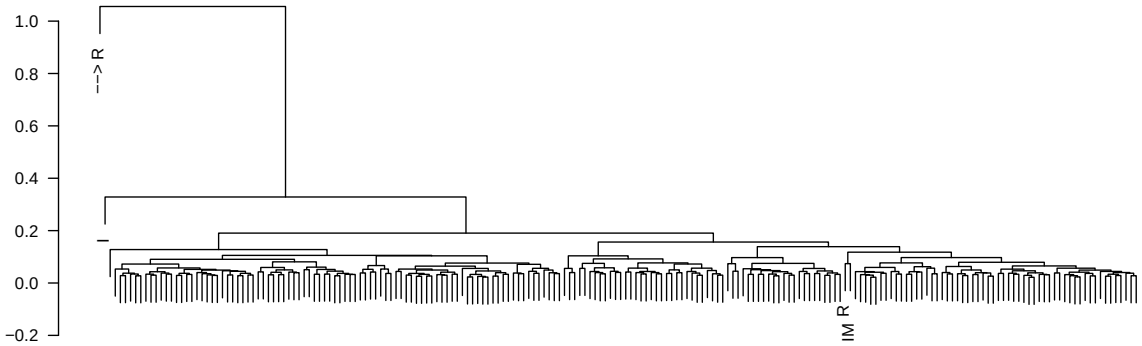

Figure 7: Dendrogram from a hierarchical clustering of the posterior ECDFs of κ_u of the 200 openBUGS runs (no symbol), the R (R), the buggy R (–> R), **INLA** (I) and the **INLA** MCMC (IM) runs.

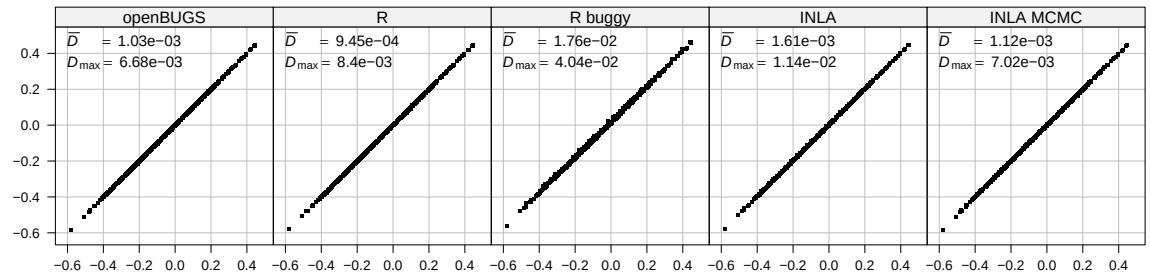

Figure 8: Comparison of the estimates for **u** of the different methods. x-axis: average mean estimates of the openBUGS, R and **INLA** implementations, y-axis: mean estimates of the specific method. $\overline{D}$ indicates the mean absolute deviation and $D_{\max}$ the maximum deviation.

a potential extension to a not-included setting is difficult. However, many cases are included and we foresee further extensions in the near future. The **INLA** method is very fast (less than 4 seconds) and thus is useful where estimation has to be fast. The MCMC method of the R-package **INLA** achieved 34 iterations per second, and we have no doubt that a stable MCMC implementation will be available soon.

An unconstrained random field should be identical to a constrained random field with an intercept with uniform prior. In **INLA** a simple flag switches between both cases. In openBUGS a constrained version is implemented, and one has to add an intercept manually for the unconstrained case. While we did not observe issues with either implementation here, they are not always exactly identical, as is also discussed in the next section.

More complex equality constraints are easily implemented in **INLA** by specifying the option `extraconstr` in `f` and in R via `spam::rmvnorm.const()`, for example. Some of the sampling engines use a so-called centering-on-the-fly approach, and the implications on the equilibrium distribution are not clear to us (see also Schrödle *et al.* 2011).

While generating many (200) long chains with 300,000 iterations each, openBUGS was not able to provide so many non-identical chains. Among the 200 chains, 7 were identical in our case. It seems that we have hit some sort of periodicity of the seed in openBUGS.

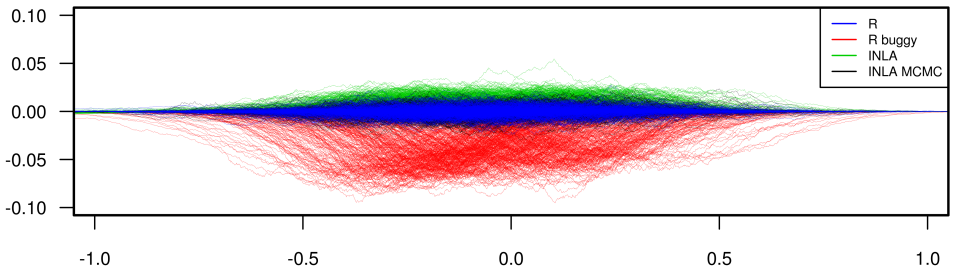

Figure 9: Comparison of posterior ECDFs for all 544 districts of the **u** component. One openBUGS run is used as a reference, and the differences to the R (blue), the incorrect R (red), and the **INLA** methods (green, black) are plotted.

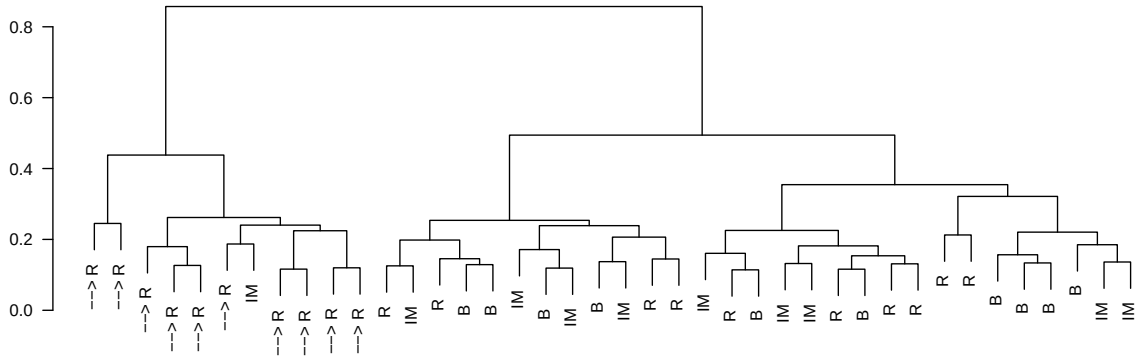

Figure 10: Dendrogram from a hierarchical clustering of the posterior ECDFs of the district Sigmaringen. Each chain was split into ten subchains. Chains from the openBUGS, the R, the incorrect R, and the **INLA** MCMC methods are denoted with 'B', 'R', '- -> R' and 'IM', respectively.

In the following, we point to possible extensions of the BYM model. Covariates that are observed for each region can be included, for example. For the oral cavity data, we could examine the effect of smoking and estimate an adjusted spatially structured component by setting $\boldsymbol{\eta} = \mathbf{u} + \mathbf{v} + \alpha\mathbf{s}$, where $\mathbf{s}$ contains observed smoking covariates of each region. This model is termed "ecological regression" and can be fitted in openBUGS; see (Lunn *et al.* 2013, p. 267) and (Bivand *et al.* 2008, p. 327) and in INLA (Schrödle and Held 2010). The latter paper also discusses the extension to spatio-temporal disease mapping. Further, it is possible to model two or more diseases jointly using a so-called shared component model (Held *et al.* 2005; Rue and Held 2005), which is provided in INLA through the function `besag2` and can be implemented in openBUGS as well. openBUGS also provides a slightly different approach via the `mv.car()` distribution, which extends `car.normal()` in a natural way. MacNab (2010) discusses the similarities between ecological regression and shared component modeling and proposes an ecological regression model that allows the researcher to account for measurement errors in the observed covariates.

3. Leroux model

An alternative model for areal count data was introduced by Leroux *et al.* (1999). In contrast to the BYM model, it has only one random effect component. Without including an intercept or

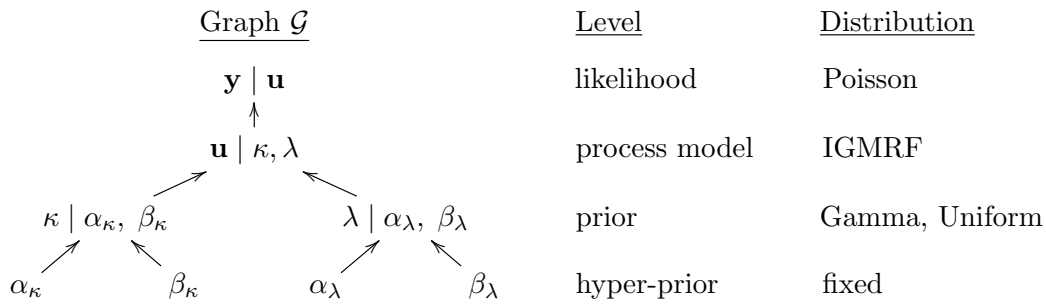

Figure 11: The variables (nodes) and their dependency structure are shown in Graph $\mathcal{G}$. The distributions and levels of the nodes in the model hierarchy are also indicated.

additional covariates, this random effect simply models the log-relative risk $\boldsymbol{\eta}$. To be consistent with the implementations from Section 3.2, we set $\mathbf{u} = \boldsymbol{\eta}$. The separation of spatially structured and *iid* variance is controlled by an additional parameter λ. To be more specific, the same likelihood function as in Equation (1) is used, and $\mathbf{u}$ is modeled by the intrinsic GMRF

$$\pi(\mathbf{u} \mid \kappa, \lambda) \propto \det{}_G(\mathbf{Q}(\lambda))^{\frac{1}{2}} \exp\Big(-\frac{\kappa}{2} \mathbf{u}^\mathsf{T} \mathbf{Q}(\lambda) \mathbf{u} \Big).$$

Where $\kappa > 0$ is a precision parameter, and $\det_G$ denotes the generalized determinant (i.e., the product of all non-zero eigenvalues). The parameter $\lambda \in (0, 1)$ defines the degree of the spatial dependency through $\mathbf{Q}(\lambda) = (1 - \lambda)\mathbf{I} + \lambda\mathbf{R}$. With appropriate (uninformative) priors for κ and λ, we get the posterior distribution

$$\pi(\mathbf{u}, \kappa, \lambda) \propto \kappa^{\frac{n}{2}-1} \det{}_G(\mathbf{Q}(\lambda))^{\frac{1}{2}} \exp\Big(\mathbf{y}^\mathsf{T}\mathbf{u} - \mathbf{e}^\mathsf{T} \exp(\mathbf{u}) - \frac{\kappa}{2} \mathbf{u}^\mathsf{T} \mathbf{Q}(\lambda) \mathbf{u} \Big). \tag{5}$$

In the next section, an R implementation of a Gibbs sampler for the Leroux model is presented. Further, three variations of this Gibbs sampler are discussed in Section 3.2, and a comparison and discussion of these variations follow in Sections 3.4 and 3.5.

3.1. R implementation

To estimate the parameters of the Leroux model, we implement a Gibbs sampler in R and sample from the posterior distribution. First, we sample $\mathbf{u}^\star$ from a normal proposal. To this end we use a second-order Taylor expansion around $\tilde{\mathbf{u}}$ of the term $\mathbf{y}^\mathsf{T}\mathbf{u} - \mathbf{e}^\mathsf{T} \exp(\mathbf{u})$ from the joint density (Equation (5)), yielding the approximation $\mathbf{u}^\mathsf{T}\mathbf{b}(\tilde{\mathbf{u}}) - \frac{1}{2}\mathbf{u}^\mathsf{T} \operatorname{diag}(\mathbf{c}(\tilde{\mathbf{u}}))\mathbf{u}$ with $\mathbf{c}(\tilde{\mathbf{u}}) = \mathbf{e} \exp(\tilde{\mathbf{u}})^\mathsf{T}$ and $\mathbf{b}(\tilde{\mathbf{u}}) = \mathbf{y} + (\tilde{\mathbf{u}} - \mathbf{1})\mathbf{c}(\tilde{\mathbf{u}})^\mathsf{T}$. This leads to the normal proposal density

$$q(\mathbf{u}, \tilde{\mathbf{u}} \mid \kappa, \lambda) \propto \exp\Big(\mathbf{u}^\mathsf{T}\mathbf{b}(\tilde{\mathbf{u}}) - \frac{1}{2}\mathbf{u}^\mathsf{T}\big(\operatorname{diag}(\mathbf{c}(\tilde{\mathbf{u}})) + \kappa\mathbf{Q}(\lambda) \big)\mathbf{u} \Big).$$

The proposal $\mathbf{u}^\star$ is accepted with probability $\min(1, \alpha_\mathbf{u})$, where

$$\log \alpha_\mathbf{u} = \log\left(\frac{\pi(\mathbf{u}^\star \mid \kappa, \lambda)}{\pi(\tilde{\mathbf{u}} \mid \kappa, \lambda)} \frac{q(\tilde{\mathbf{u}}, \mathbf{u}^\star \mid \kappa, \lambda)}{q(\mathbf{u}^\star, \tilde{\mathbf{u}} \mid \kappa, \lambda)} \right).$$

One Newton–Raphson iteration is applied to achieve an acceptance rate of $\approx 50\%$ (compare to Section 2.2). In a second step, κ is sampled from the full conditional

$$\pi(\kappa \mid \mathbf{u}, \lambda) = \kappa^{\frac{n}{2}-1} \exp\Big(-\frac{\kappa}{2} \mathbf{u}^\mathsf{T} \mathbf{Q}(\lambda) \mathbf{u} \Big).$$

Finally, λ is updated using a MH step again. This time we sample the proposal $\lambda^\star$ from a normal density truncated to $(0, 1)$, with the mean equal to the previous value of λ

$$q(\lambda^\star, \lambda \mid \mathbf{u}, \kappa) \propto 1_{(0,1)}(\lambda^\star) \exp\Big(-\frac{\tau}{2}(\lambda^\star - \lambda)^2\Big).$$

The proposed $\lambda^\star$ is then accepted with probability $\min(1, \pi(\lambda^\star \mid \mathbf{u}, \kappa)/\pi(\lambda \mid \mathbf{u}, \kappa))$. The proposal density $q(\lambda^\star, \lambda \mid \mathbf{u}, \kappa)$ does not appear in the calculation of the acceptance rate, since it is symmetric in λ and $\lambda^\star$. $\tau > 0$ is a tuning parameter for the acceptance rate of λ.

Next, we show the R code for this version of the Gibbs sampler in more detail. First, the R-packages **truncdist** (Novomestky and Nadarajah 2012), **spam** (Furrer 2013) and the Germany cancer data are loaded. The number of desired samples (300,000) and arrays for the posterior values are built. Note that we also initialize a `bpost` parameter, which we set to zero in each iteration. This is an artifact from other implementations mentioned in Section 3.2 and can be ignored.

```
R> require(spam); require(truncdist)
R> data(Oral); E <- Oral$E; Y <- Oral$Y; n <- 544
R> A <- as.matrix(adjacency.landkreis(system.file("demodata/germany.adjacency",
+                                                 package = "spam")))
R> totaln <- 300000; upost <- array(NA, c(totaln, n))
R> bpost <- kpost <- lpost <- rep(NA, totaln)
R> accept <- array(0, c(totaln, 3), list(NULL, c("beta", "u", "lambda")))
```

Inital values are set in the first position of the corresponding posterior arrays.

```
R> bpost[1] <- 0; kpost[1] <- 15; lpost[1] <- .9
R> upost[1,] <- rep(c(.1, -.1), 544 / 2); accept[1,] <- 1
```

Next, a tuning parameter for the acceptance probability is set, and repeatedly used values are calculated.

```
R> lambda.proposal.sd <- 0.0408 * 1.74
R> R <- precmat.IGMRFirreglat(A)
R> eigenR <- eigen(R); eigenR.value <- eigenR$values
R> Q <- (1 - lpost[1]) * diag.spam(544) + lpost[1] * R
R> Q.det <- sum(log(lpost[1]* eigenR.value + 1 - lpost[1]))
R> Q.struct <- chol.spam(Q)
R> postshape <- 0.5 * n - 1
```

We loop over the following steps of the Gibbs sampler: update β, which corresponds to setting it to zero (1st block), find an optimized $\tilde{\mathbf{u}}$ and update $\mathbf{u}$ with a MH step (2nd block), update κ (3rd block), and update λ with a MH step (4th block). Note that "$\star$" in the equations corresponds to "_" in the R-code.

```
R> for (i in 2:totaln) {
+    bpost[i] <- 0
+
+    u.tilde <- upost[i-1,]
+    C <- E * exp(u.tilde); B <- Y + (u.tilde - 1) * C
+    Q.tmp <- diag.spam(C) + kpost[i-1] * Q
```

```
+    u.tilde <- c(solve.spam(Q.tmp, B))
+    C.tilde <- E * exp(u.tilde); B.tilde <- Y + (u.tilde - 1) * C.tilde
+    Q.tilde <- diag.spam(C.tilde) + kpost[i-1] * Q
+    u_ <- c(rmvnorm.canonical(1, B.tilde, Q.tilde, Rstruct = Q.struct))
+    u.tilde_ <- u_
+    C_ <- E * exp(u.tilde_); B_ <- Y + (u.tilde_- 1) * C_
+    Q.tmp_ <- diag.spam(C_) + kpost[i-1] * Q
+    u.tilde_ <- c(solve.spam(Q.tmp_, B_))
+    C.tilde_ <- E * exp(u.tilde_); B.tilde_ <- Y + (u.tilde_- 1) * C.tilde_
+    log.alpha.u <- sum(Y * u_) - sum(E * exp(u_))  -
+      sum(Y * upost[i-1,]) + sum(E * exp(upost[i-1,]) ) +
+        sum(upost[i-1,] * B.tilde_) -
+          .5 * t(upost[i-1,]) %*% (diag(C.tilde_) %*% upost[i-1,]) -
+            sum(u_* B.tilde) + .5 * t(u_) %*% (diag(C.tilde) %*% u_)
+    if(exp(log.alpha.u) > runif(1)) { upost[i,] <- u_; accept[i,2] <- 1 }
+    else { upost[i,] <- upost[i-1,] }
+
+    kpost[i] <- rgamma(1, shape = postshape,
+                       rate = .5 * upost[i,] %*% (Q %*% upost[i,]))
+
+    lambda_ <- rtrunc(n = 1, spec = "norm", a = 0, b = 1,
+                      mean = lpost[i-1], sd = lambda.proposal.sd)
+    Q_ <- (1 - lambda_) * diag.spam(544) + lambda_ * R
+    Q.det_ <- sum(log(lambda_* eigenR.value + 1 - lambda_))
+    alpha.lambda <- exp(.5 * (Q.det_ -
+                              kpost[i] * upost[i,] %*% (Q_ %*% upost[i,]) -
+                              Q.det +
+                              kpost[i] * upost[i,] %*% (Q %*% upost[i,]) ))
+    if(alpha.lambda > runif(1)){
+      lpost[i] <- lambda_; Q <- Q_; Q.det <- Q.det_; accept[i,3] <- 1
+    } else { lpost[i] <- lpost[i-1] }
+  }
```

Finally, we eliminate a burn-in of 100,000 samples and keep the posterior values of every 20th loop to obtain chains of a length of 10,000 (not shown). Diagnostic plots are shown in Figure 12. The acceptance probability is tuned to $\approx 49\%$ for the spatial parameter $\mathbf{u}$ and is $\approx 40\%$ for λ.

3.2. Three variations

As mentioned earlier, we implemented three additional variations of the Gibbs sampler. The variations differ in the way the random field is updated (1 block versus 55 updated blocks) and whether there is an intercept (with almost uninformative prior). The R code for the update of $\mathbf{u}$ in 55 blocks and for the update of the intercept was taken from **CARBayes** (Lee 2013). When possible, we just exchanged blocks of code to prevent errors. The corresponding R code is provided in the supplementary material of this paper. For obvious reasons, we call the sampler introduced in the last section "1 block, no Intercept."

1 Block, Intercept Here $\boldsymbol{\eta}$ is modeled as $\beta + \mathbf{u}$ (i.e., an intercept β is added). For β an almost uninformative normally distributed prior with mean zero and variance 10^{10} is set. To

guaranty identifiability of β and $\mathbf{u}$, a so-called centering-on-the-fly approach is implemented. It simply replaces $\mathbf{u}$ by $\mathbf{u} - \mathbf{1}^\top\mathbf{u}/n$ after each draw of $\mathbf{u}$ and might lead to some artifacts that are not entirely clear to us. Other options that simulate directly constrained $\mathbf{u}$'s are implemented, e.g., in **spam**. However, we decided to not use this in order to be consistent with the package **CARBayes**. The acceptance probability was tuned to $\approx 40\%$ for all parameters.

55 Blocks, no Intercept Here the update of the spatial component $\mathbf{u}$ is separated into 55 blocks. This means that a proposal for the first block of size 10 is generated and accepted or rejected, then the second block is updated, and so forth. This version has no intercept and thus no sum-to-zero constraint. The acceptance probability was tuned to $\approx 40\%$ for all parameters.

55 Blocks, Intercept In this variation the spatial component is updated in 55 blocks and an intercept β is added with an almost uninformative normally distributed prior (mean zero and variance 10^{10}). A sum-to-zero constraint on the spatial component is set. This variation corresponds almost to the one implemented R-package **CARBayes**. The only difference is that the **CARBayes** implementation tunes the acceptance probabilities of the MH steps automatically. We run this sampler with 3 different configurations of the tuning parameters, yielding the acceptance probabilities $\approx 40\%$ for all parameters, $\approx 70\%$ for all parameters, and $\approx 35\%$ for the spatial, and $\approx 60\%$ for the other parameters, respectively. The latter is similar to the one resulting from the automatic tuning in the **CARBayes** implementation.

3.3. R-package CARBayes

The R-package **CARBayes** (Lee 2013) provides the function `poisson.leroux()`, which implements a Gibbs sampler similar to the version "55 blocks, intercept." Additionally, this version allows us to specify explanatory variables using a formula interface. The function comes with a mechanism that tunes the acceptance probability automatically. We run the function with the same settings as our implementations.

```
R> require(CARBayes)
R> out <- poisson.lerouxCAR(formula = Y ~ offset(log(E)), data = Oral, W = A,
+                           beta = 0, phi = rep(c(-.1, -1), 544 / 2),
+                           tau2 = 1 / 15, rho = .9, n.sample = 300000,
+                           prior.var.beta = 1e10, prior.max.tau2 = 1e10)
```

We eliminate a burn-in of 100,000 samples and keep the posterior values of every 20th loop to obtain chains of a length of 10,000 (not shown).

3.4. Comparison of the implementations

To compare the four different versions of the MCMC samplers from Section 3.1 and 3.2, we repeated the MCMC runs 100 times for all model implementations. Each chain had a length of 300,000 from which a burn-in of 100,000 was removed and a thinning of 20 was applied. Thus, we end up analyzing 100 chains of a length of 10,000 for λ, κ and $\mathbf{u}$ per implementation. The three additional versions (two with varying acceptance probability and one from the R-package **CARBayes**) are only shown in Figure 14 and are mentioned in the discussion thereof.

Again, many of the tools that we present are commonly used but are reported here for completeness. Figure 12 shows trace plots of $\log\kappa$ and λ for the four different MCMC variations. The trace plots of $\log\kappa$ of the implementations, with an update of the spatial component in 55

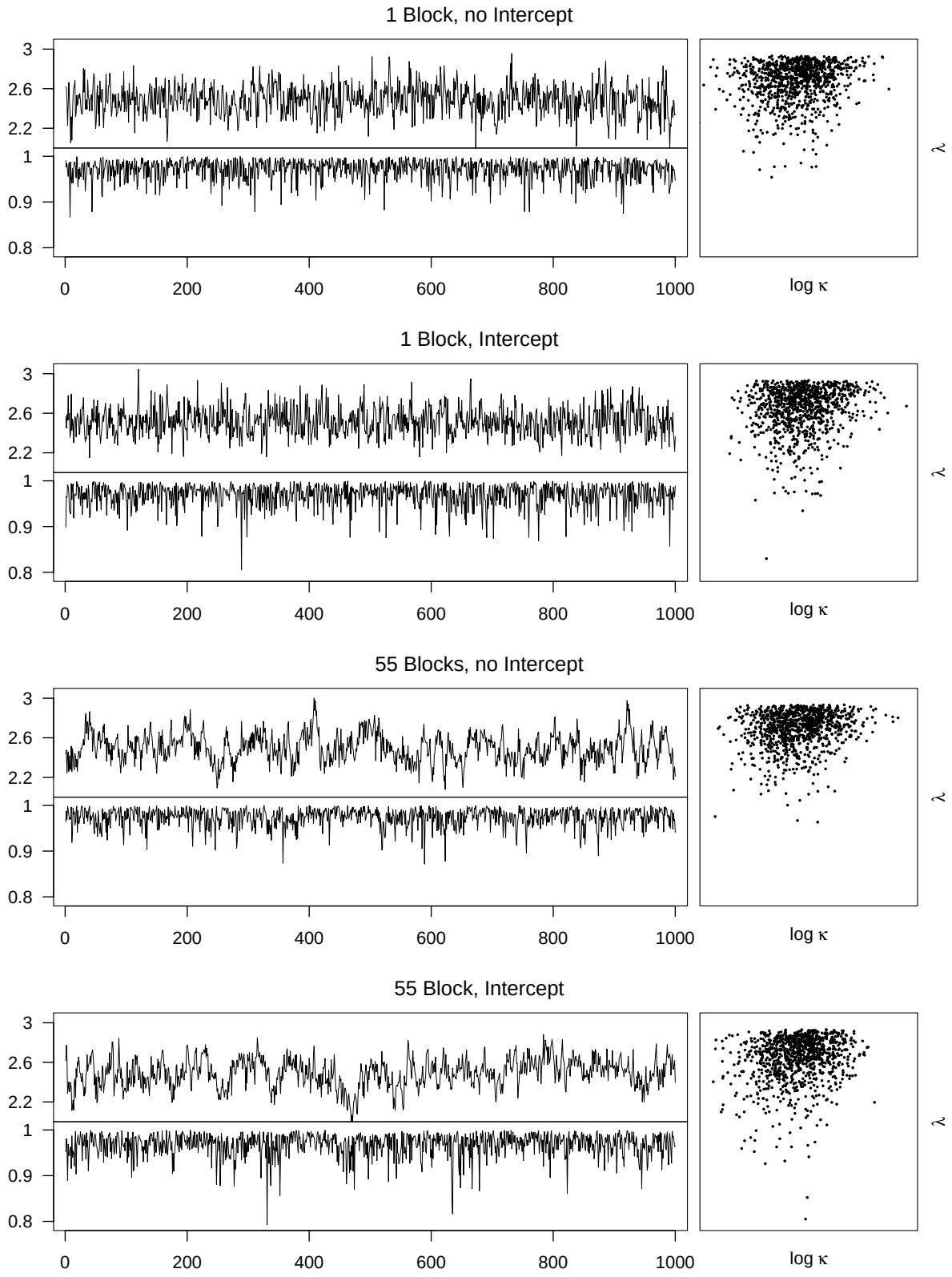

Figure 12: Diagnostic plots for the first 1,000 post burn-in samples from the four different Leroux implementations are shown. Each panel consists of trace plots for $\log \kappa$ (upper) and λ (lower), respectively. The mixing of $\log \kappa$ and λ is shown in a scatter plot (right). The chains were already thinned with a factor of 20.

	Implementation	Mean	SE	TS SE	MC SE	2.5%	50%	97.5%
κ	1 Block, no Intercept	12.13	0.0182	0.0229	0.0256	9.00	11.99	16.16
	1 Block, Intercept	12.47	0.0186	0.0243	0.0204	9.32	12.30	16.57
	55 Blocks, no Intercept	12.41	0.0184	0.0570	0.0643	9.22	12.26	16.46
	55 Blocks, Intercept	12.25	0.0179	0.0585	0.0636	9.18	12.12	16.08
λ	1 Block, no Intercept	0.972	0.00021	0.00022	0.00022	0.918	0.977	0.998
	1 Block, Intercept	0.969	0.00024	0.00025	0.00026	0.908	0.974	0.997
	55 Blocks, no Intercept	0.972	0.00021	0.00031	0.00030	0.921	0.977	0.998
	55 Blocks, Intercept	0.968	0.00024	0.00034	0.00034	0.909	0.973	0.997

Table 2: Summary table for the estimates of κ and λ generated with the four different implementations. The estimates are calculated based on 10,000 samples of one chain (generated with 300,000 MCMC iterations in total). Besides the standard error (SE), the time-series standard error (TS SE) derived with the R-package **coda** and a Monte Carlo standard error (MC SE) based on 100 replications of the simulation are given.

blocks, show higher auto correlations. This is consistent with the findings of Knorr-Held and Rue (2002) for the BYM model. The scatter plots of $\log \kappa$ and λ in the same figure show no suspicious patterns.

A numerical summary of the posterior distribution of κ and λ for the first of the 100 runs is given in Table 2. For the posterior mean the naive standard error (SE), the time-series standard error (TS SE) derived with the R-package **coda** and a Monte Carlo standard error (MC SE) based on 100 replications of each simulation are given. With respect to the TE SE, the variation in the mean estimates seems to be large. Thus, for example, the posterior mean of λ for the implementation with an intercept are lower than those for the implementations without intercept.

In Figure 13, ten different κ chains from the variation "55 Blocks, Intercept" are compared against ten κ chains from the other sampler versions. The diagnostic functions `gelman.plot()` and `geweke.diag()` from R-package **coda** were used. In the Gelman plots, each line represents the median shrink factor of a comparison of two chains. For the black line the 95% quantile is also drawn as dashed line. In the Geweke plot, ten pairs of two chains were compared by appending them to one single chain and setting `frac1` and `frac2` in the corresponding R function to 0.5. Although, we analyze long chains, the variability within and between each implementation is striking.

An overview of the 100 mean estimates for κ and λ for all variations (including those with varying acceptance probability and **CARBayes**) is given in the upper panels of Figure 14. It is reassuring that the **CARBayes** version has a large overlap with the corresponding "55 Blocks, Intercept" version. The varying acceptance probability seems to have little effect. The estimates for κ from the variations "1 Block, Intercept" and "1 Block, no Intercept" have little overlap with the other versions and none with each other. For the λ estimates, again, the pattern of higher values for implementations without intercept are visible. A reason for this feature is that some variance of $\boldsymbol{\eta}$ is absorbed by the intercept and thus lacks the precision κ of the corresponding random effect $\mathbf{u}$. The same patterns are visible in Figure 14 (bottom), where the ECDF's of κ and λ for 400 chains are drawn. A separation of the versions with and without the intercept is clearly visible. Again the variation within each of the four implementations is considerable.

Finally, we applied a hierarchical clustering to the 100 ECDFs of the four variations. The results for the parameters κ, λ, and four randomly selected regions $\eta_1, \ldots, \eta_4$ are shown in Figure 15.

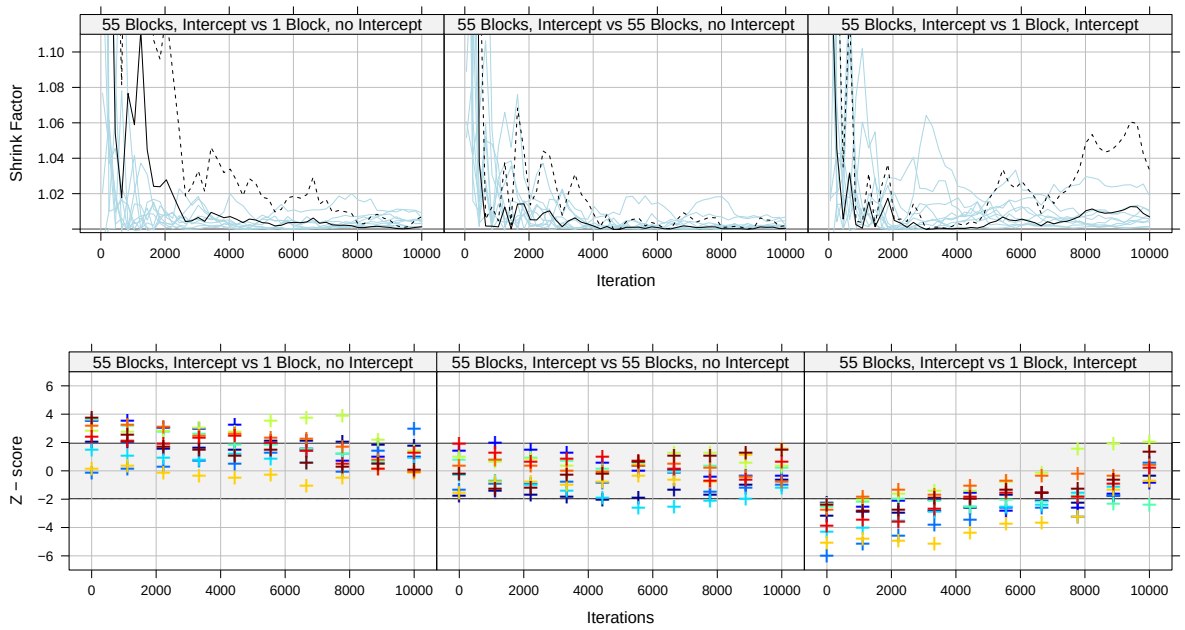

Figure 13: Comparison of 10 different κ chains from the variation "55 Blocks, Intercept" are compared against 10 κ chains from the other variations. The diagnostic functions are implemented in the R-package **coda**. Upper panels: Gelman plots with the median shrink factor for 10 comparisons (solid lines) and the 95% quantile for the first comparison (dashed line). Lower panels: Geweke plots with one color for each comparison. The two chains were compared by appending them to one single chain and setting `frac1` and `frac2` in the `geweke.diag()` function to 0.5.

For κ and λ, the same patterns (already mentioned above) are confirmed. For the η parameters no clear patterns resulted. This indicates that the spatial fields of all versions are similar, or more precisely, that the within implementation variance is larger than the between implementation variance. An exception is η_3 from the "1 Block, Intercept" version, which seems rather separated from the other versions.

3.5. Discussion, extensions, pitfalls

Naturally, many of the comments in Section 2.5 apply here as well, and we only mention a few relevant or new points. The simulation of 100 chains per version with a length of 300,000 each was only feasible within a reasonable amount of time due to a parallel implementation on a computer with several nodes. The R-package **snowfall** (Knaus 2013) greatly simplified the simultaneous generation of several chains with different seed values for the random number generator. One chain took about one hour on a single processor for the "1 Block" versions and about three hours for the versions with a "55 Block" update of **u**. This corresponds to 83 and 28 iterations per second, respectively. The difference in speed results mainly from the faster sparse matrix algebra Fortran back-end, which was accessed through the R-package **spam** for the "1 Block" versions. Some optimization of the R code with respect to speed would be possible. However, this would reduce the readability of the code. Implementing parts of the code in C++ would reduce calculation time and is planed for the **CARBayes** package (Lee 2013).

As was pointed out in the previous Section, the chains for the versions with intercept are different

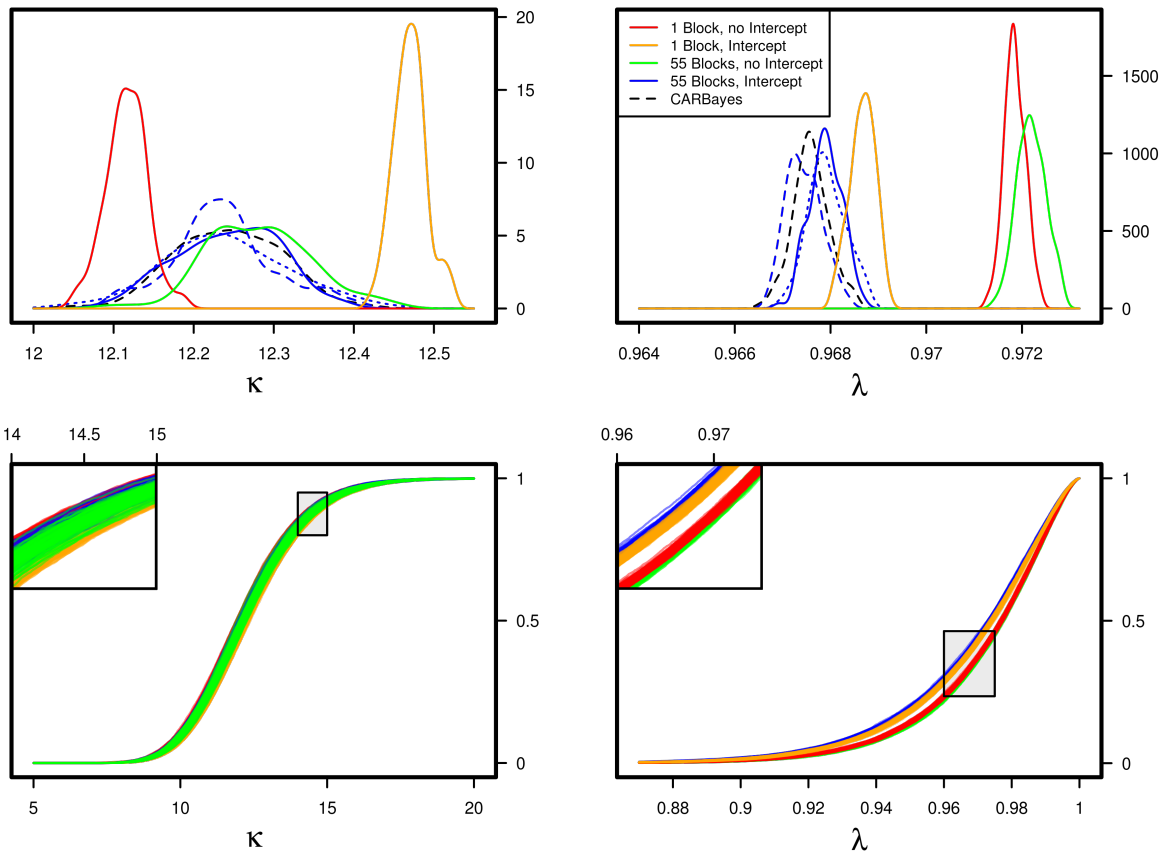

Figure 14: Upper panels: Densities of the mean estimates for κ and λ of 100 repeated simulations. The lines are either solid for an acceptance rate of $\approx 40\%$, dotted for an acceptance rate of $\approx 70\%$, or dashed for an acceptance rate similar to the **CARBayes** implementation. Lower panels: 100 ECDFs for κ and λ are drawn for the 4 different implementations with an acceptance rate of $\approx 40\%$. Each simulation considers 10,000 samples, which are taken from a MCMC run with 300,000 iterations.

from the chains without (especially the λ chains). One possible explanation is that the normal prior of the intercept with variance 10^{10} was informative enough to lead to the higher λ values for versions without intercept. Another reason for this feature might be that some variance of $\boldsymbol{\eta}$ is absorbed by the intercept and thus reduces the strength of the unstructured component of the corresponding random effect $\mathbf{u}$. Finally, it could also be an artifact of the mean centering-on-the-fly approach, which possibly has an effect on the equilibrium distribution of $\mathbf{u}$. A way to overcome this issue is to use `rmvnorm.const()` from the R-package **spam**.

To our knowledge there are no further implementations of the Leroux model readily available. Also, an extension of the model to the multivariate case with more than one diseases or a spatio-temporal model with a Leroux type of random effect would be interesting.

One possible extension of the model is "ecological regression", where additional covariates are taken into account. The R-package **CARBayes** is capable of fitting such models, and a corresponding model description can be specified though the formula interface.

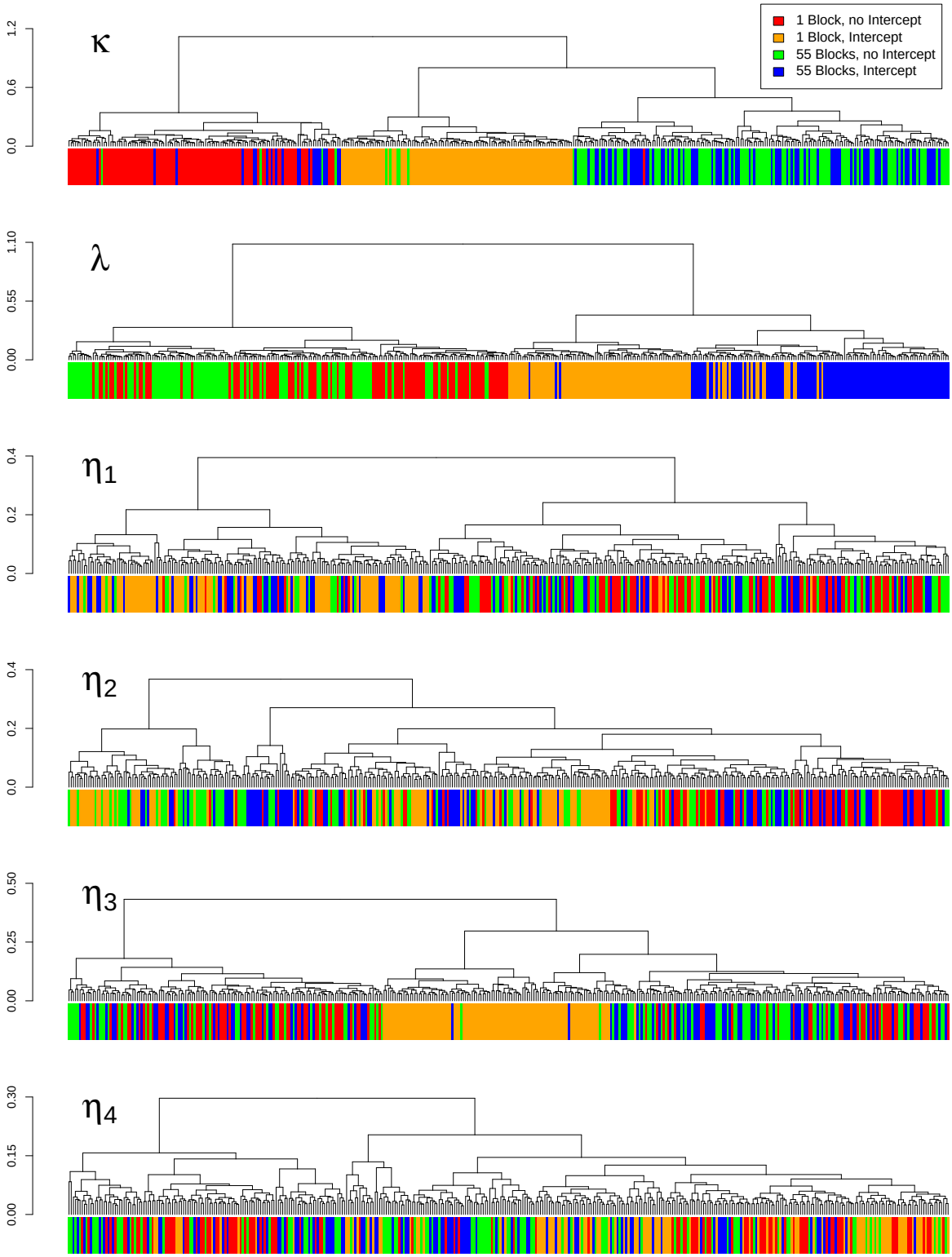

Figure 15: For the parameters κ, λ, and four randomly chosen elements of $\boldsymbol{\eta}$ dendrograms from a hierarchical clustering of the posterior ECDFs are shown. The clustering was applied to chains from the four implementation (indicated with colors) with 100 replications each. The figure is based on the same samples as Figure 14.

4. Final thoughts and remarks

Comparing several, long MCMC chains leads to the analysis of a huge number of data points. To summarize the information, several traditional methods, such as summary tables, Gelman plots, and Geweke plots, are useful (Tables 1, 2, Figures 4 and 13). In addition, we proposed a hierarchical clustering of ECDFs (Figure 7, 10 and 15). Our impression is that this approach is useful to visualize the (dis)similarity of several implementations, especially if there are many replicates of chains available. Other approaches like principal component and canonical correlation analysis (Mardia *et al.* 1979) did not lead to additional insights and are not shown in the paper. It would be interesting to compare the chains in a more formal ANOVA-like framework, which would lead to quantitative statements.

Our simulations suggest that the variability of the estimates derived from MCMC chains is considerable. One reason may be that the chains are too short (the analyzed chains had a length of 10,000 and were a sub-sample from an MCMC run with 300,000 iterations). It is the authors' impression that a decade ago it was "common" to run many chains of huge lengths ($> 10^6$) for relatively simple models (of few parameters). With rising computing power Bayesian models have become more and more complex, but the number of chains and chain lengths have not been increased; rather they have been severely decreased. It would be a very interesting bibliographic review to study the evolution of the number of parameters, the number of chains, and the chain lengths in articles published in statistical journals or in general scientific journals. This should be contrasted with the number of parameters of the BHMs and the available computing power.

Moreover, the variability between different implementations of the same model seems to be larger than the variability within one implementation and could even be systematic (compare to the λ parameter of the Leroux model for implementations with and without intercept, for instance). It would be interesting to find a formal explanation of these features in order to better understand the models and implementations. Further, an incorrectly calculated acceptance probability may lead to stable and unsuspicious chains, and it can be very difficult to detect such errors. But do such small differences matter in practice? We think that the following recommendations help identify differences that *do* matter: (1) many and long chains should be simulated, (2) if possible, different implementations should be used, (3) formal and graphical comparisons should be made.

Finally, the choice of the software environments used in this article was driven by our experiences. However, there exist more sampling engines, e.g., BayesX (Brezger *et al.* 2005) or ADMB (Fournier *et al.* 2012), with R interfaces **BayesX** (Kneib *et al.* 2011) and **R2admb** (Bolker and Skaug 2012), or the package **geoRglm** (Christensen and Ribeiro 2002), that can handle BHMs with spatially correlated random effects. Different flavors of spatial models can also be handled with the jags engine (Plummer 2012) or with **spBayes** (Finley and Banerjee 2013). The community is extremely active, as indicated by the CRAN task view `Bayesian Inference`.

Acknowledgments

We thank the Editorial team, as well as the two reviewers for their constructive and valuable comments that led to a significant improvement in the paper. Further, we thank Håvard Rue for support on the **INLA** package. We would also like to thank the Swiss National Science Foundation (SNSF-143282) and the University Research Priority Program for Global Change and Biodiversity of the University of Zurich.

References

Besag J, York J, Mollié A (1991). "Bayesian Image Restoration, with two Applications in Spatial Statistics." *The Annals of the Institute of Statistical Mathematics*, **43**(1), 1–20. doi:10.1007/BF00116466.

Bivand RS, Pebesma EJ, Rubio VG (2008). *Applied Spatial Data Analysis with R*. Use R! Springer-Verlag. ISBN 9780387781716. URL http://www.asdar-book.org.

Bivand RS, Pebesma EJ, Rubio VG (2013). *Applied Spatial Data Analysis with R*. Use R!, second edition. Springer-Verlag. ISBN 9781461476177. URL http://www.asdar-book.org.

Bolker B, Skaug H (2012). ***R2admb**: ADMB to RInterface Functions*. R package version 0.7.5.3, URL http://CRAN.R-project.org/package=R2admb.

Brezger A, Kneib T, Lang S (2005). "BayesX: Analyzing Bayesian Structural Additive Regression Models." *Journal of Statistical Software*, **14**(11), 1–22. URL http://www.jstatsoft.org/v14/i11.

Brooks SP, Gelman A (1998). "General Methods for Monitoring Convergence of Iterative Simulations." *Journal of Computational and Graphical Statistics*, **7**(4), 434–455. doi:10.1080/10618600.1998.10474787.

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

R Core Team (2012). *R: A Language and Environment for Statistical Computing*. RFoundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org.

Dhar SS, Chakraborty B, Chaudhuri P (2011). "Comparison of Multivariate Distributions Using Quantile-Quantile Plots and Related Tests." Unpublished manuscript, URL http://www.isical.ac.in/~statmath/html/publication/unblinded.pdf.

Finley AO, Banerjee S (2013). ***spBayes**: Univariate and Multivariate Spatial Modeling*. R package version 0.3-1, URL http://CRAN.R-project.org/package=spBayes.

Fournier DA, Skaug HJ, Ancheta J, Ianelli J, Magnusson A, Maunder MN, Nielsen A, Sibert J (2012). "AD Model Builder: Using Automatic Differentiation for Statistical Inference of Highly Parameterized Complex Nonlinear Models." *Optimization Methods and Software*, **27**, 233–249. doi:10.1080/10556788.2011.597854.

Furrer R (2013). *spam: SPArse Matrix*. R package version 0.41-0, URL http://stat.ethz.ch/CRAN/web/packages/spam/index.html.

Furrer R, Sain SR (2010). "**spam**: A Sparse Matrix R Package with Emphasis on MCMC Methods for Gaussian Markov Random Fields." *Journal of Statistical Software*, **36**(10), 1–25. URL http://www.jstatsoft.org/v36/i10/.

Gelfand A, Sahu S, Carlin B (1995). "Efficient Parametrization for Normal Linear Mixed Effects Models." *Biometrika*, **82**, 479–488. URL http://www.jstor.org/stable/2337527.

Gelman A, Rubin DB (1992). "Inference from iterative simulation using multiple sequences." *Statistical Science*, **7**, 457–511.

Gerber F (2013). *Disease Mapping with the Besag-York-Mollié Model Applied to a Cancer and a Worm Infections Dataset*. Master's thesis, University of Zurich, Schwitzerland.

Geweke J (1992). "Evaluating the accuracy of sampling-based approaches to calculating posterior moments." In JM Bernardo, J Berger, AP Dawid, JFM Smith (eds.), *Bayesian Statistics 4*, pp. 169–193. Oxford University Press, Oxford.

Held L, Natário I, Fenton SES, Rue H, Becker N (2005). "Towards Joint Disease Mapping." *Statistical Methods in Medical Research*, **14**(1), 61–82. doi:10.1191/0962280205sm389oa.

Knaus J (2013). ***snowfall**: Easier Cluster Computing (based on snow)*. R package version 1.84-4, URL http://CRAN.R-project.org/package=snowfall.

Kneib T, Heinzl F, Brezger A, Bove DS (2011). ***BayesX**: R Utilities Accompanying the Software Package **BayesX***. R package version 0.2-5, URL http://CRAN.R-project.org/package=BayesX.

Knorr-Held L, Best NG (2001). "A Shared Component Model for Detecting Joint and Selective Clustering of two Diseases." *Journal of the Royal Statistical Society A*, **164**(1), 73–85. doi:10.1111/1467-985X.00187.

Knorr-Held L, Rue H (2002). "On Block Updating in Markov Random Field Models for Disease Mapping." *Scandinavian Journal of Statistics*, **29**, 597–614. doi:10.1111/1467-9469.00308.

Lee D (2011). "A Comparison of Conditional Autoregressive Models used in Bayesian Disease Mapping." *Spatial and Spatio-temporal Epidemiology*, **2**(2), 79–89. doi:10.1016/j.sste.2011.03.001.

Lee D (2013). "CARBayes: An R Package for Bayesian Spatial Modeling with Conditional Autoregressive Priors." *Journal of Statistical Software*, **55**(13), 1–24. URL http://www.jstatsoft.org/v55/i13/.

Leroux B, Lei X, Breslow N (1999). *Estimation of Disease Rates in small Areas: A new Mixed Model for Spatial Dependence*. IMA volumes in mathematics and its applications. U.S. Government Printing Office. ISBN 9780387989242.

LeSage J, Pace RK (2009). *Introduction to Spatial Econometrics*. Chapman and Hall/CRC.

Li B, Smerdon JE (2012). "Defining Spatial Comparison metrics for Evaluation of Paleoclimatic Field Reconstructions of the Common Era." *Environmetrics*, **23**(5), 394–406. doi:10.1002/env.2142.

Lunn D, Jackson C, Best N, Thomas A, Spiegelhalter D (2013). *The BUGS Book: A Practical Introduction to Bayesian Analysis*. Texts in Statistical Science. Chapman and Hall/CRC. ISBN 978-1-58488-849-9. URL http://www.mrc-bsu.cam.ac.uk/bugs/thebugsbook/.

MacNab YC (2010). "On Bayesian Shared Component Disease Mapping and Ecological Regression with Errors in Covariates." *Statistics in Medicine*, **29**(11), 1239–49. doi:10.1002/sim.3875.

Mardia KV, Kent JT, Bibby JM (1979). *Multivariate Analysis*. Academic Press, London.

Mollié A (1996). "Bayesian Mapping of Disease." In WR Gilks, S Richardson, DJ Spiegelhalter (eds.), *Markov Chain Monte Carlo in Practice*, pp. 359–379. Chapman & Hall, London.

Novomestky F, Nadarajah S (2012). *truncdist: Truncated Random Variables*. R package version 1.0-1, URL http://CRAN.R-project.org/package=truncdist.

Plummer M (2012). ***rjags**: Bayesian Graphical Models using MCMC*. R package version 3-7, URL http://CRAN.R-project.org/package=rjags.

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

Roberts G, Rosenthal J (2001). "Optimal Scaling for Various Metropolis-Hastings Algorithms." *Statistical Science*, **16**, 351–367. doi:10.1214/ss/1015346320.

Rosenbaum PR (2005). "An Exact Distribution-free Test Comparing Two Multivariate Distributions Based on Adjacency." *Journal of the Royal Statistical Society B*, **67**(4), 515–530. doi:10.1111/j.1467-9868.2005.00513.x.

Rue H, Held L (2005). *Gaussian Markov Random Fields: Theory and Applications*, volume 104 of *Monographs on Statistics and Applied Probability*. Chapman & Hall, London. ISBN 9781584884323.

Rue H, Martino S, Chopin N (2009a). "Approximate Bayesian Inference for Latent Gaussian Models by using Integrated Nested Laplace Approximations." *Journal of the Royal Statistical Society B*, **71**(2), 319–392. doi:10.1111/j.1467-9868.2008.00700.x.

Rue H, Martino S, Lindgren F, Simpson D, Riebler A (2009b). *INLA: Functions which allow to perform full Bayesian analysis of latent Gaussian models using Integrated Nested Laplace Approximaxion*. URL www.r-inla.org.

Schrödle B, Held L (2010). "A Primer on Disease Mapping and Ecological Regression using **INLA**." *Computational Statistics*, **26**(2), 241–258. doi:10.1007/s00180-010-0208-2.

Schrödle B, Held L, Riebler A, Danuser J (2011). "Using Integrated Nested Laplace Approximations for the Evaluation of Veterinary Surveillance Data from Switzerland: A Case-study." *Journal of the Royal Statistical Society C*, **60**(2), 261–279. doi:10.1111/j.1467-9876.2010.00740.x.

Sturtz S, Ligges U, Gelman A (2005). "**R2WinBUGS**: A Package for Running WinBUGS from R." *Journal of Statistical Software*, **12**(3), 1–16. URL http://www.jstatsoft.org/v12/i03.

Sun Y, Genton MG, Nychka D (2012). "Exact Fast Computation of Band Depth for Large Functional Datasets: How Quickly Can One Million Curves Be Ranked?" *Stat*, pp. 68–74. doi:10.1002/sta4.8.

Waller L, Carlin B (2010). "Disease Mapping." In AE Gelfand, PJ Diggle, M Fuentes, P Guttorp (eds.), *Handbook of Spatial Statistics*. Taylor & Francis Group. Chapter 14.

Wigley T, Santer B (1990). "Statistical Comparison of Spatial Fields in Model Validation, Perturbation, and Predictability Experiments." *Journal of Geophysical Research*, **95**, 851–865. doi:10.1029/JD095iD01p00851.

Affiliation:

Reinhard Furrer
Institute of Mathematics
University of Zurich
8057 Zurich, Switzerland
E-mail: reinhard.furrer@math.uzh.ch
URL: http://www.math.uzh.ch/furrer

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