Computer Intensive Statistical Methods - Exercise 2

Martin Outzen Berild, Sindre Henriksen

March 2019

The goal of this exercise is to carry out a spatial analysis on mortality rates of oral cavity cancer in males in Germany during a 5-year period, 1986–1990, for $n = 544$ districts. Observed counts are denoted y_i, expected counts E_i.

We assume observed counts to be conditionally independent Poisson. Thus the model is

$$y_i \,|\, \eta_i \sim \text{Pois}(E_i e^{\eta_i}), \quad i = 1, \dots, n\,,$$

where $\boldsymbol{\eta} = (\eta_1, \dots, \eta_n)^\intercal$ is the log relative risk. $\boldsymbol{\eta}$ is further decomposed into

$$\boldsymbol{\eta} = \boldsymbol{u} + \boldsymbol{v}\,.$$

$\boldsymbol{u} = (u_1, \dots, u_n)^\intercal$ is spatially structured with smoothing parameter κ_u. $\boldsymbol{v} = (v_1, \dots, v_n)^\intercal$ is unstructured white noise with precision parameter κ_v, i.e. $\boldsymbol{v} \sim \mathcal{N}(\boldsymbol{0}, \kappa_v^{-1} I)$.

Next we introduce a spatially correlated effect is by assuming that neighbouring districts are more similar than distant districts. For this purpose, a neighbourhood has to be defined for each district. We assume that two districts are neighbours if they share a common border. If we consider a single district, and condition on only the neighbours with which it shares a border, this is a first-order autoregressive process, or intrinsic Gaussian Markov random field with density

$$\begin{aligned} \text{p}(\boldsymbol{u} \,|\, \kappa_u) &\propto \kappa_u^{(n-1)/2} \exp\left\{ -\frac{\kappa_u}{2} \sum_{i \sim j} (u_i - u_j)^2 \right\} \\ &= \kappa_u^{(n-1)/2} e^{-\kappa_u \boldsymbol{u}^\intercal R \boldsymbol{u}/2}\,, \end{aligned} \tag{1}$$

i.e.

$$\boldsymbol{u} \,|\, \kappa_u \sim \mathcal{N}(\boldsymbol{0}, (\kappa_u R)^{-1})\,.$$

The sum in (1) goes over all pairs of neighbouring districts $i \sim j$ which is defined by the geographical map. The neighbourhood structure is consequently defined in the structure matrix R where

$$R_{ij} = \begin{cases} n_i, & i = j \\ -1, & i \sim j \\ 0, & \text{otherwise} \end{cases}\,.$$

Here, η_i denotes the number of neighbouring districts of district i. The distribution of $\boldsymbol{\eta}$, conditional on the spatial component u and κ_v, is

$$\boldsymbol{\eta} \,|\, \boldsymbol{u}, \kappa_v \sim \mathcal{N}(\boldsymbol{u}, \kappa_v^{-1} I)\,.$$

The precision terms κ_u and κ_v are assigned gamma prior distributions

$$\begin{aligned} \kappa_u &\sim \mathrm{Gamma}(\alpha_u, \beta_u)\,, \\ \kappa_v &\sim \mathrm{Gamma}(\alpha_v, \beta_v)\,, \end{aligned}$$

where $\mathrm{Gamma}(\alpha, \beta)$ denotes the gamma density with shape parameter α and rate parameter β. Here, we will use $\alpha_u = \alpha_v = 1$ and $\beta_u = \beta_v = 0.01$.

Exercise 1

(a) For the posterior distribution $\mathrm{p}(\boldsymbol{\eta}, \boldsymbol{u}, \kappa_u, \kappa_v \,|\, \boldsymbol{y})$ we have

$$\begin{aligned} \mathrm{p}(\boldsymbol{\eta}, \boldsymbol{u}, \kappa_u, \kappa_v \,|\, \boldsymbol{y}) &\propto \mathrm{p}(\boldsymbol{y}, \boldsymbol{\eta}, \boldsymbol{u}, \kappa_u, \kappa_v) \\ &\propto \mathrm{p}(\boldsymbol{y} \,|\, \boldsymbol{\eta}, \boldsymbol{u}, \kappa_u, \kappa_v)\mathrm{p}(\boldsymbol{\eta} \,|\, \boldsymbol{u}, \kappa_u, \kappa_v)\mathrm{p}(\boldsymbol{u} \,|\, \kappa_u, \kappa_v)\mathrm{p}(\kappa_u \,|\, \kappa_v)\mathrm{p}(\kappa_v) \\ &= \mathrm{p}(\boldsymbol{y} \,|\, \boldsymbol{\eta})\mathrm{p}(\boldsymbol{\eta} \,|\, \boldsymbol{u}, \kappa_v)\mathrm{p}(\boldsymbol{u} \,|\, \kappa_u)\mathrm{p}(\kappa_u)\mathrm{p}(\kappa_v)\,. \end{aligned}$$

We have that

$$\begin{aligned} \mathrm{p}(\boldsymbol{y} \,|\, \boldsymbol{\eta}) &= \prod_{i=1}^{n} \frac{(E_i e^{\eta_i})^{y_i} \exp\{-E_i e^{\eta_i}\}}{y_i!} \\ &= \exp\left\{\sum_{i=1}^{n} \eta_i y_i - E_i e^{\eta_i}\right\} \cdot \prod_{i=1}^{n} \frac{E_i^{y_i}}{y_i!} \\ \mathrm{p}(\boldsymbol{\eta} \,|\, \boldsymbol{u}, \kappa_v) &\propto \kappa_v^{n/2} \exp\left\{-\frac{1}{2}\kappa_v(\boldsymbol{\eta} - \boldsymbol{u})^\intercal(\boldsymbol{\eta} - \boldsymbol{u})\right\} \qquad (2) \\ \mathrm{p}(\boldsymbol{u} \,|\, \kappa_u) &\propto \kappa_u^{(n-1)/2} \exp\left\{-\frac{1}{2}\kappa_u \boldsymbol{u}^\intercal R \boldsymbol{u}\right\} \\ \mathrm{p}(\kappa_u) &\propto \kappa_u^{\alpha_u - 1} e^{-\beta_u \kappa_u} \\ \mathrm{p}(\kappa_v) &\propto \kappa_v^{\alpha_v - 1} e^{-\beta_v \kappa_v}\,. \end{aligned}$$

Thus

$$\begin{aligned} \mathrm{p}(\boldsymbol{\eta}, \boldsymbol{u}, \kappa_u, \kappa_v \,|\, \boldsymbol{y}) \propto \kappa_u^{(n-1)/2 + \alpha_u - 1} \kappa_v^{n/2 + \alpha_v - 1} \exp\Bigg\{ &- \beta_u \kappa_u - \beta_v \kappa_v \\ &- \frac{\kappa_u}{2}\boldsymbol{u}^\intercal R \boldsymbol{u} - \frac{\kappa_v}{2}(\boldsymbol{\eta} - \boldsymbol{u})^\intercal(\boldsymbol{\eta} - \boldsymbol{u}) + \sum_{i=1}^{n} \eta_i y_i - E_i e^{\eta_i}\Bigg\}\,. \end{aligned}$$

(b) Let $\tilde{f}(\eta_i)$ be the second order Taylor approximation of

$$f(\eta_i) = y_i \eta_i - E_i e^{\eta_i}$$

at a points z_i. Then

$$\begin{aligned}
\tilde{f}(\eta_i) &= f(z_i) + f'(z_i)(\eta_i - z_i) + \frac{1}{2}f''(z_i)(\eta_i - z_i)^2 \\
&= y_i z_i - E_i e^{z_i} + (y_i - E_i e^{z_i})(\eta_i - z_i) - \frac{1}{2}E_i e^{z_i}(\eta_i - z_i)^2 \\
&= E_i e^{z_i}(z_i - \frac{1}{2}z_i^2 - 1) + (y_i + E_i e^{z_i}[z_i - 1])\eta_i - \frac{1}{2}E_i e^{z_i}\eta_i^2 \\
&= a_i + b_i\eta_i - \frac{1}{2}c_i\eta_i^2 \,,
\end{aligned}$$

where $a_i = E_i e^{z_i}(z_i - \frac{1}{2}z_i^2 - 1)$, $b_i = y_i + E_i e^{z_i}(z_i - 1)$ and $c_i = E_i e^{z_i}$.

(c) The full conditional densities of κ_u and κ_v are found by using Bayes' rule and the conditionals in (2). They are given by

$$\begin{aligned}
\mathrm{p}(\kappa_u \,|\, \boldsymbol{y}, \boldsymbol{\eta}, \boldsymbol{u}, \kappa_v) &= \mathrm{p}(\kappa_u \,|\, \boldsymbol{u}) \\
&\propto \mathrm{p}(\boldsymbol{y} \,|\, \boldsymbol{\eta})\mathrm{p}(\boldsymbol{\eta} \,|\, \boldsymbol{u}, \kappa_v)\mathrm{p}(\boldsymbol{u} \,|\, \kappa_u)\mathrm{p}(\kappa_u)\mathrm{p}(\kappa_v) \\
&\propto \mathrm{p}(\boldsymbol{u} \,|\, \kappa_u)\mathrm{p}(\kappa_u) \\
&\propto \kappa_u^{(n-1)/2+\alpha_u-1} \exp\left\{-\kappa_u\left(\frac{1}{2}\boldsymbol{u}^\intercal R\boldsymbol{u} + \beta_u\right)\right\}
\end{aligned}$$

and

$$\begin{aligned}
\mathrm{p}(\kappa_v \,|\, \boldsymbol{y}, \boldsymbol{\eta}, \boldsymbol{u}, \kappa_u) &= \mathrm{p}(\kappa_v \,|\, \boldsymbol{\eta}, \boldsymbol{u}) \\
&\propto \mathrm{p}(\boldsymbol{y} \,|\, \boldsymbol{\eta})\mathrm{p}(\boldsymbol{\eta} \,|\, \boldsymbol{u}, \kappa_v)\mathrm{p}(\boldsymbol{u} \,|\, \kappa_u)\mathrm{p}(\kappa_u)\mathrm{p}(\kappa_v) \\
&\propto \mathrm{p}(\boldsymbol{\eta} \,|\, \boldsymbol{u}, \kappa_v)\mathrm{p}(\kappa_v) \\
&\propto \kappa_u^{n/2+\alpha_v-1} \exp\left\{-\kappa_v\left[\frac{1}{2}(\boldsymbol{\eta} - \boldsymbol{u})^\intercal(\boldsymbol{\eta} - \boldsymbol{u}) + \beta_v\right]\right\} .
\end{aligned}$$

Thus κ_u and κ_v are Gamma distributed with parameters given in the following:

$$\kappa_u \,|\, \boldsymbol{u} \sim \mathrm{Gamma}\left(\frac{n-1}{2} + \alpha_u, \frac{1}{2}\boldsymbol{u}^\intercal R\boldsymbol{u} + \beta_u\right) \tag{3}$$

and

$$\kappa_v \,|\, \boldsymbol{\eta}, \boldsymbol{u} \sim \mathrm{Gamma}\left(\frac{n}{2} + \alpha_v, \frac{1}{2}(\boldsymbol{\eta} - \boldsymbol{u})^\intercal(\boldsymbol{\eta} - \boldsymbol{u}) + \beta_v\right) . \tag{4}$$

The full conditional posterior density $\mathrm{p}(\boldsymbol{u} \,|\, \boldsymbol{y}, \boldsymbol{\eta}, \kappa_u, \kappa_v)$ is given by

$$\begin{aligned}
\mathrm{p}(\boldsymbol{u} \,|\, \boldsymbol{y}, \boldsymbol{\eta}, \kappa_u, \kappa_v) &= \mathrm{p}(\boldsymbol{u} \,|\, \boldsymbol{\eta}, \kappa_u, \kappa_v) \\
&\propto \mathrm{p}(\boldsymbol{y} \,|\, \boldsymbol{\eta})\mathrm{p}(\boldsymbol{\eta} \,|\, \boldsymbol{u}, \kappa_v)\mathrm{p}(\boldsymbol{u} \,|\, \kappa_u)\mathrm{p}(\kappa_u)\mathrm{p}(\kappa_v) \\
&\propto \mathrm{p}(\boldsymbol{\eta} \,|\, \boldsymbol{u}, \kappa_v)\mathrm{p}(\boldsymbol{u} \,|\, \kappa_u) \\
&\propto \exp\left\{-\frac{1}{2}\boldsymbol{u}^\intercal(\kappa_v I + \kappa_u R)\boldsymbol{u} + \boldsymbol{u}^\intercal(\kappa_v\boldsymbol{\eta})\right\} .
\end{aligned}$$

This is the canonical form of a normal distribution with parameters given in the following:

$$\boldsymbol{u} \,|\, \boldsymbol{\eta}, \kappa_u, \kappa_v \sim \mathcal{N}((\kappa_v I + \kappa_u R)^{-1}\kappa_v \boldsymbol{\eta}, (\kappa_v I + \kappa_u R)^{-1})\,. \tag{5}$$

Finally, the full conditional posterior density $\mathrm{p}(\boldsymbol{\eta} \,|\, \boldsymbol{y}, \boldsymbol{u}, \kappa_u, \kappa_v)$ is given by

$$\begin{aligned}
\mathrm{p}(\boldsymbol{\eta} \,|\, \boldsymbol{y}, \boldsymbol{u}, \kappa_u, \kappa_v) &= \mathrm{p}(\boldsymbol{\eta} \,|\, \boldsymbol{y}, \boldsymbol{u}, \kappa_v) \\
&\propto \mathrm{p}(\boldsymbol{y} \,|\, \boldsymbol{\eta})\mathrm{p}(\boldsymbol{\eta} \,|\, \boldsymbol{u}, \kappa_v)\mathrm{p}(\boldsymbol{u} \,|\, \kappa_u)\mathrm{p}(\kappa_u)\mathrm{p}(\kappa_v) \\
&\propto \mathrm{p}(\boldsymbol{y} \,|\, \boldsymbol{\eta})\mathrm{p}(\boldsymbol{\eta} \,|\, \boldsymbol{u}, \kappa_v) \\
&\propto \exp\left\{ \sum_{i=1}^{n} (y_i \eta_i - E_i e^{\eta_i}) - \frac{\kappa_v}{2}(\boldsymbol{\eta} - \boldsymbol{u})^\intercal (\boldsymbol{\eta} - \boldsymbol{u}) \right\} \\
&\propto \exp\left\{ -\frac{1}{2}\boldsymbol{\eta}^\intercal(\kappa_v I)\boldsymbol{\eta} + \boldsymbol{\eta}^\intercal(\kappa_v \boldsymbol{u}) + f(\boldsymbol{\eta}) \right\}\,,
\end{aligned}$$

where $f(\boldsymbol{\eta}) = \sum_{i=1}^{n}(y_i\eta_i - E_i e^{\eta_i}) = \sum_{i=1}^{n} f(\eta_i)$. Approximating $f(\boldsymbol{\eta})$ by $\tilde{f}(\boldsymbol{\eta}) = \sum_{i=1}^{n} \tilde{f}(\eta_i) = \sum_{i=n}^{n}(a_i + b_i\eta_i - c_i\eta_i^2/2) = \sum_{i=1}^{n} a_i + \boldsymbol{\eta}^\intercal \boldsymbol{b} + \boldsymbol{\eta}^\intercal C \boldsymbol{\eta}/2$, where $\boldsymbol{b} = (b_1, \ldots, b_n)$ and C is the diagonal matrix with $\boldsymbol{c} = (c_1, \ldots, c_n)$ on the diagonal, yields the approximation of $\mathrm{p}(\boldsymbol{\eta} \,|\, \boldsymbol{y}, \boldsymbol{u}, \kappa_v)$

$$\mathrm{q}(\boldsymbol{\eta} \,|\, \boldsymbol{z}, \boldsymbol{y}, \boldsymbol{u}, \kappa_u, \kappa_v) = \mathrm{q}(\boldsymbol{\eta} \,|\, \boldsymbol{z}, \boldsymbol{y}, \boldsymbol{u}, \kappa_v) \tag{6}$$

$$\propto \exp\left\{ -\frac{1}{2}\boldsymbol{\eta}^\intercal(\kappa_v I + C)\boldsymbol{\eta} + \boldsymbol{\eta}^\intercal(\kappa_v \boldsymbol{u} + \boldsymbol{b}) \right\}\,, \tag{7}$$

where $\boldsymbol{z} = (z_1, \ldots, z_n)$, i.e. the vector at which the Taylor approximation is computed. (6) is the canonical form of the density of the Normal distribution

$$\mathcal{N}((\kappa_v I + C)^{-1}(\kappa_v \boldsymbol{u} + b), (\kappa_v I + C)^{-1}))\,.$$

Exercise 2

From the formulations in exercise 1, a MCMC sampler can be implemented. The data used in this work is the '*Oral*' dataset. The libraries used in exercise 2 is

```r
library(spam)
library(tidyverse)
library(fields, warn.conflicts = FALSE)
source("./data/ex2_additionalFiles/dmvnorm.R")
library(colorspace)
library(ggpubr)
```

Next we import the data and create a list *input*, which is used in all of the functions in this exercise. It contains α and β for the prior and full conditional posterior distribution for κ_u and κ_v, the neighbourhood structure R, observed counts $\boldsymbol{y}$ and expected counts $\boldsymbol{E}$.

```r
load("./data/ex2_additionalFiles/tma4300_ex2_Rmatrix.Rdata")
str(Oral)
attach(Oral)

# a list of all the input variables to make code more readable
input <- list(
  y = Oral$Y,
  E = Oral$E,
  n = length(Oral$Y),
  alpha = 1,
  beta = 0.01,
  R = R
  )
```

Then we are ready to start implementing samplers for the full conditionals. First we look at the full conditional of κ_u, given by $p(\kappa_u \,|\, \boldsymbol{y}, \kappa_v^{(m-1)}, \boldsymbol{\eta}^{(m-1)}, \boldsymbol{u}^{(m-1)})$. The distribution can be found in Equation (3) and is implemented in the function `r_kappa_u`.

```r
# Draw samples from the full condition of kappa_u
r_kappa_u <- function(input,u){
  shape = (input$n-1)/2 + input$alpha
  rate = 1/2* t(u) %*% input$R %*% u + input$beta
  return(rgamma(n = 1, shape = shape, rate = rate))
}
```

The full conditional of κ_v, given by $p(\kappa_v \,|\, \boldsymbol{y}, \kappa_u^m, \boldsymbol{\eta}^{(m-1)}, \boldsymbol{u}^{(m-1)})$, which has the distribution given in Equation (4). The samples of κ_v are generated in the function `r_kappa_v`.

```r
# Draw samples from the full condition of kappa_v
r_kappa_v <- function(input,eta,u){
  shape = input$n/2 + input$alpha
  rate = 1/2 * t(eta - u) %*% (eta - u) + input$beta
  return(rgamma(n = 1 ,shape = shape, rate = rate))
}
```

To draw samples from of $\boldsymbol{u}$ we use the the distribution of the full conditional, $p(\boldsymbol{u} \,|\, \boldsymbol{y}, \kappa_u^{(m)}, \kappa_v^{(m)}, \boldsymbol{\eta}^{(m-1)})$, given in Equation (5). The sampler is implemented in `r_u`.

```r
# draw samples from the full conditional of u
r_u <- function(input,kappa_u,kappa_v,eta){
  Q = diag.spam(kappa_v, input$n) + kappa_u*input$R
  b = kappa_v * eta
  return(c(rmvnorm.canonical(n = 1,  b = b, Q = Q)))
}
```

The proposal samples $\boldsymbol{\eta}^*$ is drawn from the approximation of the posterior full conditional for $\boldsymbol{\eta}$ found in Equation (6). In the proposal density $q(\boldsymbol{\eta}^* \mid \boldsymbol{z}, \boldsymbol{y}, \boldsymbol{u}^{(m)}, \kappa_u^{(m)}, \kappa_v^{(m)})$ we used the approximation around $\boldsymbol{z} = \boldsymbol{\eta}^{(m-1)}$. From this we implement the function, `r_eta_prop`, which generates a proposal sample.

```
# draw samples from the proposal density of eta
r_eta_prop <- function(input,z,u,kappa_v){
  c_vec = get_c(input,z)
  b_vec = get_b(input,z)
  b = kappa_v*u + b_vec
  Q = diag.spam(kappa_v, input$n) + diag.spam(c_vec)
  return(c(rmvnorm.canonical(n = 1, b = b, Q = Q)))
}
```

Whether or not the proposal $\boldsymbol{\eta}^*$ is accepted, is determined by the acceptance probability

$$\alpha = \min\left(1, \frac{p(\boldsymbol{\eta}^* \mid \boldsymbol{y}, \kappa_v^{(m)}, \kappa_u^{(m)}, \boldsymbol{u}^{(m)})}{p(\boldsymbol{\eta}^{(m-1)} \mid \boldsymbol{y}, \kappa_v^{(m)}, \kappa_u^{(m)}, \boldsymbol{u}^{(m)})} \frac{q(\boldsymbol{\eta}^{(m-1)} \mid \boldsymbol{\eta}^*, \boldsymbol{y}, \boldsymbol{u}^{(m)}, \kappa_u^{(m)}, \kappa_v^{(m)})}{q(\boldsymbol{\eta}^* \mid \boldsymbol{\eta}^{(m-1)}, \boldsymbol{y}, \boldsymbol{u}^{(m)}, \kappa_u^{(m)}, \kappa_v^{(m)})}\right).$$

If the proposal is accepted $\boldsymbol{\eta}^{(m)} = \boldsymbol{\eta}^*$, and if not $\boldsymbol{\eta}^{(m)} = \boldsymbol{\eta}^{(m-1)}$. To help with the calculation of the acceptance probability we create two functions, `d_eta_p` and `d_eta_q`, that return p-values of the posterior density $p(\eta \mid ...)$ and the proposal density $q(\eta \mid ....)$. All these probability calculations are done in log scale.

```
# finding the p-value of eta from the
# full condition p(eta|...)
d_eta_p <- function(input,eta,kappa_v,u){
  return(-1/2*t(eta)%*%diag.spam(kappa_v,input$n)%*%eta +
           t(eta)%*%(kappa_v*u) +
           t(eta)%*%input$y -
           t(exp(eta))%*%input$E)
}
```

```r
# finding the p-value of eta from the
# proposal density q(eta|...)
d_eta_q <- function(input,eta,z,kappa_v,u){
  c_vec = get_c(input,z)
  b_vec = get_b(input,z)
  b = kappa_v*u + b_vec
  Q = diag.spam(kappa_v,input$n) + diag.spam(c_vec)
  return(dmvnorm.canonical(x = eta, b = b, Q = Q, log = TRUE))
}
```

```r
# calculating the acceptance probability
acceptance_prob <- function(input,eta_prop,eta,kappa_v,u){
  return(min(1,exp(
      d_eta_p(input, eta_prop, kappa_v, u) +
      d_eta_q(input, eta, eta_prop, kappa_v, u) -
      d_eta_p(input, eta, kappa_v, u) -
      d_eta_q(input, eta_prop, eta, kappa_v, u))))
}
```

All the components are now implement, and we can create the MCMC-algorithm. The initial condition for κ_v we use the mean of what we have observed to be the mean after we have run an simulation, and therefore $\kappa_v = 180$. The initial condition of $\boldsymbol{u}$ is sampled from a uniform distribution $\boldsymbol{u} = \mathcal{U}[0,1]$. These values are used to sample the initial proposal of $\boldsymbol{\eta}$. To accept or reject a proposal $\boldsymbol{\eta}^*$, we draw from $x \sim \mathcal{U}[0,1]$, and accept if the $x \leq$ acceptance. The function `myMCMC` returns a list containing $\boldsymbol{\eta}$-, $\boldsymbol{u}$-, κ_u- and κ_v-samples and a vector of acceptance probabilities.

```r
# running MCMC simulation
# return a list of eta, u, kappa_u, kappa_v, and the acceptance
M <- 70000
set.seed(123)
myMCMC <- function(input, M){
  # for keeping track of completition
  pb <- txtProgressBar(min = 0, max = M, style = 3)
  # chosen u
  u = runif(input$n)
  # eta from proposal density
  eta = r_eta_prop(input,u,u,kappa_v =180)
  # storing all etas,vs, kappas and the acceptance prob
  eta_samples = matrix(NA,nrow=M,ncol=input$n)
  u_samples = matrix(NA,nrow=M,ncol=input$n)
  kappa_u_samples = vector()
  kappa_v_samples = vector()
  accept_vec = vector()
  for (i in seq(1,M)){
    # for keeping track of completition
    setTxtProgressBar(pb, i)
    # drawing kappas
    kappa_u = r_kappa_u(input,u)
    kappa_v = r_kappa_v(input,eta,u)
    # drawin u
    u = r_u(input,kappa_u,kappa_v,eta)
    # drawing eta
    eta_prop = r_eta_prop(input,eta,u,kappa_v)
    # cacluating acceptance
    accept_prob = acceptance_prob(input,eta_prop,eta,kappa_v,u)
    if(runif(1) < accept_prob){
      eta = eta_prop
    }
    eta_samples[i,] = eta
    u_samples[i,] = u
    accept_vec = c(accept_vec, accept_prob)
    kappa_u_samples[i] = kappa_u
    kappa_v_samples[i] = kappa_v
  }
  return(list(
    eta = eta_samples,
    u = u_samples,
    accept = accept_vec,
    kappa_u = kappa_u_samples,
    kappa_v = kappa_v_samples
  ))
}
```

To get the running time of our MCMC-algorithm we used the R function `system.time` on the function call and stored this in the list *samples* returned from the `myMCMC` function. The list is then saved along with the *input*-list declared above to save computation time in later exercises.

```r
# runs the MCMC and stores the time it took
run_time <- system.time(samples <- myMCMC(input, M))
# adding runtime to the list from MCMC
samples$run_time = run_time
# saving for use in other code
save(samples,file = "data/samples.Rdata")
save(input,file ="data/input.Rdata")
cat("Average acceptance rate:", mean(samples$accept))
cat("Run time of MCMC:", as.numeric(run_time[1]))
```

```
## Average acceptance rate: 0.7313069
## Run time of MCMC: 338.064
```

From the output of the code above we can see that the acceptance rate is around 0.73 and the time used to run for to generate 70000 samples is 525 seconds.

Exercise 3

In exercise 2 we have run an MCMC-algorithm and generated $M = 70000$ samples. In this exercise we will look at these results and determine the goodness of our MCMC algorithm and potential improvements. First we define the libraries we used.

```
library(tidyverse)
library(spam)
library(ggpubr)
set.seed(123)
load("data/input.Rdata")
load("data/samples.Rdata")
```

To create the figures we used `ggplot` from the **tidyverse** package. From the samples of $\boldsymbol{\eta}$ we can calculate the samples of $\boldsymbol{v}$ by the equation $\boldsymbol{v} = \boldsymbol{\eta} - \boldsymbol{u}$. We look at 3 randomly chosen components of $\boldsymbol{u}$ and $\boldsymbol{v}$ and the samples of κ_u and κ_v.

```
r_cols <- sort(sample(1:input$n,3,replace = F))
M <- length(samples$eta[,1])
burnin = 20000
steps <- seq(1,M)
burnin_step = seq(burnin,M)

MCMC_list <- data.frame(
  steps = steps,
  v1 = samples$eta[steps,r_cols[1]] - samples$u[steps,r_cols[1]],
  v2 = samples$eta[steps,r_cols[2]] - samples$u[steps,r_cols[2]],
  v3 = samples$eta[steps,r_cols[3]] - samples$u[steps,r_cols[3]],
  u1 = samples$u[steps,r_cols[1]],
  u2 = samples$u[steps,r_cols[2]],
  u3 = samples$u[steps,r_cols[3]],
  kappa_u = samples$kappa_u[steps],
  kappa_v = samples$kappa_v[steps],
  is_burnin = c(rep(TRUE,burnin),rep(FALSE,M-burnin)))
save(MCMC_list,file = "data/mcmc_df.Rdata")
```

(a) To start out we look at trace plots with a chosen burn in period removed. The size of the burn-in is chosen by looking at the trace plots in figure 1 and by the results of the Geweke test in 3c. We chose 20000 to be our burn-in. Which means that we use 50000 samples as usable samples.

```r
# trace plots with burn in removed
fig_3a <- ggarrange(
  ggplot(MCMC_list[burnin_step,], aes(x = steps, y = v1)) +
    geom_line(color = "grey24") + rremove("x.text") +
    rremove("xlab") + rremove("ylab"),
  ggplot(MCMC_list[burnin_step,], aes(x = steps, y = v2)) +
    geom_line(color = "grey24") + rremove("x.text") +
    rremove("xlab") + rremove("ylab"),
  ggplot(MCMC_list[burnin_step,], aes(x = steps, y = v3)) +
    geom_line(color = "grey24") + rremove("x.text") +
    rremove("xlab") + rremove("ylab"),
  ggplot(MCMC_list[burnin_step,], aes(x = steps, y = u1)) +
    geom_line(color = "grey24") + rremove("x.text") +
    rremove("xlab") + rremove("ylab"),
  ggplot(MCMC_list[burnin_step,], aes(x = steps, y = u2)) +
    geom_line(color = "grey24") + rremove("x.text") +
    rremove("xlab") + rremove("ylab"),
  ggplot(MCMC_list[burnin_step,], aes(x = steps, y = u3)) +
    geom_line(color = "grey24") + rremove("x.text") +
    rremove("xlab") + rremove("ylab"),
  ggplot(MCMC_list[burnin_step,], aes(x = steps,y = kappa_u)) +
    geom_line(color = "grey24") + rremove("x.text") +
    rremove("xlab") + rremove("ylab"),
  ggplot(MCMC_list[burnin_step,], aes(x = steps, y = kappa_v)) +
    geom_line(color = "grey24") + rremove("ylab"),
  nrow = 8,
  ncol = 1,
  labels = c(sprintf("v[m = %d]",r_cols[1]),
             sprintf("v[m = %d]",r_cols[2]),
             sprintf("v[m = %d]",r_cols[3]),
             sprintf("u[m = %d]",r_cols[1]),
             sprintf("u[m = %d]",r_cols[2]),
             sprintf("u[m = %d]",r_cols[3]),
             sprintf("  kappa_u"),sprintf("  kappa_v")),
  font.label = list(size = 10, color = "firebrick1"),
  label.x = 0,
  label.y = 1,
  hjust= -1.7,
  vjust = 1.9
)
fig_3a
```

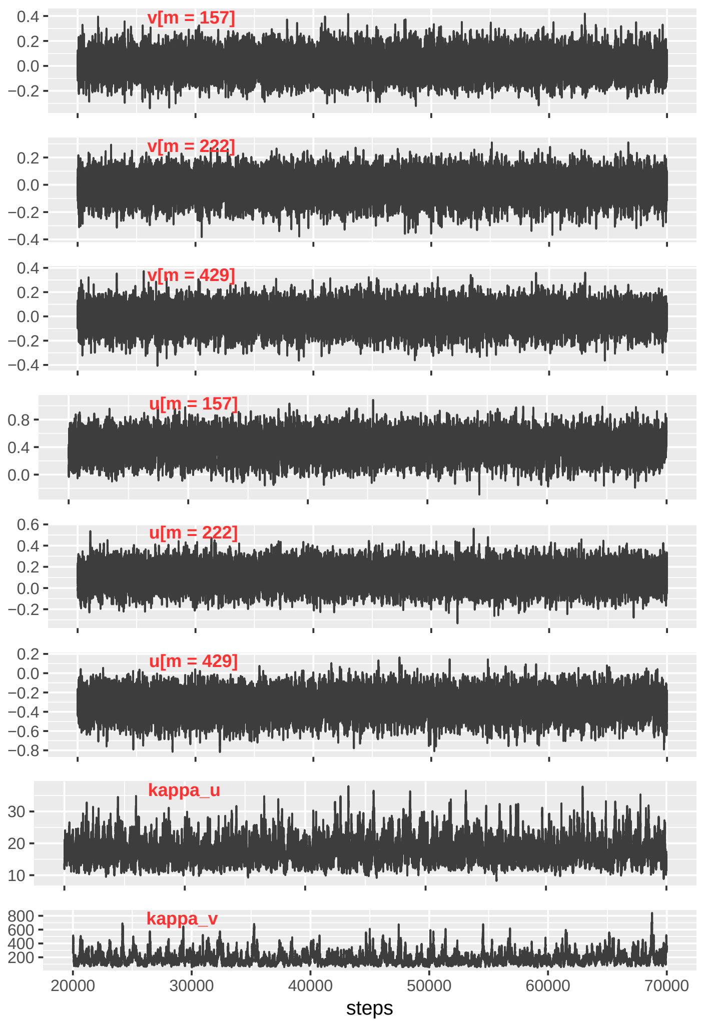

Figure 1: Trace plot of the parameters of interest after removing the burn-in.

Looking at the trace plots for the three different parameters for $\boldsymbol{u}$ and $\boldsymbol{v}$ in figure 1, we see that the parameters explore much of the domain, and look like a band. This is a good indication of convergence of the MCMC. κ_u and κ_v seem to explore less of the domain, and they tend to explore more higher values than the mean than smaller values. Also we see the mean of the $\boldsymbol{v}$ components is close to zero which is what we should expect for white noise.

(b) Using the same samples we will now look at their autocorrelation. We will do this by using the R-function `acf`, and then create autocorrelation plots of the different samples.

```r
# autocorrelation plots with burn in removed
MCMC_acf <- data.frame(
  lag=acf(MCMC_list$kappa_v[burnin_step],plot=F,lag.max=200)$lag,
  v1 = acf(MCMC_list$v1[burnin_step],plot = F,lag.max = 200)$acf,
  v2 = acf(MCMC_list$v2[burnin_step],plot = F,lag.max = 200)$acf,
  v3 = acf(MCMC_list$v3[burnin_step],plot = F,lag.max = 200)$acf,
  u1 = acf(MCMC_list$u1[burnin_step],plot = F,lag.max = 200)$acf,
  u2 = acf(MCMC_list$u2[burnin_step],plot = F,lag.max = 200)$acf,
  u3 = acf(MCMC_list$u3[burnin_step],plot = F,lag.max = 200)$acf,
  kappa_u = acf(MCMC_list$kappa_u[burnin_step],
              plot = F,lag.max = 200)$acf,
  kappa_v = acf(MCMC_list$kappa_v[burnin_step],
              plot = F,lag.max = 200)$acf
)



fig_3b <- ggarrange(
  ggplot(MCMC_acf,aes(x = lag, y = v1)) + rremove("x.text") +
    geom_hline(aes(yintercept = 0)) +
    rremove("xlab") + rremove("ylab") +
    geom_segment(aes(xend = lag, yend = 0)),
  ggplot(MCMC_acf,aes(x = lag, y = v2)) + rremove("x.text") +
    geom_hline(aes(yintercept = 0)) +
    rremove("xlab") + rremove("ylab") +
    geom_segment(aes(xend = lag, yend = 0)),
  ggplot(MCMC_acf,aes(x = lag, y = v3)) + rremove("x.text") +
    geom_hline(aes(yintercept = 0)) +
    rremove("xlab") + rremove("ylab") +
    geom_segment(aes(xend = lag, yend = 0)),
  ggplot(MCMC_acf,aes(x = lag, y = u1)) + rremove("x.text") +
    geom_hline(aes(yintercept = 0)) +
    rremove("xlab") + rremove("ylab") +
    geom_segment(aes(xend = lag, yend = 0)),
  ggplot(MCMC_acf,aes(x = lag, y = u2)) + rremove("x.text") +
    geom_hline(aes(yintercept = 0)) +
    rremove("xlab") + rremove("ylab") +
    geom_segment(aes(xend = lag, yend = 0)),
  ggplot(MCMC_acf,aes(x = lag, y = u3)) + rremove("x.text") +
    geom_hline(aes(yintercept = 0)) +
    rremove("xlab") + rremove("ylab") +
    geom_segment(aes(xend = lag, yend = 0)),
  ggplot(MCMC_acf,aes(x = lag, y = kappa_u)) + rremove("x.text") +
    geom_hline(aes(yintercept = 0)) +
    rremove("xlab") + rremove("ylab") +
    geom_segment(aes(xend = lag, yend = 0)),
  ggplot(MCMC_acf,aes(x = lag, y = kappa_v)) +
    geom_hline(aes(yintercept = 0)) +
    rremove("xlab") + rremove("ylab") +
    geom_segment(aes(xend = lag, yend = 0)),
  nrow = 8,
  ncol = 1,
  labels = c(sprintf("v[m = %d]",r_cols[1]),
             sprintf("v[m = %d]",r_cols[2]),
             sprintf("v[m = %d]",r_cols[3]),
             sprintf("u[m = %d]",r_cols[1]),
             sprintf("u[m = %d]",r_cols[2]),
             sprintf("u[m = %d]",r_cols[3]),
             sprintf("  kappa_u"),sprintf("  kappa_v")),
  font.label = list(size = 10, color = "firebrick1"),
  label.x = 0,
  label.y = 1,
  hjust= -1.7,
  vjust = 1.9
)
fig_3b
```

In figure 2 we can see that the autocorrelation for $\boldsymbol{v}$ is small, but we can see that for the $\boldsymbol{u}$ components there is a small autocorrelation for small lags and they look to be independent around lag$\approx$ 20. On the other hand κ_u and κ_v we can see a larger degree of autocorrelation. κ_v is close to independent around a lag of size 70 and κ_u needs almost a lag of 150 to be independent.

(c) Next we will test the convergence of the Markov chain using the `geweke.diag` function from the R-package **coda**. Geweke's diagnostic is done by comparing the location of the sampled parameter on two different time intervals. Usually one compares the last half of the chain with some smaller part in the start of the chain. We have chosen the first part to be 10% of the start of the chain after the burn-in and the last interval to be 50% of the last part of the chain. Let first interval of a parameter be Θ_A of length n_A, with the mean $\bar{\Theta}_A$ and the standard error s_A. And the last interval be Θ_B of length n_B, with mean $\bar{\Theta}_B$ and standard

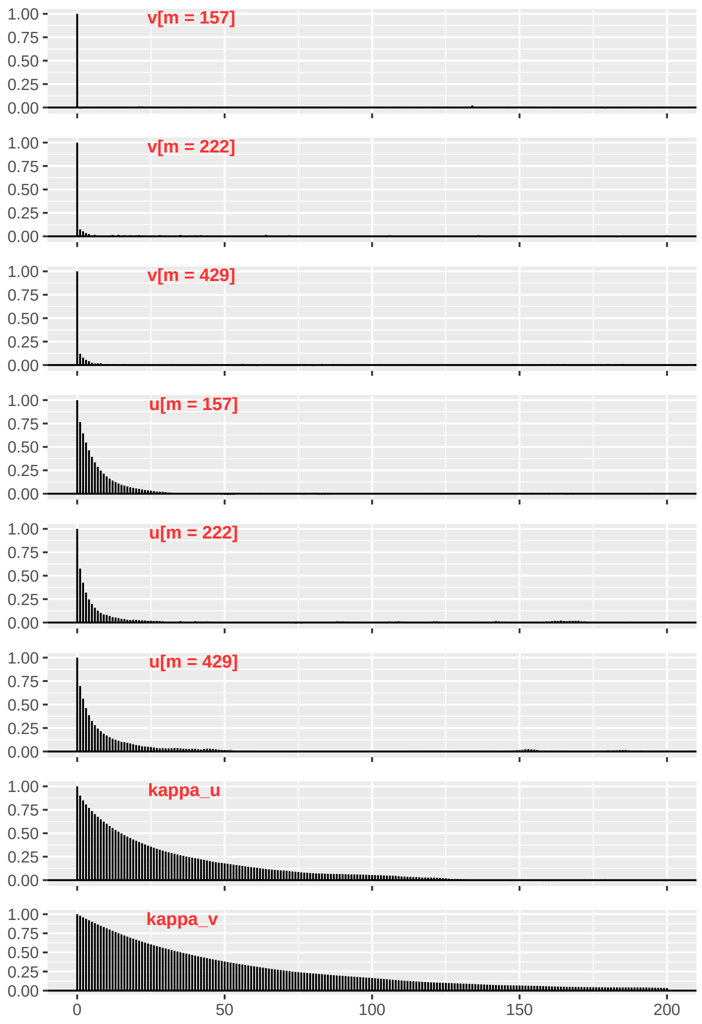

Figure 2: Autocorrelation plots for the parameters of interest after the burn-in is removed.

Table 1: Result of Gweke Statistic on the parameters

	z_scores	p_values
v1	0.4698480	0.6384636
v2	-0.3531959	0.7239416
v3	1.5577774	0.1192860
u1	-0.7297690	0.4655314
u2	-0.4070320	0.6839845
u3	-0.5171681	0.6050388
kappa_u	-0.4465260	0.6552173
kappa_v	0.3613914	0.7178069

error s_B. Then the Geweke's statistics is given by Equation (8).

$$Z_\Theta = \frac{\bar{\Theta}_A - \bar{\Theta}_B}{\sqrt{\frac{s_A}{n_A} + \frac{s_B}{n_B}}} \tag{8}$$

If the samples then are asymptotically independent and drawn from the stationary distribution of the chain, the Geweke's statistic, will be standard normally distributed, $\mathcal{N}(0,1)$, and the p-values are given by

$$p\text{-value} = P(|Z| \geq |z|).$$

This mean that we can test the convergence of the chain on the p-values of the z-statistics. We can then perform hypothesis tests

$$\text{H}_0 : P(|Z| \geq |z|) > 0.05 \ \text{ vs. } \ \text{H}_1 : P(|Z| \geq |z|) < 0.05, \tag{9}$$

with the chosen significance level of 0.05. We could also chose a interval of z for which the Geweke's statistic needs to be within to have convergence. We calculate the Geweke diagnostic with the following code, removing the burn-in from the chain.

```r
library(coda)
library(kableExtra)
z_scores <- geweke.diag(MCMC_list[seq(20000,M),2:9],
                        frac1=0.1, frac2=0.5)$z
geweke_diag <- data.frame(
  z_scores = z_scores,
  p_values = 2*pnorm(abs(z_scores),lower.tail = FALSE)
)
kable(geweke_diag,caption =
        "\\label{tab:geweke}Result of Gweke
      Statistic on the parameters",
      booktabs = T)
```

From Table 1 returned from the code we see that the hypothesis H_0 is accepted for κ_u, κ_v and the three randomly chosen components of $\boldsymbol{u}$ and $\boldsymbol{v}$, which indicates that the chain has converged with a burn-in of 20000. We can also look at the size of the burn-in and at which values the hypothesis doesn't hold. This we have done by plotting the z-statistics for different sizes of burn-in, and since it is normally distributed we can decide which interval the values need to be within for the sequence to have converged. A interval of $Z \in [-1.6, 1.6]$ gives a significance level of 0.055.

```r
testBurningGeweke <- function(MCMC_list, M){
  burn = numeric()
  z_name = numeric()
  z = numeric()
  for (i in seq(1,11)){
    burnin[i] = 2000*(i-1)
    z_score<-geweke.diag(MCMC_list[seq(burnin[i],M),2:9],
                         frac1=0.1, frac2=0.5)$z
    burn = c(burn,rep(burnin[i],8))
    z = c(z,as.vector(z_score))
    z_name = c(z_name,names(z_score))
  }
  return(data.frame(burnin = burn,
                    z_statistic = z,
                    Parameter = z_name))
}
z_scores_burnin<-testBurningGeweke(MCMC_list, M)
burnin_test_plot<-ggplot()+
  geom_point(data= z_scores_burnin,
             aes(x=z_statistic,y=burnin,color=Parameter),size=2)+
  geom_rect(aes(xmin=-1.6,xmax = 1.6, ymin=-200,ymax=20200 ),
            fill = "blue",alpha = 0.1)+
  ylab("Burn-in")+
  xlab("Z-statistic")+
  labs(colour="Parameter")
burnin_test_plot
```

From figure 3 we can see that the chain has converged with a burn-in of 18000 according to the Geweke diagnostic. We choose to use a burn-in of 20000 samples.

Exercise 4

In this exercise we will calculate the effective samples size (ESS) of the precision parameters κ_u and κ_v. This is an estimate of the number of independent samples, using the autocorrelations, that are generated with the MCMC algorithm. The

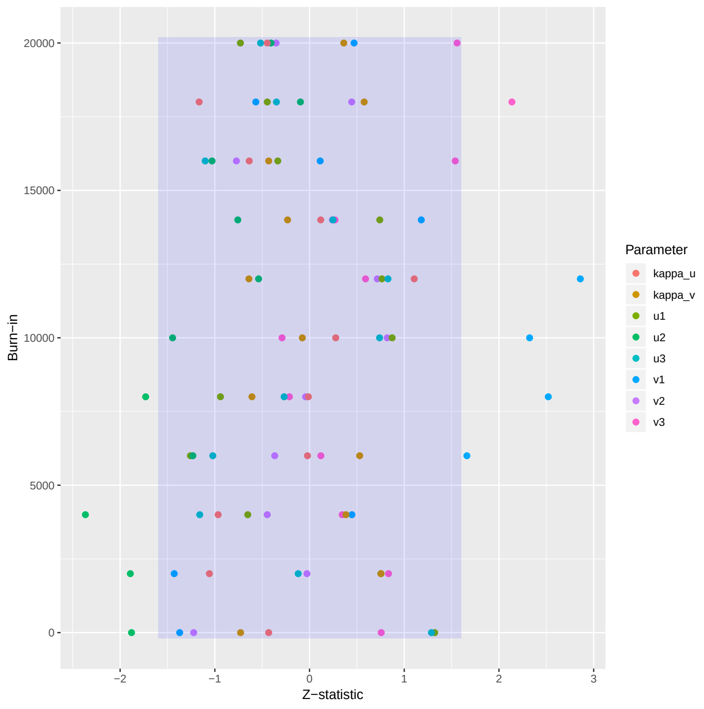

Figure 3: A comparison of the Geweke's Statistical of our chain with different burn-in. The y-axis is the burn-in size, the x-axis is the z-statistic and the colors of the points is the different parameters. The blue box is the significance level of $p = 0.055$.

Table 2: Effective sample size of the parameters

v1	49110.3057
v2	28830.4649
v3	28020.1093
u1	4571.4594
u2	7791.1068
u3	4411.2654
kappa_u	1111.0257
kappa_v	492.6939

autocorrelations are given by Equation 10, where k is the lag, which is the number of samples used, and $\rho(k)$ is the autocorrelation at lag k.

$$\tau = 1 + 2\sum_{k=1}^{\infty} \rho(k) \tag{10}$$

With this we can calculate the ESS as

$$\text{ESS} = \frac{N}{\tau} \tag{11}$$

To calculate the ESS we used `effectiveSize` from the R-package **coda**, shown in the code bellow and the results is displayed in the Table 2.

```r
# effective sample size
library(tidyverse)
library(coda)
library(kableExtra)
load("data/input.Rdata")
load("data/mcmc_df.Rdata")
ess <- effectiveSize(MCMC_list[MCMC_list$is_burnin==F, 2:9])
kable(ess,caption = "\\label{tab:ess}Effective sample size
      of the parameters",booktabs = T)
```

From Table 2 we can see that the effective sample size or the independent samples is for $\kappa_u = 1111$ and for $\kappa_v = 493$. The components of $\boldsymbol{v}$ has pretty large effective sample size, and $\boldsymbol{u}$ has a little less. This is what we would expect from the autocorrelation plots in figure 2. We can also see from this figure that the lag for κ_u and κ_v needs to be much larger before the samples could be considered to be independent. A method that could be used to improve the effective sample size is block updates of correlated parameters. Block updates could also lead to very low acceptance probabilities, however.

Now we will look at the relative ESS. This is calculated by dividing the mean of the estimated sample size of κ_u and κ_v by the time used to generate the effective samples in the MCMC.

```
load("../code/data/samples.Rdata")
time <- as.numeric(samples$run_time[1])
relESS <- mean(c(ess[7],ess[8]))/time
cat("Relative effective sample size:", relESS)
```

```
## Relative effective sample size: 2.371917
```

In other words our MCMC-algorithm generates ≈ 2.37 effective samples per second. In our case this might not be that useful, but if we were to change the implementation of the MCMC, we could compare the relative sample size of our two implementations to see if the change was better for our model or not. As if we were to implement a block update model, this might affect the run time, and by looking at the relative sample size value we could determine if the change would be good.

Exercise 5

Now we look at the results of our MCMC algorithm. We will do this by plotting the exponential of the posterior median of $\boldsymbol{u}$ the spatial structure component with the burn-in removed, which means median($exp(\boldsymbol{u})$). In figure 4 we see the standardised mortality rates given by our data y_i/E_i and the spatial structured effect.

```
library(spam)
library(colorspace)
load("data/input.Rdata")
load("data/samples.Rdata")
burnin = 5000
M = length(samples$u[,1])
u_med_exp <- function(u){
  temp_median = apply(u,2, median, na.rm = TRUE)
  posterior = exp(temp_median)
}
# color scheme
col <- diverge_hcl(8) # blue - red
pdf(file = "../figures/germany_5a.pdf", width=10, height=8)
par(mfrow = c(1,2))
# standardised mortality rates
germany.plot(Oral$Y/Oral$E,
             col=col,
             legend=TRUE,
             main="Standardised mortaility rates",
             cex.main=1)
# spatial structured effects
```

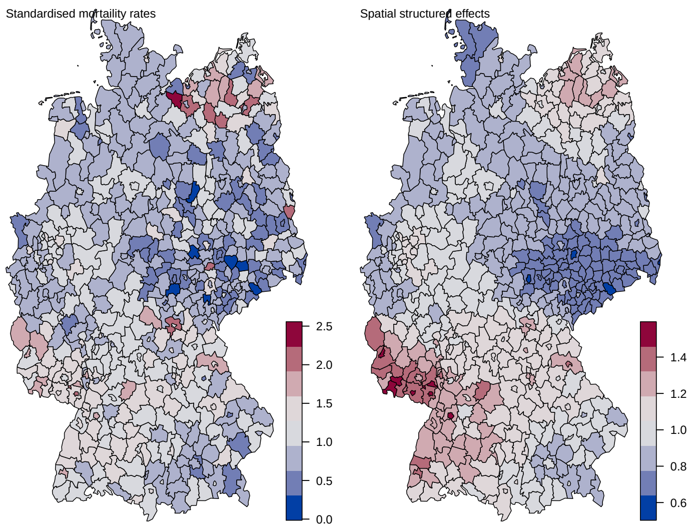

Figure 4: The plot on the left side is the standardised mortality rates, and on the right side we have the spatial structured effect of oral cavity cancer mortality rates. Low values of mortality rates are blue and high values are red.

```
germany.plot(u_med_exp(samples$u[seq(burnin,M),]),
             col = col,
             legend=TRUE,
             main="Spatial structured effects",
             cex.main=1)
```

Exercise 6

We now implement the same model as before using `R-INLA`. The latent Gaussian model given in (1) is termed `besag` in `R-INLA`. The unstructured random effects are defined in the latent model `iid`. `R-INLA` represents precision parameters τ as $\theta = \log \tau$ and defines the priors on θ, so we need to set our hyperpriors (the priors for κ_u and κ_v) as `loggamma`. The parametrization is such that the parameters are still α_u, β_u and α_v, β_v, respectively. We set no intercept so the model is the same as before, and set `constr=FALSE` in the `besag` part to account for this. The model is implemented and various plots created with the following code.

```
# Libraries
library(INLA)
library(spam)
library(fields, warn.conflict=FALSE)
library(colorspace)
```

```r
library(ggplot2)
library(tibble)
library(gridExtra)
library(latex2exp)

# Load and prepare data
# str(Oral)
attach(Oral)
col = diverge_hcl(8)  # blue - red
germany.plot(Oral$Y/Oral$E, col=col, legend=TRUE)
g = system.file("demodata/germany.graph", package="INLA")

# Variables
alpha = 1
beta = 0.01
region = 1:length(Y)
# Define formula and run INLA
hyper = list(prec=list(prior="loggamma", param=c(alpha, beta)))
formula = Y ~
  -1 +
  f(region_struct, model="besag", graph.file=g, constr=FALSE,
    hyper=hyper) +
  f(region_random, model="iid", hyper=hyper)
data = list(Y=Y, E=E, region_struct=region, region_random=region)
result_a = inla(formula, family="poisson", data=data, E=E,
                control.compute=list(dic=TRUE))
# summary(result_a)

# Plot median of posterior divided by E in map
# (Y/E = E*exp(eta)/E = exp(eta))
y_median_a = exp(result_a$summary.random$region_struct$`0.5quant`)
pdf("../figures/y_median_a.pdf",width = 4, height = 4)
germany.plot(y_median_a, col=col, legend=TRUE)
dev.off()

# Improve estimates of the posterior marginals
result_a = inla.hyperpar(result_a)

# Get and smooth marginals
rns = c(157, 222, 429) ########################################
kappa_u_marginal = result_a$marginals.hyperpar$
  `Precision for region_struct`
kappa_u_marginal = inla.smarginal(kappa_u_marginal)
kappa_v_marginal = result_a$marginals.hyperpar$
  `Precision for region_random`
kappa_v_marginal = inla.smarginal(kappa_v_marginal)
```

```
i1 = paste0("index.", rns[1])
i2 = paste0("index.", rns[2])
i3 = paste0("index.", rns[3])
u1_marginal = inla.smarginal(
  result_a$marginals.random$region_struct[[i1]])
u2_marginal = inla.smarginal(
  result_a$marginals.random$region_struct[[i2]])
u3_marginal = inla.smarginal(
  result_a$marginals.random$region_struct[[i3]])
v1_marginal = inla.smarginal(
  result_a$marginals.random$region_random[[i1]])
v2_marginal = inla.smarginal(
  result_a$marginals.random$region_random[[i2]])
v3_marginal = inla.smarginal(
  result_a$marginals.random$region_random[[i1]])

# Load mcmc sample data
load("data/mcmc_df.Rdata")
MCMC_list = MCMC_list[!(MCMC_list$is_burnin),]
kappa_u_samples = tibble(x=MCMC_list$kappa_u)
kappa_v_samples = tibble(x=MCMC_list$kappa_v)
u1_samples = tibble(x=MCMC_list$u1)
u2_samples = tibble(x=MCMC_list$u2)
u3_samples = tibble(x=MCMC_list$u3)
v1_samples = tibble(x=MCMC_list$v1)
v2_samples = tibble(x=MCMC_list$v2)
v3_samples = tibble(x=MCMC_list$v3)

# Plot marginals and histograms
p_kappa_u = ggplot(as_tibble(kappa_u_marginal)) +
  geom_line(aes(x, y), col="darkred") +
  geom_histogram(
    data=kappa_u_samples, aes(x, y=..density..), size=0.1,
    bins=70, fill="cornflowerblue", col="white", alpha=0.5) +
  labs(x=expression(kappa[u]), y="") +
  theme(plot.margin = unit(c(0.1, 0.1, 0, -0.4), "cm"))
p_kappa_v = ggplot(as_tibble(kappa_v_marginal)) +
  geom_line(aes(x, y), col="darkred") +
  geom_histogram(
    data=kappa_v_samples, aes(x, y=..density..), size=0.1,
    bins=70, fill="cornflowerblue", col="white", alpha=0.5) +
  labs(x=expression(kappa[v]), y="") +
  xlim(c(min(kappa_u_marginal$x), 1000)) +
  theme(plot.margin = unit(c(0, 0.1, 0, -0.4), "cm"))
p_u1 = ggplot(as_tibble(u1_marginal)) +
  geom_line(aes(x, y), col="darkred") +
  geom_histogram(
    data=u1_samples, aes(x, y=..density..), size=0.1,
    bins=70, fill="cornflowerblue", col="white", alpha=0.5) +
  labs(x=TeX(paste0("u_{", rns[1], "}")), y="") +
  theme(plot.margin = unit(c(0, 0.1, 0, -0.4), "cm")) +
  xlim(c(min(u1_marginal$x), 1.5))
p_u2 = ggplot(as_tibble(u2_marginal)) +
  geom_line(aes(x, y), col="darkred") +
  geom_histogram(
    data=u2_samples, aes(x, y=..density..), size=0.1,
    bins=70, fill="cornflowerblue", col="white", alpha=0.5) +
  labs(x=TeX(paste0("u_{", rns[2], "}")), y="") +
  theme(plot.margin = unit(c(0, 0.1, 0, -0.4), "cm"))
p_u3 = ggplot(as_tibble(u3_marginal)) +
  geom_line(aes(x, y), col="darkred") +
  geom_histogram(
    data=u3_samples, aes(x, y=..density..), size=0.1,
    bins=70, fill="cornflowerblue", col="white", alpha=0.5) +
  labs(x=TeX(paste0("u_{", rns[3], "}")), y="") +
  theme(plot.margin = unit(c(0, 0.1, 0, -0.4), "cm"))
p_v1 = ggplot(as_tibble(v1_marginal)) +
  geom_line(aes(x, y), col="darkred") +
  geom_histogram(
    data=v1_samples, aes(x, y=..density..), size=0.1,
    bins=70, fill="cornflowerblue", col="white", alpha=0.5) +
  labs(x=TeX(paste0("v_{", rns[1], "}")), y="") +
  theme(plot.margin = unit(c(0, 0.1, 0, -0.4), "cm"))
p_v2 = ggplot(as_tibble(v2_marginal)) +
  geom_line(aes(x, y), col="darkred") +
  geom_histogram(
    data=v2_samples, aes(x, y=..density..), size=0.1,
    bins=70, fill="cornflowerblue", col="white", alpha=0.5) +
  labs(x=TeX(paste0("v_{", rns[2], "}")), y="") +
  theme(plot.margin = unit(c(0, 0.1, 0, -0.4), "cm"))
p_v3 = ggplot(as_tibble(v3_marginal)) +
  geom_line(aes(x, y), col="darkred") +
  geom_histogram(
    data=v3_samples, aes(x, y=..density..), size=0.1,
    bins=70, fill="cornflowerblue", col="white", alpha=0.5) +
  labs(x=TeX(paste0("v_{", rns[3], "}")), y="") +
  theme(plot.margin = unit(c(0, 0.1, 0, -0.4), "cm"))
p_marginals = grid.arrange(
  p_kappa_u, p_kappa_v, p_u1, p_v1, p_u2, p_v2, p_u3, p_v3,
  ncol=2)
```

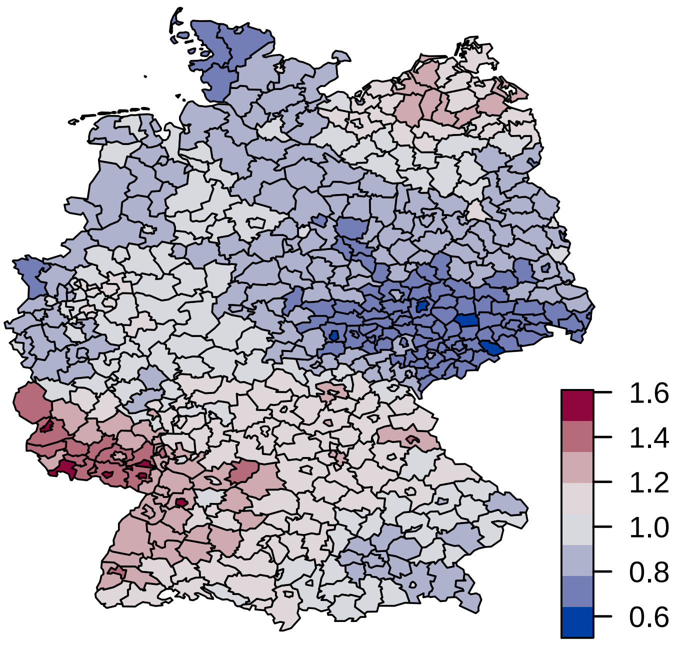

Figure 5: Estimated standardized mortality rates y_i/E_i using the median of the structured effect in the first model created using R_INLA in Exercise 6.

```r
ggsave("../figures/posterior_marginals.pdf", plot=p_marginals,
       width=5, height=7, units="in", dpi=300)
```

Figure 5 shows the estimated standardized mortality rates y_i/E_i using the median of the structured effect in the first model created using R-INLA. I.e. we estimate y_i by $E_i \,\text{median}(e^{u_i}) = E_i e^{\text{median}(u_i)}$, since e^x is monotone, and use the posterior marginal of u to get an estimate for median(u). This is a simple estimate of the rate of $y_i \mid \eta_i$. It gives a good qualitative impression of the mortality rates - individuals living in the south-west or north-east have increased probability of dying of oral cavity cancer, while the probability is lower e.g. in the middle eastern part of Germany. That is, according to our model.

Figure 6 shows histograms of MCMC-samples (50000 samples after burn-in with 20000 samples) and posterior marginals obtained using INLA for κ_u, κ_v and three randomly chosen components of $\boldsymbol{u}$ and $\boldsymbol{v}$. The histograms and the marginals obtained using INLA are closely matched, although the samples for v_{429} are biased a little to the left compared to the marginal obtained by INLA. The INLA estimates are obtained within few seconds, while the MCMC algorithm used sevaral minutes. The fact that the results are so similar is a good indication that both models are implemented correctly and work as expected.

Next we create models including the effect of a covariate representing cigarette consumption. This straight forward and quickly done using R-INLA (after some

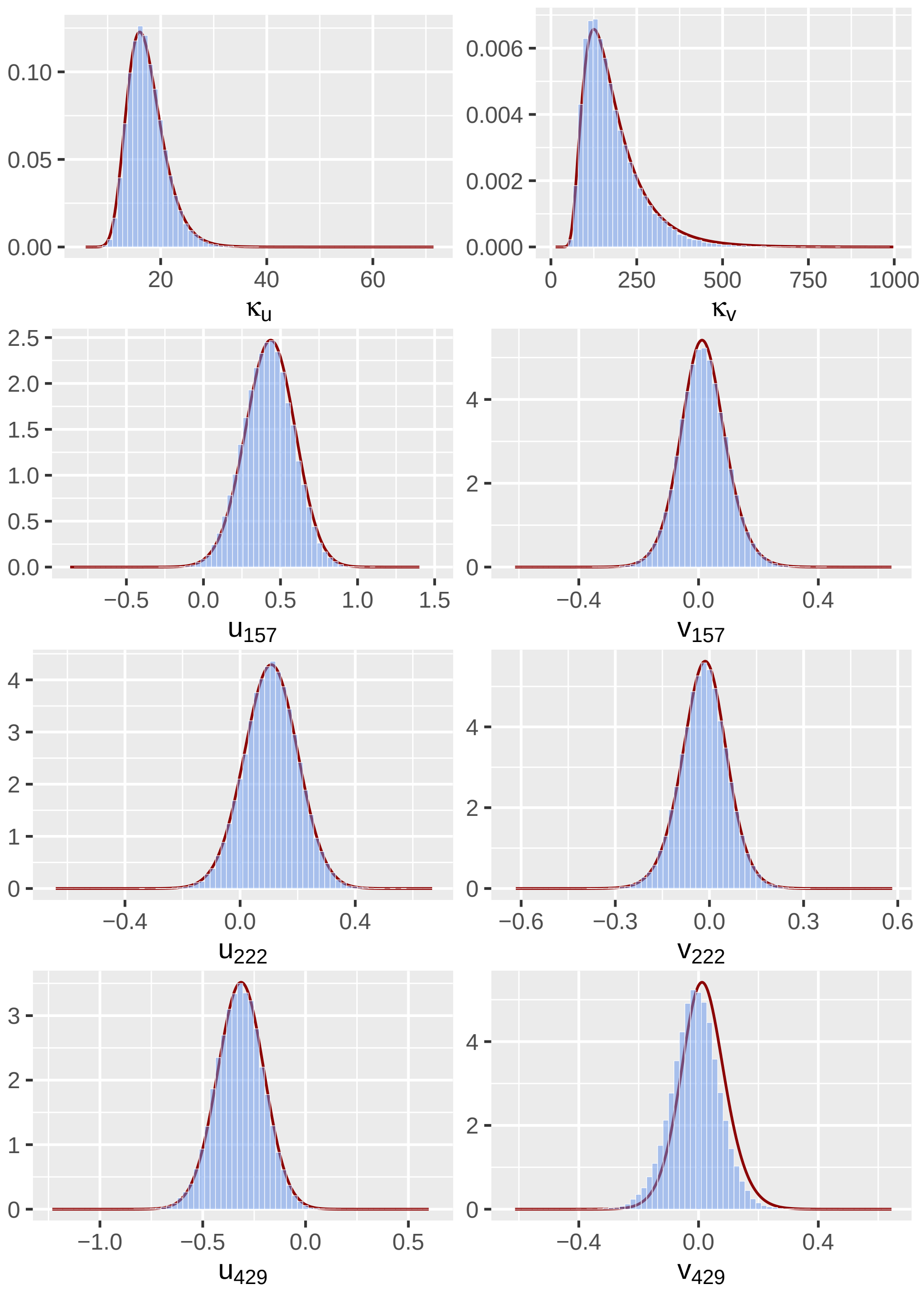

Figure 6: Histograms of MCMC-samples and posterior marginals obtained by using INLA (red lines) for κ_u, κ_v and three randomly chosen components of $\boldsymbol{u}$ and $\boldsymbol{v}$.

practice). We first incorporate the covariate as a linear effect, i.e. $\eta_i = u_i + v_i + \beta_{smoke}$. The value of β_{smoke} is estimated to 0.0046, with standard deviation 0.0011. Then we use a second order random walk to enable non-linearity, using a Gamma prior with the default parameters $\alpha = 1$, $\beta = 5 \cdot 10^{-5}$. The models are implemented in the following code. We also get the DIC values for all the INLA models and plot the posterior median and a 95% credible interval of the non-linear covariate.

```r
smoking = read.table("./data/ex2_additionalFiles/smoking.dat")
data["smoking"] = smoking

# Define formula and run INLA with smoking as a linear effect
formula_b_lin = Y ~
  -1 +
  f(region_struct, model="besag", graph.file=g, constr=FALSE,
    hyper=hyper) +
  f(region_random, model="iid", hyper=hyper) +
  smoking
result_b_lin = inla(formula_b_lin, family="poisson", data=data,
                    E=E, control.compute=list(dic=TRUE))
# summary(result_b_lin)

# Define formula and run INLA with smoking as a rw2 effect
formula_b_rw2 = Y ~
  -1 +
  f(region_struct, model="besag", graph.file=g, constr=FALSE,
    hyper=hyper) +
  f(region_random, model="iid", hyper=hyper) +
  f(smoking, model="rw2")
result_b_rw2 = inla(formula_b_rw2, family="poisson", data=data,
                    E=E, control.compute=list(dic=TRUE))
# summary(result_b_rw2)

# Get DIC for all the models
smoking_none_DIC = result_a$dic$dic
smoking_lin_DIC = result_b_lin$dic$dic
smoking_rw2_DIC = result_b_rw2$dic$dic

# Plot posterior median and 95% CI of non-linear effect
smoking_beta = result_b_lin$summary.fixed$mean
smoking_val = result_b_rw2$summary.random$smoking$ID
smoking_eff_025 = result_b_rw2$summary.random$smoking$`0.025quant`
smoking_eff_5 = result_b_rw2$summary.random$smoking$`0.5quant`
smoking_eff_975 = result_b_rw2$summary.random$smoking$`0.975quant`

data = tibble(
  x = smoking_val,
  lin = smoking_beta * smoking_val + smoking_eff_5[1],
  med = smoking_eff_5,
  ci_low = smoking_eff_025,
  ci_high = smoking_eff_975
)
plot_smoking = ggplot(data) +
  geom_ribbon(aes(x, ymin=ci_low, ymax=ci_high), alpha=0.2) +
  geom_line(aes(x, y=lin, col="linear")) +
  geom_line(aes(x, y=med, col="rw2")) +
  geom_line(aes(x, y=ci_low, col="95CI")) +
  geom_line(aes(x, y=ci_high, col="95CI")) +
  # scale_color_manual(values=c("black", "pink", "yellow")) +
  labs(x="smoking", y="effect", col="")
ggsave("../figures/smoking_effect.pdf", plot=plot_smoking,
       width=5.5, height=3, units="in", dpi=300)
```

The DIC values for the model without the smoking covariate, the model with the smoking covariate as a linear effect and the model with the smoking covariate as a non-linear effect are 3286.9, 3276.9 and 3276.9, respectively. The DIC (deviance information criterion) measures the fit to the data while penalizing the complexity of the model (lower values are better). Using this criteria we conclude that including smoking as a covariate results in better models.

The difference between the two models including the smoking covariate seems to be very small. Considering the plot in figure 7 this is not surprising, since the non-linear effect of smoking in the third model is not very different from the linear effect in the second model - except for a constant, which will likely be compensated for in the structured part of the model. Since the second model has a lower DIC, and it also is a simpler model, which is preferable, this seems to be the optimal model of the three.

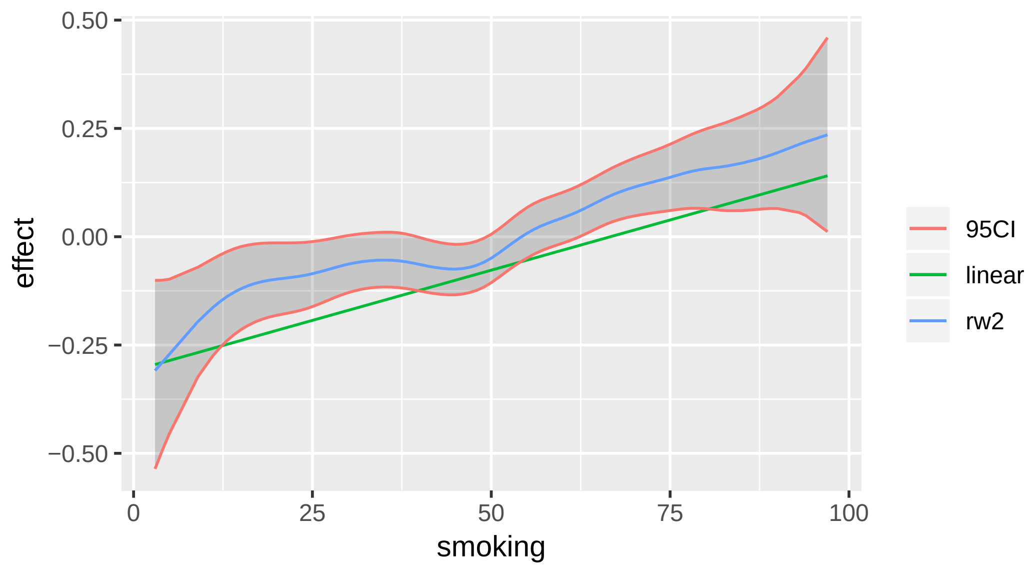

Figure 7: Posterior median and 95% credible interval of the effect of the smoking covariate modelled as a second order random walk. The green line is the effect when modelled linearly, with a constant subtracted for the sake of comparison.