TMA4300Ex2

Karine H. Foss & August S. Mathisen

18 2 2019

Comments:

- Need to explain better what $q(\eta)$ is. This is proposal distribution?
- Which elements in the canonical form of mvnorm are the precision matrix etc?
- ex3: says u and v in text; should be eta?
- ex3: should we remove burn-in period before performing test? (and acf?)
- ex5: says exp(u), but should be eta??

The goal of this project is to study the distribution of oral cancer in Germany. We will mainly study this problem through the lens of a spatial Bayesian model, byt we will also briefly discuss the effects of smoking. Our data set is composed of the number of lip cancer patients in each district of Germany. We will assume that the number of cases are conditionally independent Poisson distrubuted variables. That is

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

where y_i is the number of cases in district i, E_i is an expected number of cases determined by a districts demography and η_i is the log-relative risk associated with the district. Initially we will only take the spatial effects into account, and we therefore let $\boldsymbol{\eta} = \boldsymbol{u} + \boldsymbol{v}$, where $\boldsymbol{v} \sim N(0, \kappa_v^{-1}\mathbf{I})$ is a noise term and $\boldsymbol{u}$ is a spatially structured component with distribution

$$\text{p}(\boldsymbol{u}|\kappa_u) \propto \kappa_u^{(n-1)/2} \exp\left\{-\frac{\kappa_u}{2}\boldsymbol{u}^T\mathbf{R}\boldsymbol{u}\right\}.$$

Here $\mathbf{R}$ is a neighbour matrix with n_i, the number of neighbour districts to district i, on the diagonal, -1 if district i and j are neighbours and zero otherwise. This will be a sparse non-negaitve definit matrix, and is therefore a possible precision matrix. The conditional distribution of $\boldsymbol{\eta}$ is then

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

κ_u and κ_v are the hyperparameters of the problem, and both precisions parameters are assumed to be gamma distributed. That is

$$\kappa_u \sim \text{Gamma}(\alpha_u, \beta_u)\kappa_v \sim \text{Gamma}(\alpha_v, \beta_v).$$

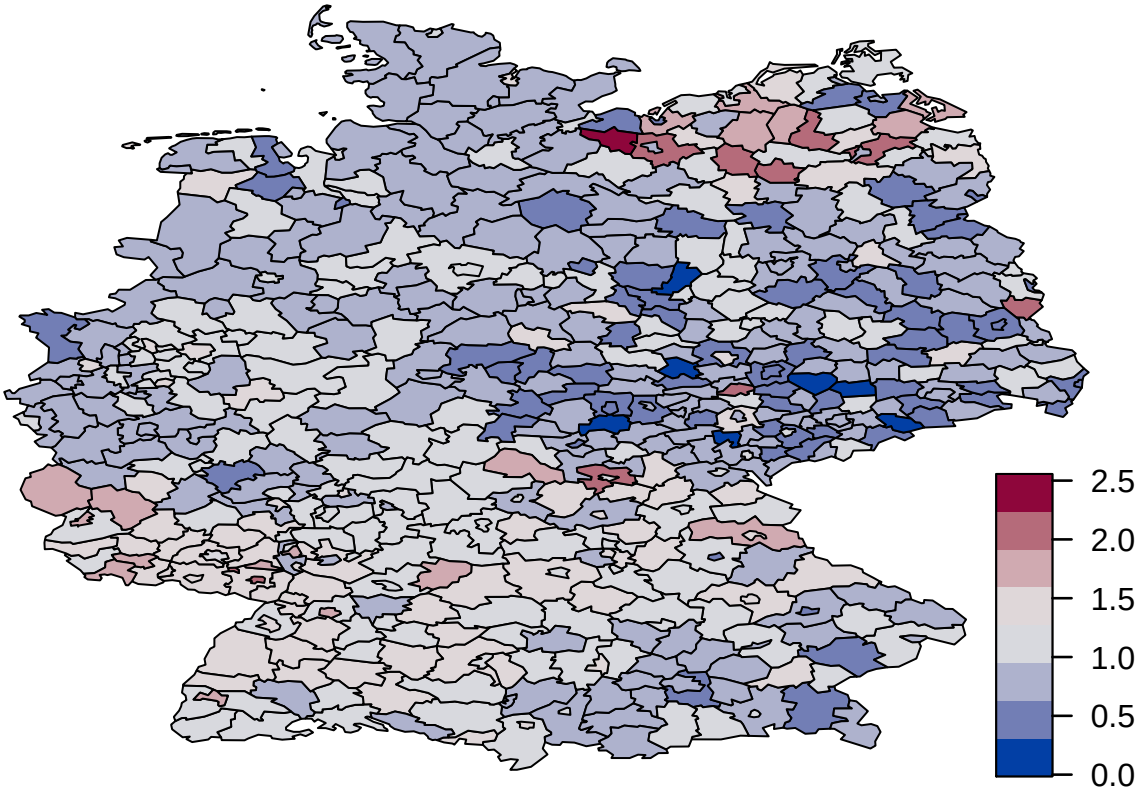
We are going to assume $\alpha_u = \alpha_v = 1$ and $\beta_u = \beta_v = 0.01$. In order to study this spatial structure, we will implement a Gibbs sampler with a Metropolis-Hastings step to sample from the full posterior distribution.

```
# Loading libraries
library(ggplot2)
library(gridExtra)
library(spam)  # load the data
str(Oral)  # see structure of data
```

```
## 'data.frame':    544 obs. of  3 variables:
##  $ Y  : int  18 62 44 12 18 27 20 29 39 21 ...
##  $ E  : num  16.4 45.9 44.7 16.3 26.9 ...
##  $ SMR: num  1.101 1.351 0.985 0.735 0.668 ...
```

```
# ???data.frame???: 544 obs. of 3 variables: $ Y : int 18 62 44 12 18
# 27 20 29 39 21 . . .  $ E : num 16.4 45.9 44.7 16.3 26.9 . . .  $
# SMR: num 1.101 1.351 0.985 0.735 0.668 . . .
attach(Oral)  # allow direct referencing to Y and E
# load some libraries to generate nice map plots
```

```
library(fields, warn.conflict = FALSE)
library(colorspace)
source("additionalFiles/dmvnorm.R")


col <- diverge_hcl(8)  # blue - red
# use a function provided by spam to plot the map together with the
# mortality rates
germany.plot(Oral$Y/Oral$E, col = col, legend = TRUE)
```




```
load("additionalFiles/tma4300_ex2_Rmatrix.Rdata")

# Set seed so that the task can be reproduced
set.seed(42)
```

Exercise 1: Derivations

a)

From the definition of conditional probability and the nature of the assumptions in this project, we know that the full conditional distribtuion has the form

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

By inserting the corresponding probabilities, this becomes

$$\begin{aligned}
\mathrm{p} &\propto \left(\prod_{i=1}^{n} (E_i e^{\eta_i})^{y_i}\, e^{E_i e^{\eta_i}}\right) |\kappa_v \mathrm{I}|^{\frac{1}{2}}\, e^{-\frac{\kappa_v}{2}(\boldsymbol{\eta}-\boldsymbol{u})^T(\boldsymbol{\eta}-\boldsymbol{u})} \kappa_u^{(n-1)/2} e^{-\frac{\kappa_u}{2}\boldsymbol{u}^T\mathbf{R}\boldsymbol{u}} \kappa_u^{\alpha_u-1} e^{-\beta_u\kappa_u} \kappa_v^{\alpha_v-1} e^{-\beta_v\kappa_v} \\
&\propto \kappa_u^{\frac{n-1}{2}+\alpha_u-1} \kappa_v^{\frac{n}{2}+\alpha_v-1} \exp\left\{-\beta_u\kappa_u - \beta_v\kappa_v - \frac{\kappa_v}{2}(\boldsymbol{\eta}-\boldsymbol{u})^T(\boldsymbol{\eta}-\boldsymbol{u}) - \frac{\kappa_u}{2}\boldsymbol{u}^T\mathbf{R}\boldsymbol{u} + \sum_i (y_i\eta_i - E_i e^{\eta_i})\right\}.
\end{aligned}$$

b)

The sum over e^{η_i} in the posterior means that the full conditional of η_i is difficult to sample from. We therefore want to approximate the distribution of $\boldsymbol{\eta}$ it with a multivariate normal in order to use Metropolis-Hastings steps for it. We define the function

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

which has derivatives

$$\begin{aligned}
f'(\eta_i) &= y_i - E_i e^{\eta_i} \\
f''(\eta_i) &= -E_i e^{\eta_i}.
\end{aligned}$$

This yeilds the following Taylor series expansion of f around z_i,

$$\begin{aligned}
\tilde{f}(\eta_i) &= y_i z_i - E_i e^{z_i} + (y_i - E_i e^{z_i})(\eta_i - z_i) + \frac{1}{2}(-E_i e^{z_i})(\eta_i - z_i)^2 \\
&= a(z_i) + b(z_i)\eta_i - \frac{1}{2}c(z_i)\eta_i^2,
\end{aligned}$$

where $a(z_i) = E_i e^{z_i}(z_i - z_i^2/2 - 1)$, $b(z_i) = y_i + E_i e^{z_i}(z_i - 1)$ and $c(z_i) = E_i e^{z_i}$.

c)

As $\mathrm{p}(\theta_i|\boldsymbol{\theta}_{\setminus i}, \boldsymbol{y}) \propto \mathrm{p}(\boldsymbol{\theta}|\boldsymbol{y})$, we can quite easily find the full conditionals from the posterior. Using this, we see that

$$\mathrm{p}(\kappa_u|\boldsymbol{y}, \kappa_v, \boldsymbol{\eta}, \boldsymbol{u}) \propto \kappa_u^{(n-1)/2+\alpha_u-1} e^{-(\beta_u + \frac{1}{2}\boldsymbol{u}^T\mathbf{R}\boldsymbol{u})\kappa_u}.$$

We recognise this as the core of a gamma distribution which means that the full conditional density of κ_u is gamma$(\frac{n-1}{2} + \alpha_u, \beta_u + \frac{1}{2}\boldsymbol{u}^T\mathbf{R}\boldsymbol{u})$. By the same reasoning, we see that gamma$(\frac{n}{2} + \alpha_v, \beta_v + \frac{1}{2}(\boldsymbol{\eta}-\boldsymbol{u})^T(\boldsymbol{\eta}-\boldsymbol{u}))$ is the full conditional density of κ_v. Similarly

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

We recognise this as the canonical form of a multivariate normal distribution. All these distributions are easy to sample from, and can be used in the Gibbs algorithm directly.

The full conditional distribution for $\boldsymbol{\eta}$ however takes the form

$$\mathrm{p}(\boldsymbol{\eta}|\boldsymbol{y},\boldsymbol{u},\kappa_u,\kappa_v) \propto \exp\left\{-\frac{\kappa_v}{2}\left(\boldsymbol{\eta}-\boldsymbol{u}\right)^T\left(\boldsymbol{\eta}-\boldsymbol{u}\right)+\sum_i f(\eta_i)\right\}.$$

This does not correspond to any standard distribution, but by applying the approximation $\tilde{f}(\eta_i)$, we get

$$\begin{aligned}\mathrm{q}(\boldsymbol{\eta}|\boldsymbol{z},\boldsymbol{y},\boldsymbol{u},\kappa_u,\kappa_v) &\propto \exp\left\{-\frac{\kappa_v}{2}\boldsymbol{\eta}^T\boldsymbol{\eta}+\kappa_v\boldsymbol{\eta}^T\boldsymbol{u}-\frac{1}{2}\boldsymbol{\eta}^T\mathrm{diag}(c(\boldsymbol{z}))\boldsymbol{\eta}+\boldsymbol{\eta}^T b(\boldsymbol{z})\right\}\\ &= \exp\left\{-\frac{1}{2}\boldsymbol{\eta}^T\Big(\kappa_v\mathbf{I}+\mathrm{diag}(c(\boldsymbol{z}))\Big)\boldsymbol{\eta}+\boldsymbol{\eta}^T(\kappa_u\boldsymbol{u}+b(\boldsymbol{z}))\right\},\end{aligned}$$

where $\boldsymbol{z}=[z_1,...,z_n]^T$ is the point around which we Taylor expand f, $b(\boldsymbol{z})=[b(z_1),...,b(z_n)]^T$ and $c(\boldsymbol{z})=[c(z_1),...,c(z_n)]^T$. q is the canonical form of a multivariate normal distribution, and can be used for Metropolis-Hastings steps for $\boldsymbol{\eta}$.

Exercise 2: Implementation of the MCMC sampler

Before we can implement the Metropolis-Hastings part of the sampler, we need to simplfy the expression for the acceptance propability α. If we first consider the ratio between true the full conditionals of $\boldsymbol{\eta}^*$ and $\boldsymbol{\eta}$ we can simply insert into the expression found in 1c)

$$\frac{\mathrm{p}(\boldsymbol{\eta}^*|\boldsymbol{y},\boldsymbol{u},\kappa_u,\kappa_v)}{\mathrm{p}(\boldsymbol{\eta}|\boldsymbol{y},\boldsymbol{u},\kappa_u,\kappa_v)} = \exp\left\{-\frac{\kappa_v}{2}\boldsymbol{\eta}^{*T}\boldsymbol{\eta}^*+\boldsymbol{\eta}^{*T}(\kappa_v\boldsymbol{u}+\boldsymbol{y})-\exp(\boldsymbol{\eta}^*)^T\mathbf{E}+\frac{\kappa_v}{2}\boldsymbol{\eta}^T\boldsymbol{\eta}-\boldsymbol{\eta}^T(\kappa_v\boldsymbol{u}+\boldsymbol{y})+\exp(\boldsymbol{\eta})^T\mathbf{E}\right\}.$$

Here $\boldsymbol{\eta}^*$ is the proposed m'th step, $\boldsymbol{\eta}$ the value of the $(m-1)$'th step while $\boldsymbol{u}$, κ_u and κ_v are the m'th step values. The ratio between the proposal distributions is possible to calculate using the function `dmvnorm.canonical()`. Checking for acceptance therefore consists of evaluating the log of the above equation, finding the values for q from `dmvnorm.canonical()` and then cheking wether the log of your decision variable is smaller than the resulting quantity.

(this is cute and all, but probably belongs in an appendix as it is not used)

The ratio between the proposal distributions can also be found by insertion

$$\begin{aligned}\frac{\mathrm{q}(\boldsymbol{\eta}|\boldsymbol{\eta}^*,\boldsymbol{y},\boldsymbol{u},\kappa_u,\kappa_v)}{\mathrm{q}(\boldsymbol{\eta}^*|\boldsymbol{\eta},\boldsymbol{y},\boldsymbol{u},\kappa_u,\kappa_v)} =& \frac{\left|\kappa_v\mathrm{I}+\mathrm{diag}(c(\boldsymbol{\eta}^*))\right|^{\frac{1}{2}}}{\left|\kappa_v\mathrm{I}+\mathrm{diag}(c(\boldsymbol{\eta}))\right|^{\frac{1}{2}}}\cdot\\ &\exp\Big\{-\frac{1}{2}\boldsymbol{\eta}^T\Big(\kappa_v\mathrm{I}+\mathrm{diag}(c(\boldsymbol{\eta}^*))\Big)\boldsymbol{\eta}+\boldsymbol{\eta}^T(\kappa_u\boldsymbol{u}+b(\boldsymbol{\eta}^*))+\\ &\frac{1}{2}\boldsymbol{\eta}^{*T}\Big(\kappa_v\mathrm{I}+\mathrm{diag}(c(\boldsymbol{\eta}))\Big)\boldsymbol{\eta}^*-\boldsymbol{\eta}^{*T}(\kappa_u\boldsymbol{u}+b(\boldsymbol{\eta}))\Big\}.\end{aligned}$$

(dette er stygt, men jeg vet ikke hvordan det kan bli penere). Multiplying these two ratios and using the fact that $b(\boldsymbol{z})=\boldsymbol{y}+\mathrm{diag}(c(\boldsymbol{z}))\boldsymbol{z}-c(\boldsymbol{z})$ and that $\Sigma c(\boldsymbol{z})=\exp(\boldsymbol{z})^T\mathbf{E}$, we get

$$\alpha=\min\left\{1,\frac{\prod_i(\kappa_v+c(\eta_i^*))}{\prod_i(\kappa_v+c(\eta_i))}\exp\left[c(\boldsymbol{\eta})^T\Big(\mathrm{diag}(\boldsymbol{\eta}^*)(\frac{1}{2}\boldsymbol{\eta}^*-\boldsymbol{\eta})+\vec{1}\Big)-c(\boldsymbol{\eta}^*)^T\Big(\mathrm{diag}(\boldsymbol{\eta})(\frac{1}{2}\boldsymbol{\eta}-\boldsymbol{\eta}^*)+\vec{1}\Big)\right]\right\}$$

```r
# Define functions to be used in the MCMC

# The c-function from the
c_func = function(z, E) {
    exp(z) * E
}

b_func = function(z, y, E) {
    y + c_func(z, E) * (z - 1)
}

drawKappaU = function(n, alpha_u, beta_u, u, R) {
    rgamma(1, shape = (n - 1)/2 + alpha_u, rate = beta_u + 0.5 * t(u) %*%
        R %*% u)
}

drawKappaV = function(n, alpha_v, beta_v, eta, u) {
    rgamma(1, shape = n/2 + alpha_v, rate = beta_v + 0.5 * sum((eta -
        u)^2))
}

drawU = function(n, kappa_u, kappa_v, eta, R) {
    t(rmvnorm.canonical(1, kappa_v * eta, kappa_u * R + diag.spam(kappa_v,
        n, n)))
}

drawEta = function(n, z, y, kappa_v, kappa_u, u, E) {
    t(rmvnorm.canonical(1, kappa_v * u + b_func(z, y, E), diag.spam(as.vector(c_func(z,
        E))) + diag.spam(kappa_v, n, n)))
}

logFullCondEta = function(eta, kappa_v, u, y, E) {
    -kappa_v * 0.5 * t(eta) %*% eta + t(eta) %*% u * kappa_v + t(eta) %*%
        y - t(exp(eta)) %*% E
}

# Not in use anymore:
calculateLogAcceptanceProb = function(etaProp, etaPrev, kappa_v, E) {
    c_prop = c_func(etaProp, E)
    c_prev = c_func(etaPrev, E)
    logProb = (sum(log(kappa_v + c_prop)) - sum(log(kappa_v + c_prev)))/2 +
        t(c_prev * etaProp) %*% (0.5 * etaProp - etaPrev) - t(c_prop *
        etaPrev) %*% (0.5 * etaPrev - etaProp) + sum(c_prev) - sum(c_prop) +
        t(etaProp) %*% c_prev - t(etaPrev) %*% c_prop
}
```

Now we implement the MCMC algorithm. It updates each of the components at a time. That is, it follows the Gibbs sampler. For $\boldsymbol{\eta}$, a metropolis-hastings step is needed, as the full conditional is not on a form that can easily be sampled from in itself. The proposal density is a gaussian density found from Taylor expansion around the value in the previous step.

```r
# MCMC function
MCMCOral = function(M, y, starting_values, alpha_u, beta_u, alpha_v,
    beta_v, E, R) {
```

```
# Get computing time
time_start = Sys.time()

# Fetching starting values
eta_prev = starting_values[["eta"]]
u = starting_values[["u"]]
kappa_u = starting_values[["kappa_u"]]
kappa_v = starting_values[["kappa_v"]]

n = length(y)

# Create matrices to contain all steps
eta_values = matrix(nrow = n, ncol = M)
u_values = matrix(nrow = n, ncol = M)
kappa_u_vals = vector(length = M)
kappa_v_vals = vector(length = M)

eta_values[, 1] = eta_prev
u_values[, 1] = u
kappa_v_vals[1] = kappa_v
kappa_u_vals[1] = kappa_u

acceptance_rates = vector(length = M - 1)
num_accepted = 0

for (i in 2:M) {
    # Generate next step for u and kappas
    kappa_u = drawKappaU(n, alpha_u, beta_u, u, R)
    kappa_v = drawKappaV(n, alpha_v, beta_v, eta_prev, u)
    u = drawU(n, kappa_u, kappa_v, eta_prev, R)

    # prpose eta
    eta = drawEta(n, eta_prev, y, kappa_v, kappa_u, u, E)

    # generate acceptance variable
    decision_var = runif(1)

    # check for acceptance log_acceptance =
    # calculateLogAcceptanceProb(eta, eta_prev, kappa_v, E)
    p_eta = logFullCondEta(eta, kappa_v, u, y, E)
    p_eta_prev = logFullCondEta(eta_prev, kappa_v, u, y, E)
    q_eta = dmvnorm.canonical(eta, kappa_v * u + b_func(eta_prev,
        y, E), diag.spam(as.vector(c_func(eta_prev, E) + kappa_v)),
        log = TRUE)
    q_eta_prev = dmvnorm.canonical(eta_prev, kappa_v * u + b_func(eta,
        y, E), diag.spam(as.vector(c_func(eta, E)) + kappa_v), log = TRUE)
    log_acceptance = p_eta - p_eta_prev + q_eta_prev - q_eta


    # Using log_acceptance further
    acceptance_rates[i] = min(1, exp(log_acceptance))

    # Changed from log_acceptance to acceptance_rates[i]:
```

```
        if (decision_var < acceptance_rates[i]) {
            eta_prev = eta
            num_accepted = num_accepted + 1
        }

        # Update value matrix
        eta_values[, i] = eta_prev
        u_values[, i] = u
        kappa_v_vals[i] = kappa_v
        kappa_u_vals[i] = kappa_u
    }
    time_end = Sys.time()
    # Return all values in a list
    return(list(eta = eta_values, u = u_values, kappa_u = kappa_u_vals,
        kappa_v = kappa_v_vals, acceptance_rates = acceptance_rates,
        num_accepted = num_accepted, time = difftime(time_end, time_start,
            units = "secs")))
}
```

Known values, the number of samples and initial values are set before we run the sampler.

```
# Set parameters
M = 50000
y = Oral$Y
E = Oral$E
alpha_u = 1
alpha_v = 1
beta_u = 0.01
beta_v = 0.01

# Set initial values
n = length(y)
u_start = matrix(0, ncol = 1, nrow = n)
eta_start = matrix(0, nrow = n, ncol = 1)
kappa_u_start = 100
kappa_v_start = 100
starting_values = list(eta = eta_start, u = u_start, kappa_u = kappa_u_start,
    kappa_v = kappa_v_start)
```

```
# Run function. set.seed(4300) samples =
# MCMCOral(M,y,starting_values,alpha_u,beta_u,alpha_v,beta_v,E,R)

# Save and load, so we don't need to do it over and over again
# save(samples, file = 'MCMCsamples.RData', ascii=TRUE)
load("MCMCsamples.RData")
M = length(samples$kappa_u)
```

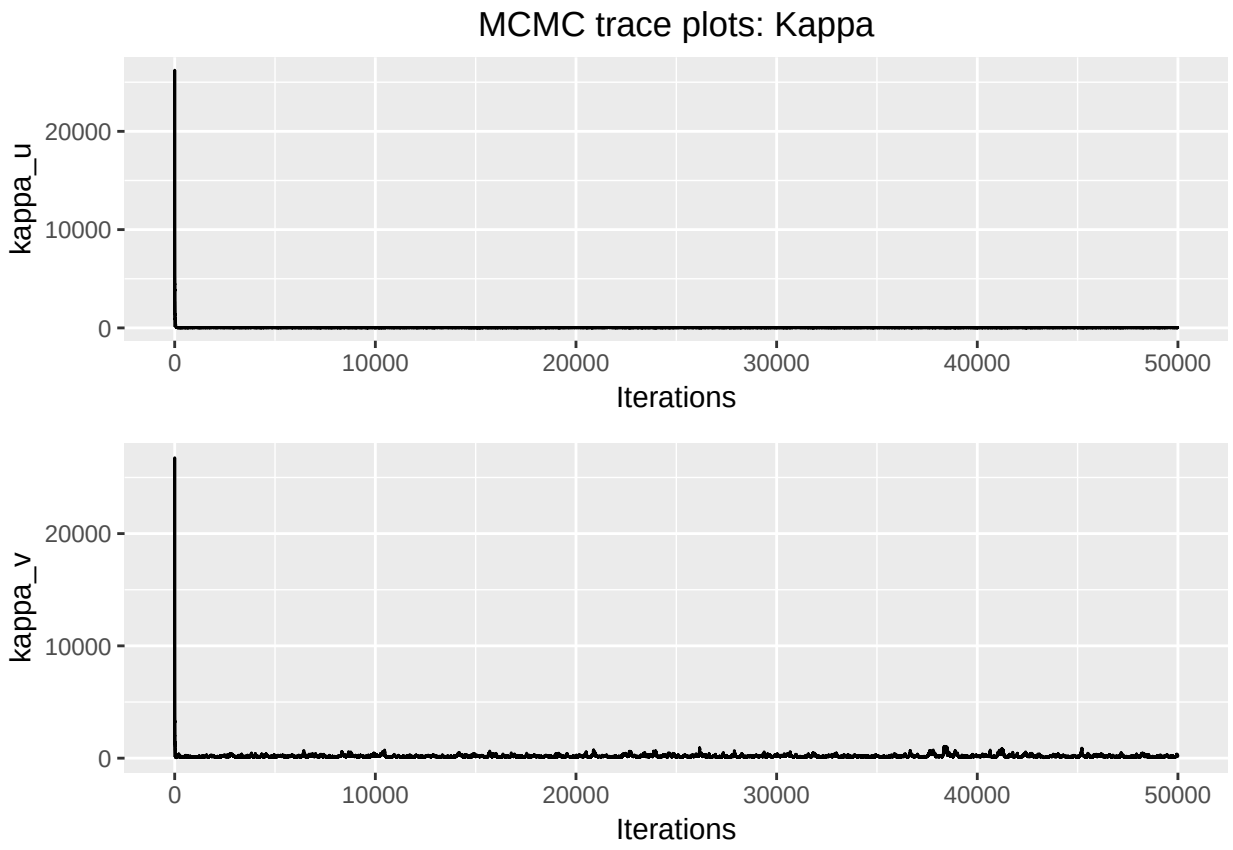
Exercise 3: Convergence diagnostics

We will now do some analysis of the result of the MCMC. First, we will obtain some diagnostic summaries for a set of the parameters. We will try to conclude whether the chain has converged to the target/stationary distribuition.

a) Trace plots

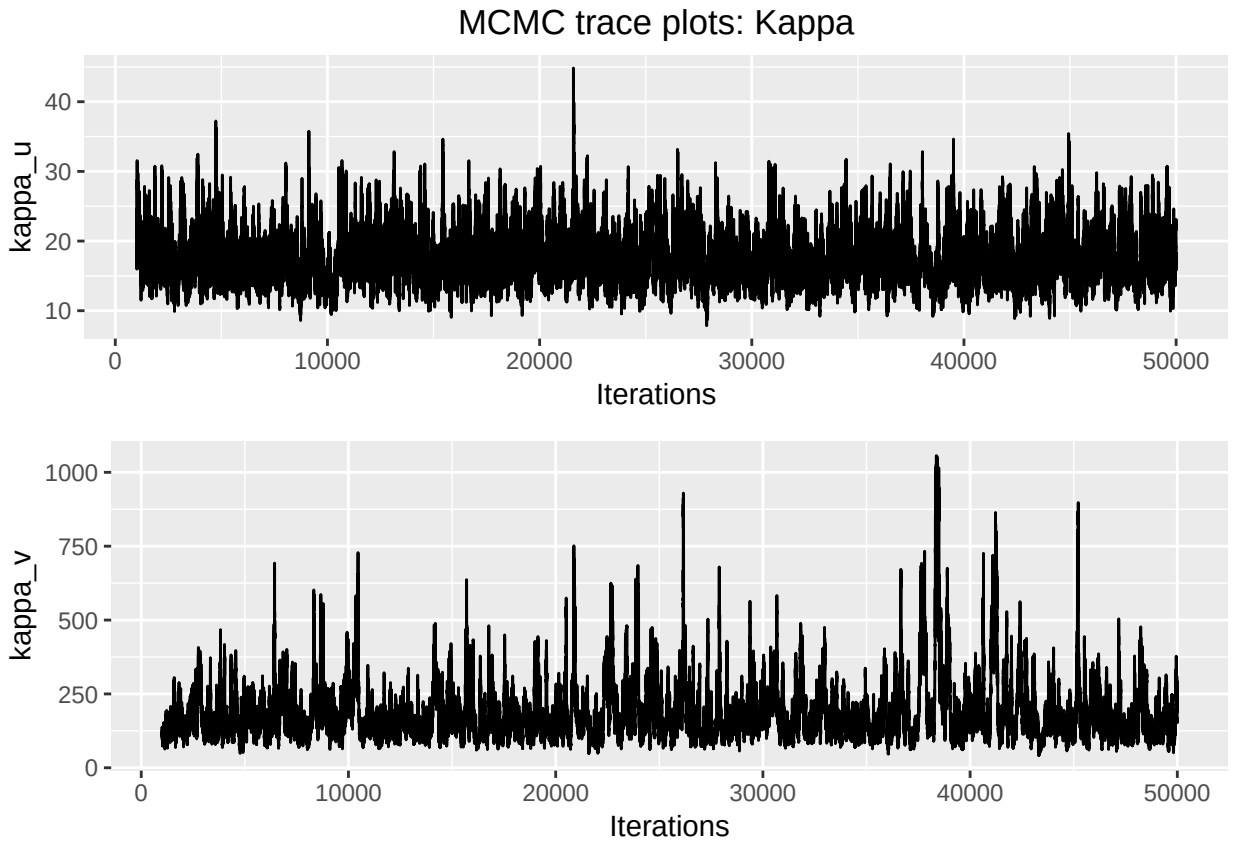
First, we look at the trace plots for κ_u and κ_v. Figure **x** shows all samples, including the burn-in period in the beginning. In figure **y** the burn-in period is disregarded, to show the variability in the parameters, and figure **z** shows just a small part of the samples, and the individual steps in the algorithm is more visible here.

```
# x Trace plots Kappa
start = 1
trace.kappa_u = ggplot(data.frame(kappa = samples$kappa_u[start:M]),
    aes(x = start:M, y = kappa)) + geom_line() + ggtitle("MCMC trace plots: Kappa") +
    theme(plot.title = element_text(hjust = 0.5)) + xlab("Iterations") +
    ylab("kappa_u")

trace.kappa_v = ggplot(data.frame(kappa = samples$kappa_v[start:M]),
    aes(x = start:M, y = kappa)) + geom_line() + xlab("Iterations") +
    ylab("kappa_v")

trace.kappa_plot = grid.arrange(grobs = list(trace.kappa_u, trace.kappa_v),
    ncol = 1)
```

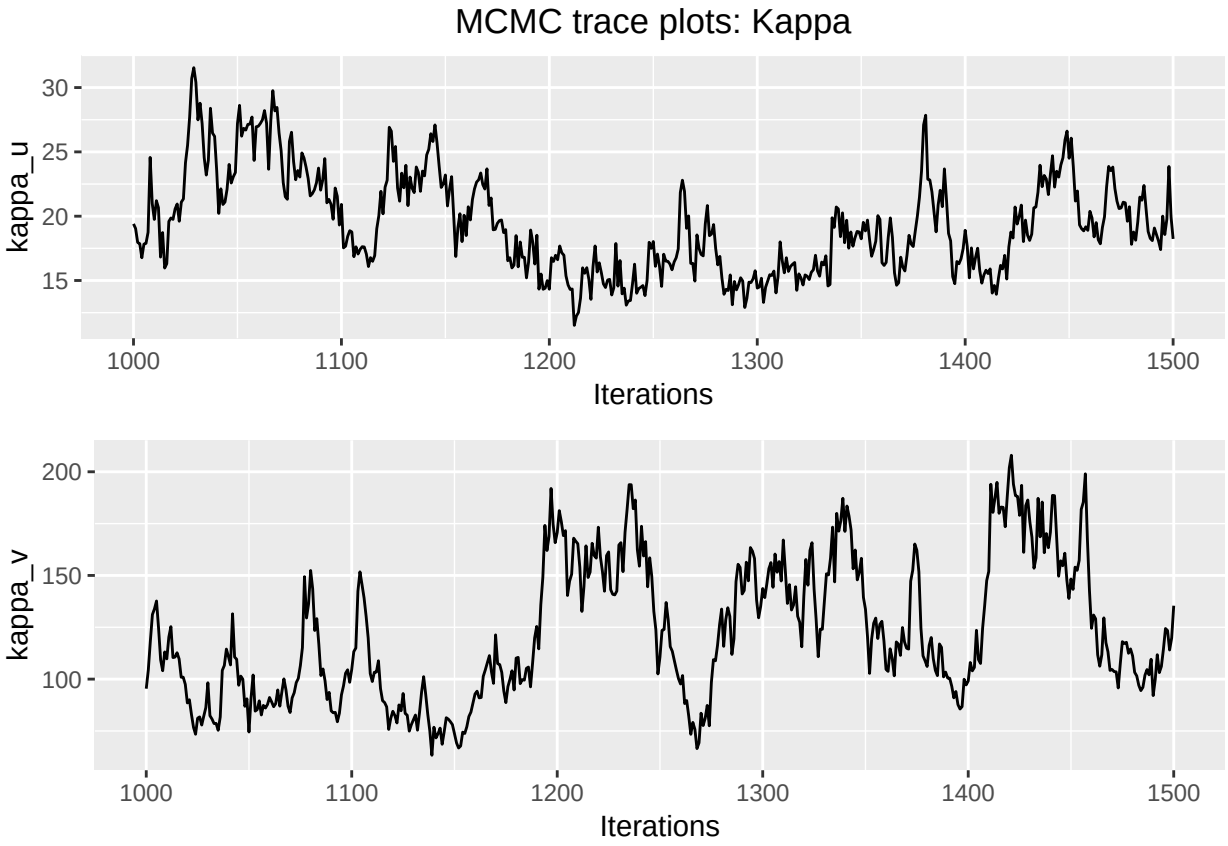



```
# y Trace plots Kappa
start = 1000
trace.kappa_u = ggplot(data.frame(kappa = samples$kappa_u[start:M]),
    aes(x = start:M, y = kappa)) + geom_line() + ggtitle("MCMC trace plots: Kappa") +
    theme(plot.title = element_text(hjust = 0.5)) + xlab("Iterations") +
    ylab("kappa_u")

trace.kappa_v = ggplot(data.frame(kappa = samples$kappa_v[start:M]),
```

```
    aes(x = start:M, y = kappa)) + geom_line() + xlab("Iterations") +
    ylab("kappa_v")

trace.kappa_plot = grid.arrange(grobs = list(trace.kappa_u, trace.kappa_v),
    ncol = 1)
```



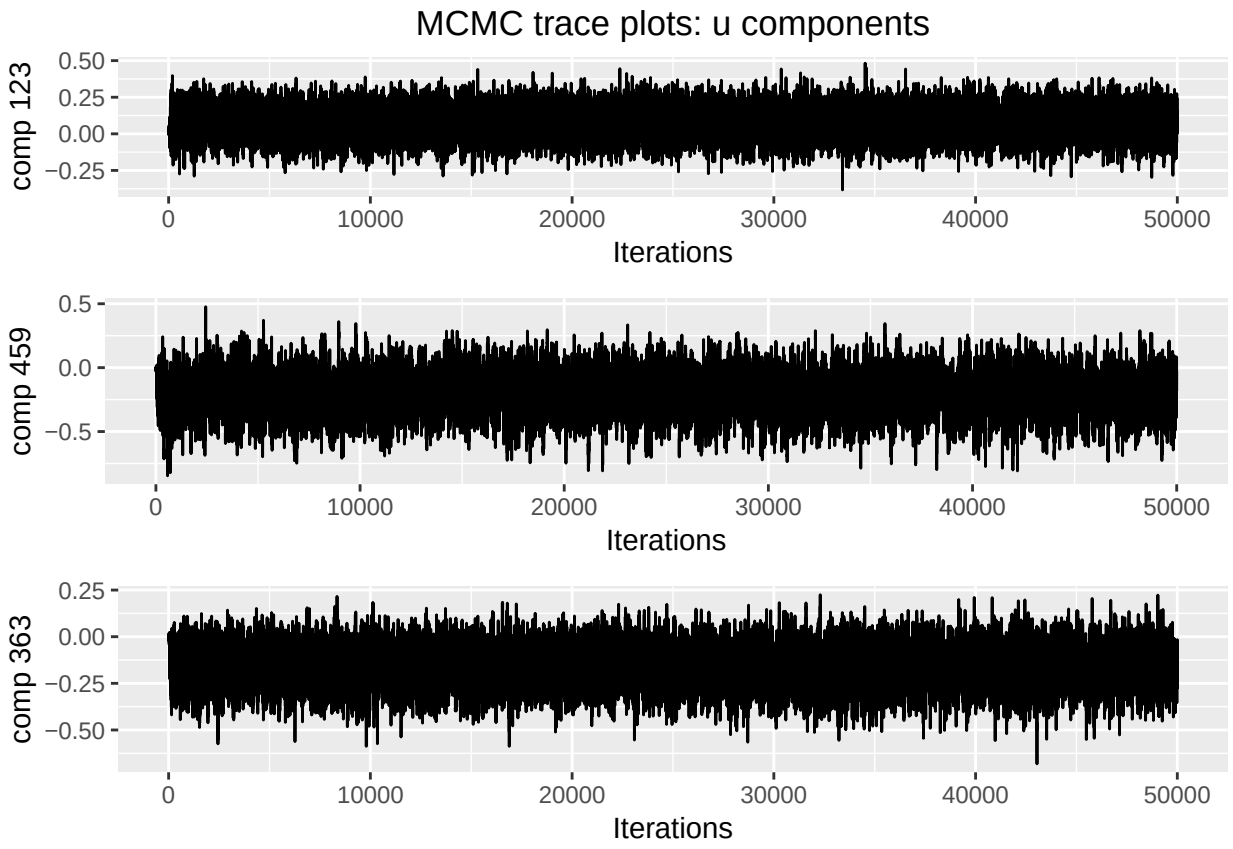

```
# z Trace plots Kappa
start = 1000
stop = 1500
trace.kappa_u = ggplot(data.frame(kappa = samples$kappa_u[start:stop]),
    aes(x = start:stop, y = kappa)) + geom_line() + ggtitle("MCMC trace plots: Kappa") +
    theme(plot.title = element_text(hjust = 0.5)) + xlab("Iterations") +
    ylab("kappa_u")

trace.kappa_v = ggplot(data.frame(kappa = samples$kappa_v[start:stop]),
    aes(x = start:stop, y = kappa)) + geom_line() + xlab("Iterations") +
    ylab("kappa_v")

trace.kappa_plot = grid.arrange(grobs = list(trace.kappa_u, trace.kappa_v),
    ncol = 1)
```



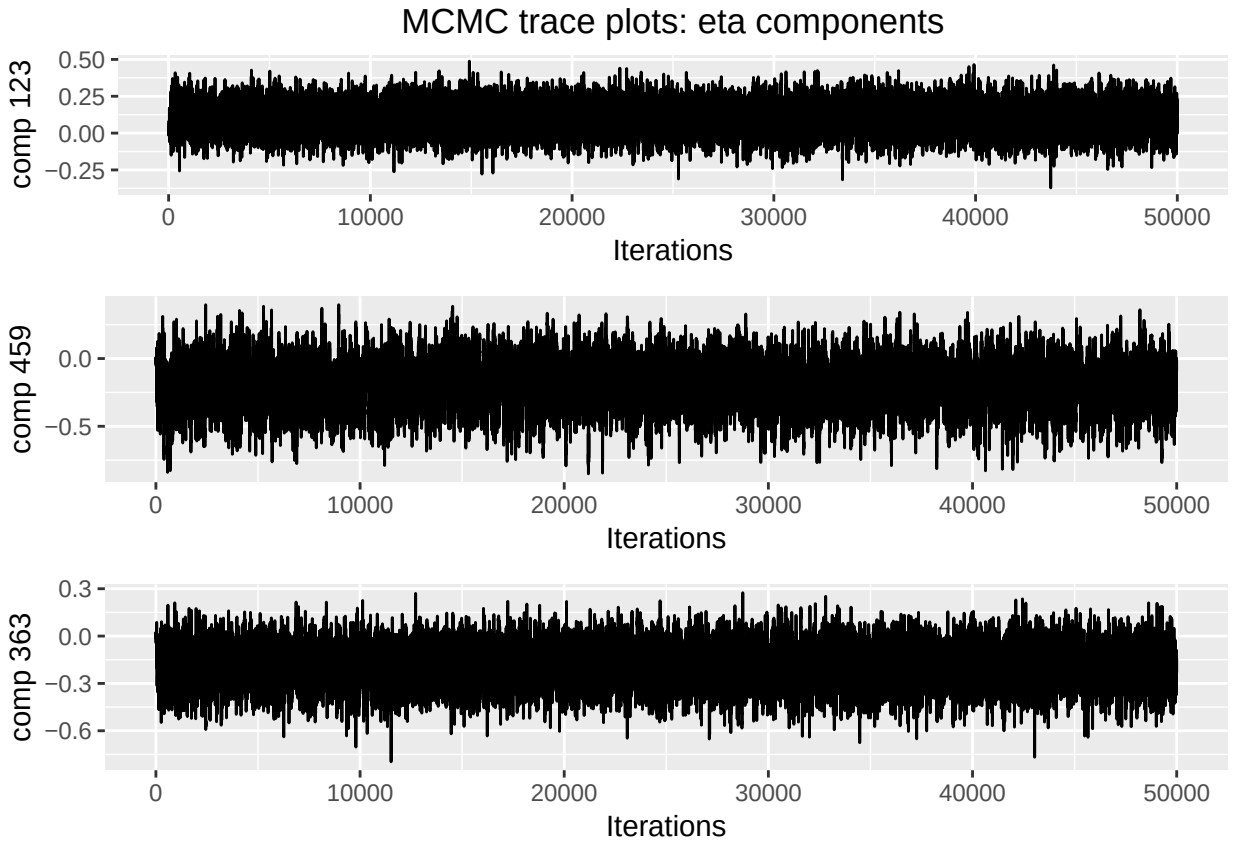
u and eta or u and v?? Since the vectors of $\boldsymbol{\eta}$ and $\boldsymbol{u}$ are of high dimensions, trace plots are only made for a few, randomly chosen components.

```
# Trace plots somecomponents of u and eta Choose components:
set.seed(4301)
comp = ceiling(n * runif(3))
trace.u = list()
trace.eta = list()

# labels
lab = paste("comp", comp)

# Make first plots with title
trace.u[[1]] = ggplot(data.frame(u = samples$u[comp[1], ]), aes(x = 1:M,
    y = u)) + geom_line() + ggtitle("MCMC trace plots: u components") +
    theme(plot.title = element_text(hjust = 0.5)) + xlab("Iterations") +
    ylab(lab[1])

trace.eta[[1]] = ggplot(data.frame(eta = samples$eta[comp[1], ]), aes(x = 1:M,
    y = eta)) + geom_line() + ggtitle("MCMC trace plots: eta components") +
    theme(plot.title = element_text(hjust = 0.5)) + xlab("Iterations") +
    ylab(lab[1])

# Next plots do not need title
for (i in 2:length(comp)) {
    trace.u[[i]] = ggplot(data.frame(u = samples$u[comp[i], ]), aes(x = 1:M,
        y = u)) + geom_line() + xlab("Iterations") + ylab(lab[i])
```

```
    trace.eta[[i]] = ggplot(data.frame(eta = samples$eta[comp[i], ]),
        aes(x = 1:M, y = eta)) + geom_line() + xlab("Iterations") + ylab(lab[i])
}

trace.u_plot = grid.arrange(grobs = trace.u, ncol = 1)
```




```
trace.eta_plot = grid.arrange(grobs = trace.eta, ncol = 1)
```



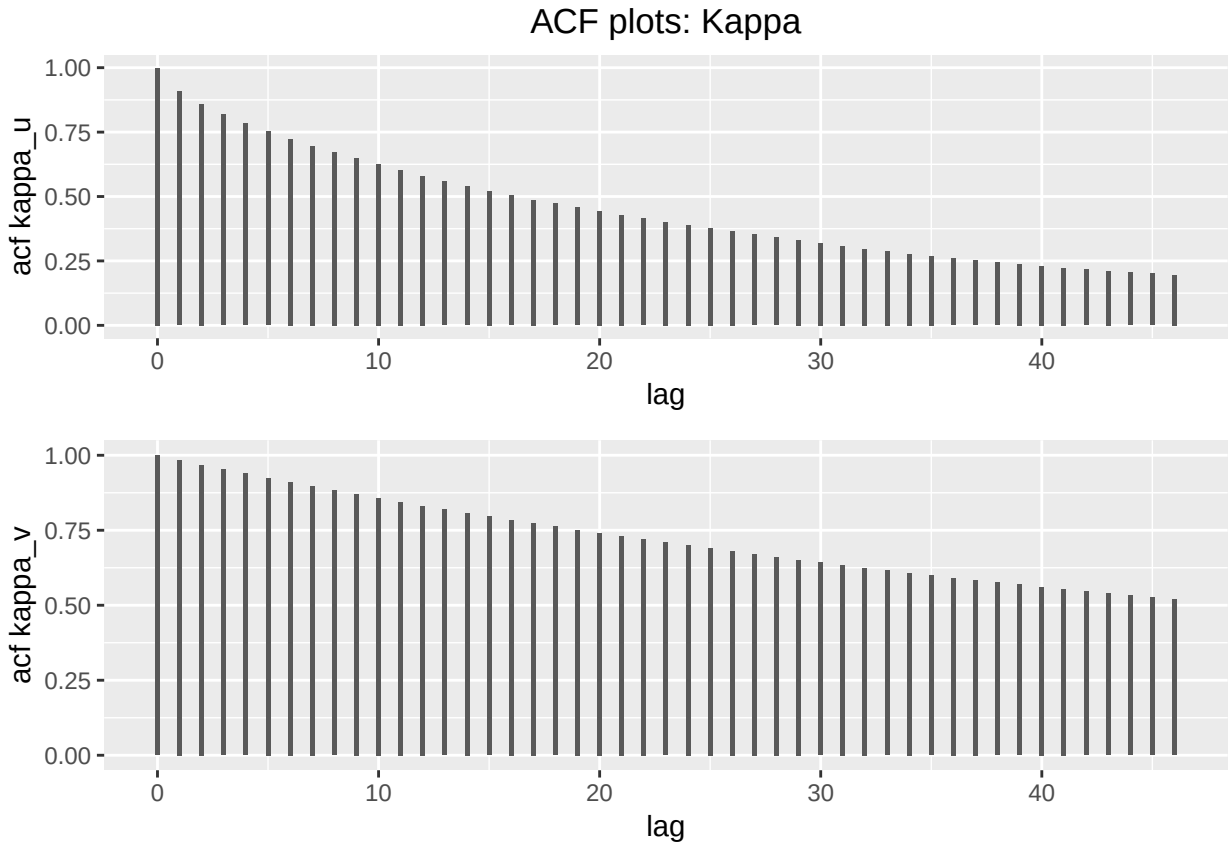
$\boldsymbol{\eta}$ and $\boldsymbol{u}$ seem to have very small burn-in period. To be on the safe side, in the further analysis, the first 1000 samples are disregarded.

```
# Define burn-in area
burnins = 1:1000
```

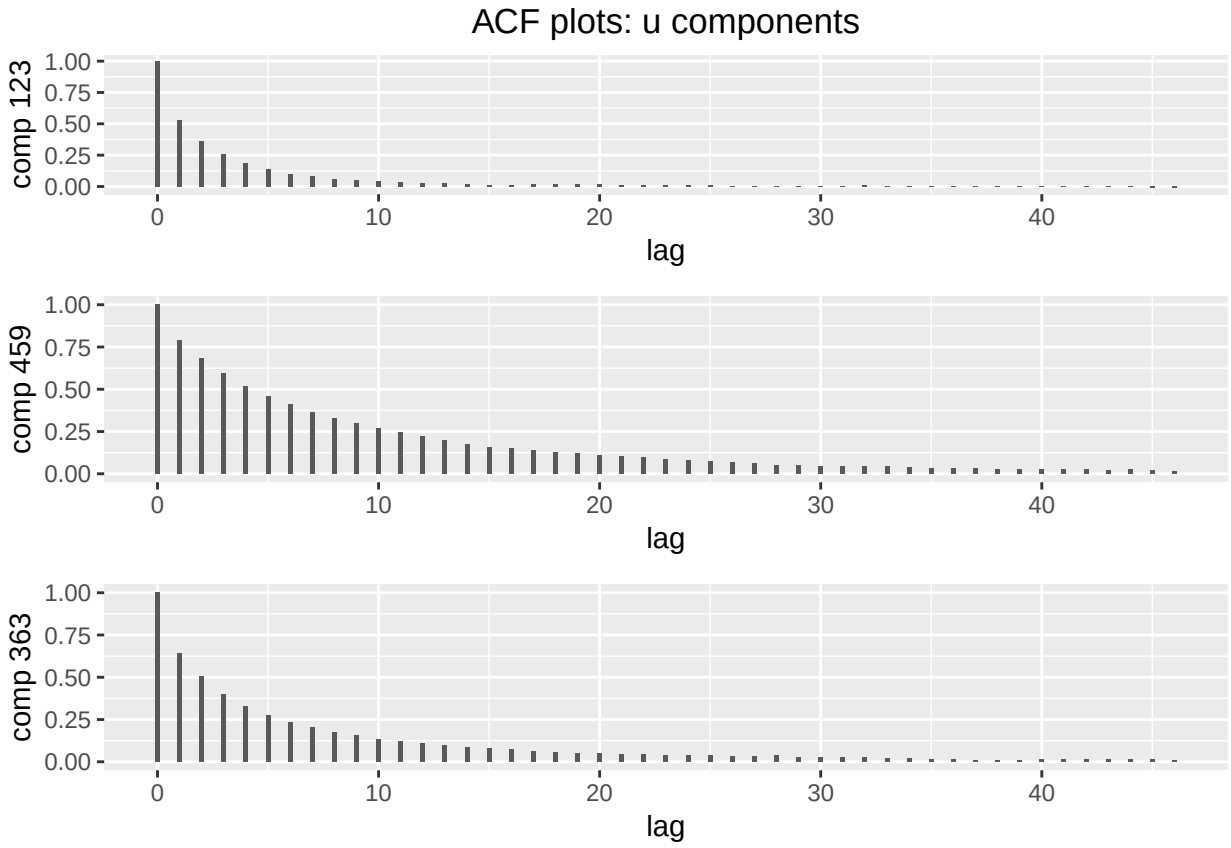
b) Autocorrelation plots

Autocorrelation plots are made for the same components as the trace plots.

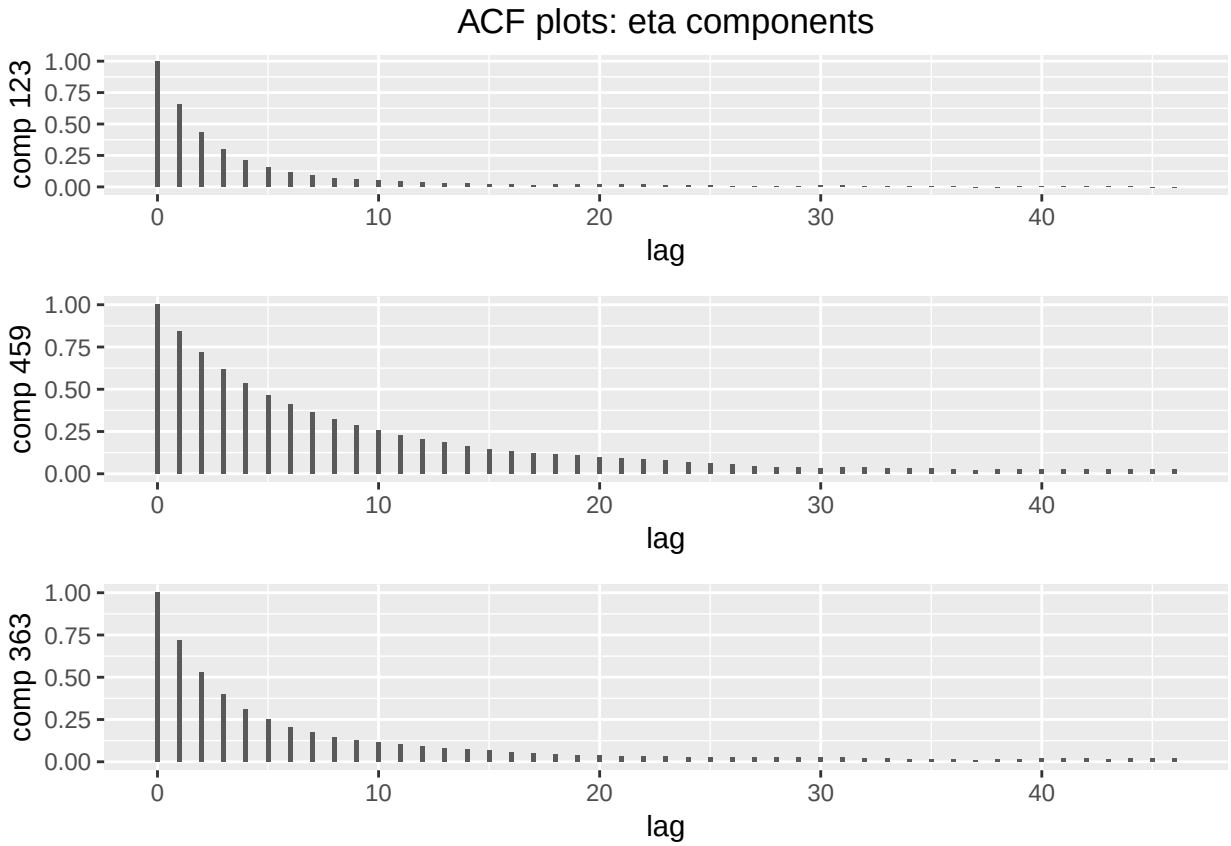
```
acf.kappa_u = acf(samples$kappa_u[-burnins], plot = FALSE)
acf.kappa_v = acf(samples$kappa_v[-burnins], plot = FALSE)

plot1 = ggplot(data.frame(acf = acf.kappa_u$acf, lag = acf.kappa_u$lag),
    aes(x = lag, y = acf)) + geom_col(width = 0.2) + ggtitle("ACF plots: Kappa") +
    theme(plot.title = element_text(hjust = 0.5)) + ylab("acf kappa_u")

plot2 = ggplot(data.frame(acf = acf.kappa_v$acf, lag = acf.kappa_v$lag),
    aes(x = lag, y = acf)) + geom_col(width = 0.2) + ylab("acf kappa_v")

acf.kappa_plot = grid.arrange(grobs = list(plot1, plot2), ncol = 1)
```




```
# acf for u and eta
plot_u = list()
plot_eta = list()

# Make first plots with title
acf.u = acf(samples$u[comp[1], -burnins], plot = FALSE)
plot_u[[1]] = ggplot(data.frame(acf = acf.u$acf, lag = acf.u$lag), aes(x = lag,
    y = acf)) + geom_col(width = 0.2) + ggtitle("ACF plots: u components") +
    theme(plot.title = element_text(hjust = 0.5)) + ylab(lab[1])

acf.eta = acf(samples$eta[comp[1], -burnins], plot = FALSE)
plot_eta[[1]] = ggplot(data.frame(acf = acf.eta$acf, lag = acf.eta$lag),
    aes(x = lag, y = acf)) + geom_col(width = 0.2) + ggtitle("ACF plots: eta components") +
    theme(plot.title = element_text(hjust = 0.5)) + ylab(lab[1])

# Next plots do not need title
for (i in 2:length(comp)) {
    acf.u = acf(samples$u[comp[i], -burnins], plot = FALSE)
    plot_u[[i]] = ggplot(data.frame(acf = acf.u$acf, lag = acf.u$lag),
        aes(x = lag, y = acf)) + geom_col(width = 0.2) + ylab(lab[i])

    acf.eta = acf(samples$eta[comp[i], -burnins], plot = FALSE)
    plot_eta[[i]] = ggplot(data.frame(acf = acf.eta$acf, lag = acf.eta$lag),
        aes(x = lag, y = acf)) + geom_col(width = 0.2) + ylab(lab[i])
}

acf.u_plot = grid.arrange(grobs = plot_u, ncol = 1)
```




```
acf.eta_plot = grid.arrange(grobs = plot_eta, ncol = 1)
```



We see that the steps are quite correlated for the κs. From the trace plot in figure **z**, this tendency can be spotted. The mixing does not seem to be very good; the steps are often quite small, so that it takes quite a lot of steps to explore the whole areas.

Convergence test

geweke.diag() performs a test for convergence of the markov chain. It takes one fraction of the beginning of the chain and one fraction of the end of the MCMC. The burn-in period is disregarded. If the chain has converged to the stationary distribution, the two means should be equal. The geweke test statistic is asymptotically standard normal distributed. The null hypothesis is that the chain has converged.

```
library(coda)

# Make data frame of samples
u_df = as.data.frame(t(samples$u))
eta_df = as.data.frame(t(samples$eta))

# Apply geweke test:
gew.kappa_u = geweke.diag(samples$kappa_u[-burnins])
gew.kappa_v = geweke.diag(samples$kappa_v[-burnins])

gew.u = geweke.diag(u_df[-burnins, ])
gew.eta = geweke.diag(eta_df[-burnins, ])


p_u = 2 * pnorm(abs(gew.u$z), lower.tail = FALSE)
p_eta = 2 * pnorm(abs(gew.eta$z), lower.tail = FALSE)
```

```
p_kappa = c(2 * pnorm(abs(gew.kappa_u$z), lower.tail = FALSE), 2 * pnorm(abs(gew.kappa_v$z),
    lower.tail = FALSE))
```

```
## kappa_u:
```

```
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##  var1
## 1.461
```

```
##   kappa_u:     test stat 1.461278  p-value 0.1439391
```

```
##   kappa_v:     test stat -2.099535  p-value 0.03576979
```

```
##   u comp 123 : test stat 0.2592473  p-value 0.7954444
```

```
##   u comp 459 : test stat 1.653132  p-value 0.09830406
```

```
##   u comp 363 : test stat 1.407897  p-value 0.1591617
```

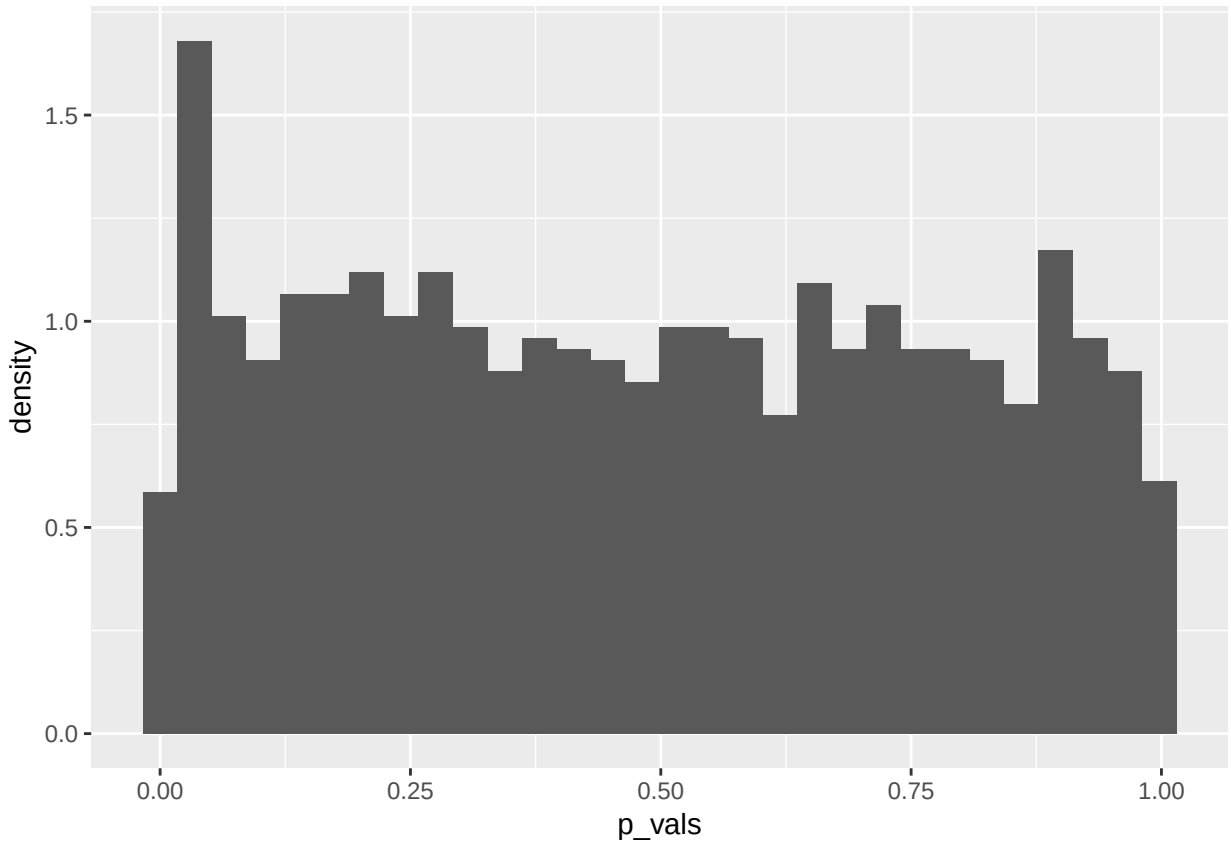
```
## eta comp 123 : test stat 0.4056651  p-value 0.6849887
```

```
## eta comp 459 : test stat 1.757081  p-value 0.07890408
```

```
## eta comp 363 : test stat 0.9271182  p-value 0.3538651
```

We make a histogram of all p-values to see whether they seem to be uniformly distributed, as they would be if the null hypothesis is true and the chain has converged.

```
ggplot(data = data.frame(p_vals = c(p_u, p_eta, p_kappa), var = c(rep(1,
    length = length(p_u)), rep(2, length = length(p_eta)), rep(3, length = length(p_kappa))))) +
    geom_histogram(aes(x = p_vals, y = ..density..))
```



Conclusion? For some components, isolated, we will conclude from the geweke test that the samples have not converged to the target distribution. However, overall, the p-values seem at least not far from uniformly distributed. From the trace plots, it looks like the chain has converged. The autocorrelation plots show quite high correlation for some components, but some correlation cannot be avoided, as the proposed steps depend on the previous steps. Al in all, it seems the MCMC has converged, but of course, we cannot be completely sure.

Exercise 4: Effective sample set

```
effSize.kappa_u = effectiveSize(samples$kappa_u[-burnins])
effSize.kappa_v = effectiveSize(samples$kappa_v[-burnins])
```

interpret the values These values mean that out of the M samples of the MCMC, we "effectively" have 939 and 348 samples from the posterior distributions of κ_u and κ_v, respectively. These are **very low!** values.**??**

How could the MCMC be changed to increas the effective sample size?

```
comp_time = as.numeric(samples$time)
# comp_time = 100 #For testing when the time was not computed
relEffSize.kappa_u = effSize.kappa_u/comp_time
relEffSize.kappa_v = effSize.kappa_v/comp_time
```

```
## Relative effective sample size for kappa_u is  4.870931
```

```
## Relative effective sample size for kappa_v is  1.804706
```

The relaive effective sample size is the effective sample size obtained per second of running the MCMC. This

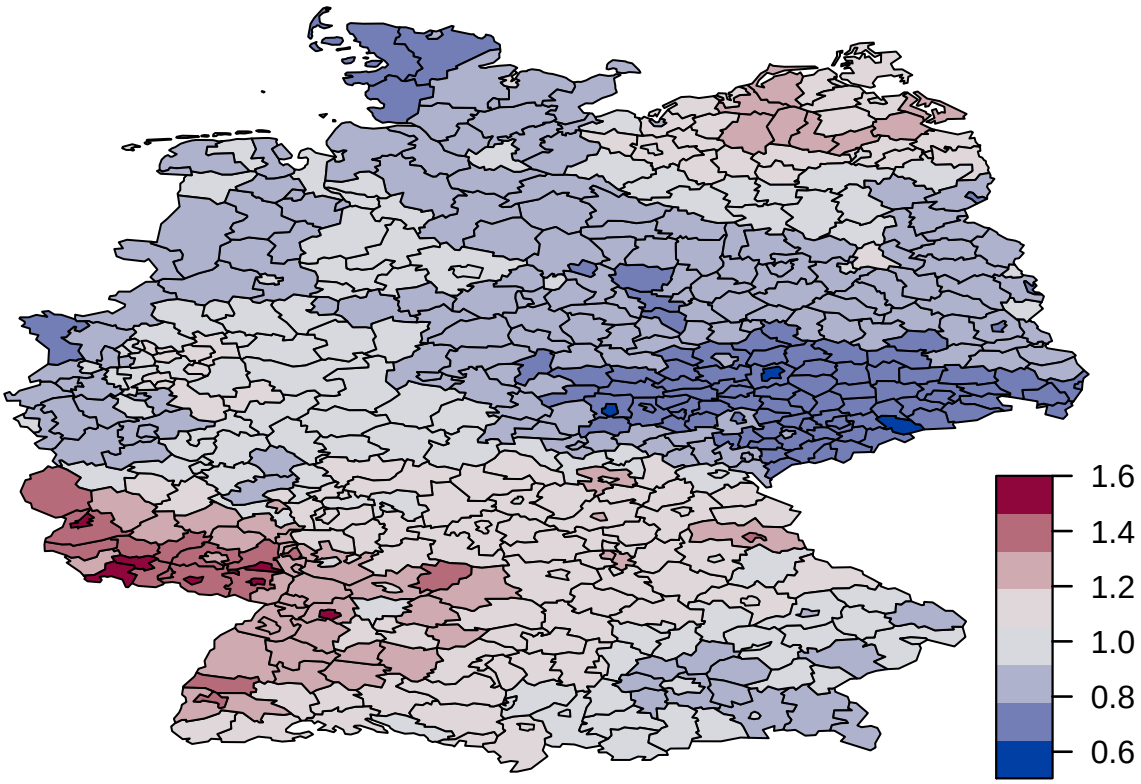
could be interesting, as it gives us some meaure of how efficient the MCMC algorithm is. Low computer time is not good in itself if the algorithm does not draw enough samples from the target distribution. Likewise, large effective sample size is of course desireble, but if the algorithm has to run for an unreasonable long time, some improvement of the algorithm or sampling approach should maybe be considered.

These numbers in themselves might not be too informative, but in relation to other approaches, they will give some measure of which to use.

Exercise 5: Interpretation of result

u or eta?

```
post_median = exp(apply(samples$u, 1, median))

germany.plot(post_median, col = col, legend = TRUE)
```



We see that the general tendencies from the data (y_i/E_i) in the beginning of the report are still present in this posterior distribution, but that the posterior distribution of the expected number of cancers per region is smoother across region. **meeeh**

Exercise 6: INLA

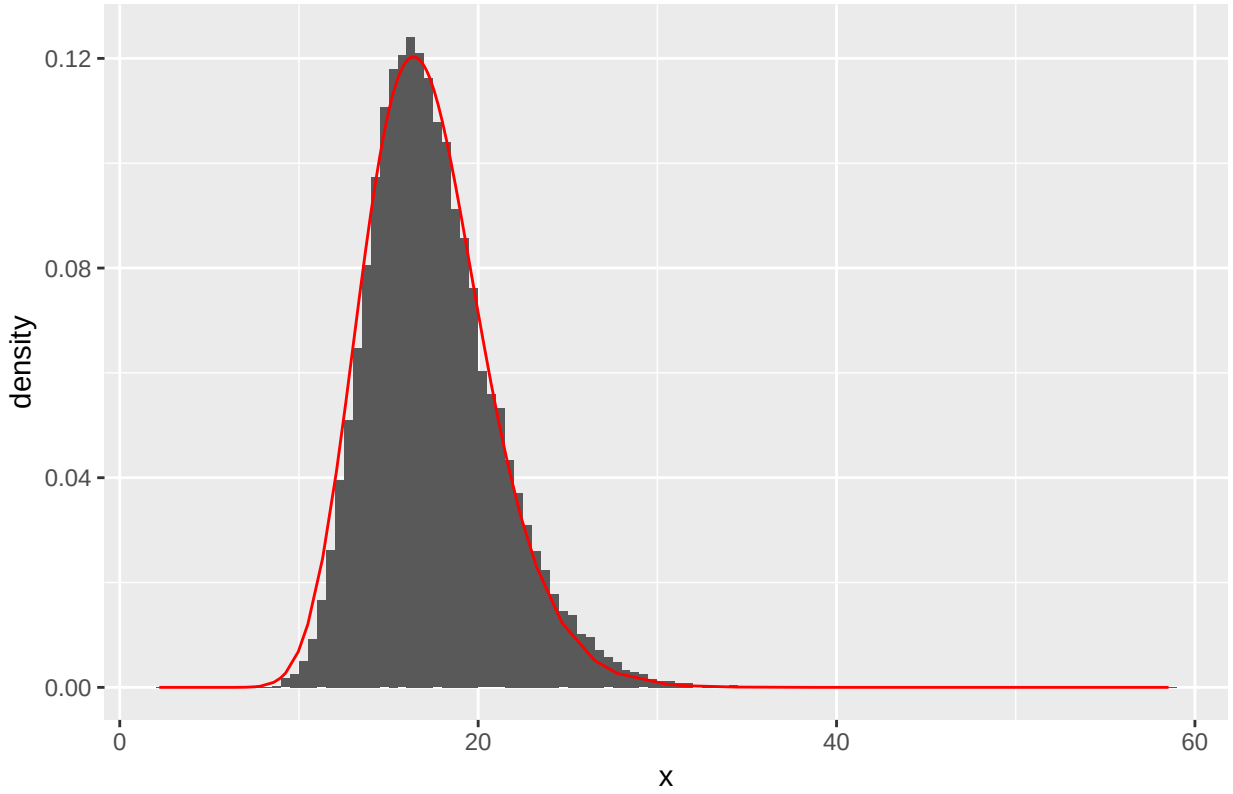
a)

R-INLA can also be used to approximate oral cancer in Germany.

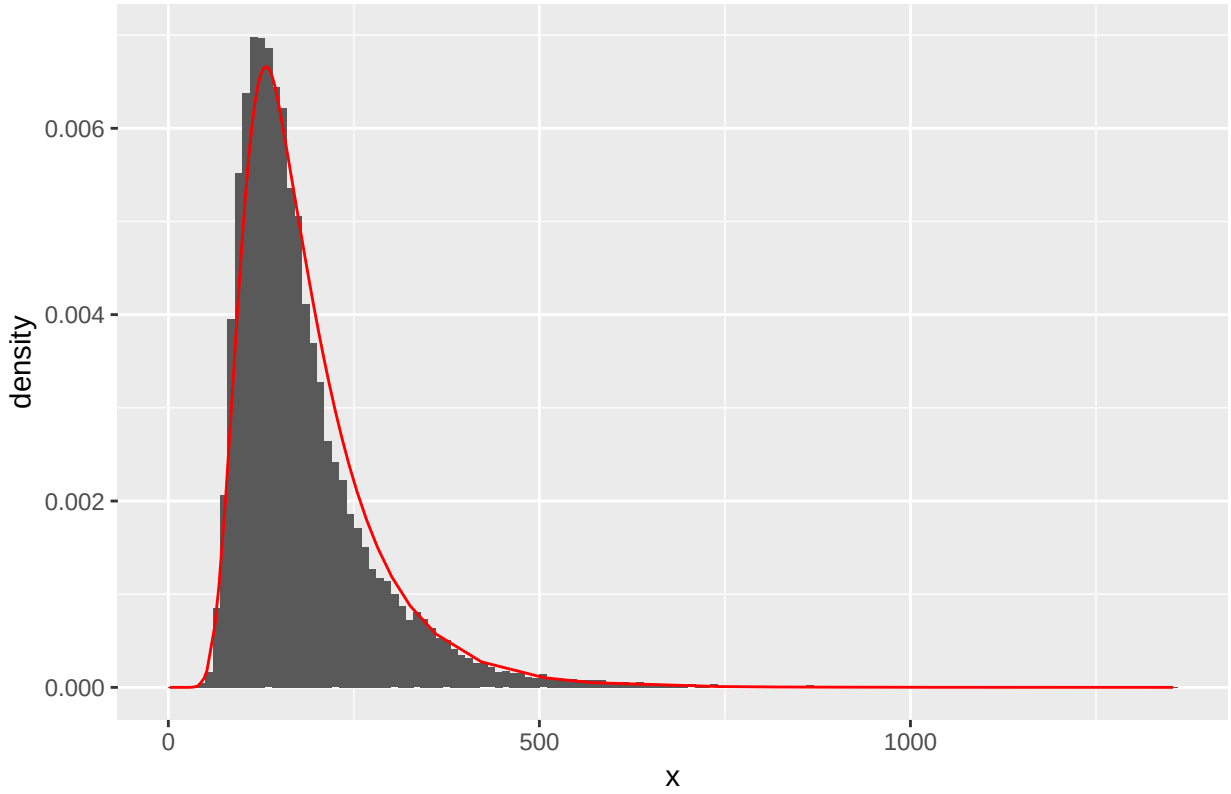
```r
# Build the same model using INLA
library(INLA)
g <- system.file("demodata/germany.graph", package = "INLA")

# Known values and parameters (copied from task2)
y = Oral$Y
E = Oral$E
alpha_u = 1
alpha_v = 1
beta_u = 0.01
beta_v = 0.01
n = length(y)
OralData = cbind(Oral, Region = Germany$region, Region.struct = Germany$region)
```

```r
INLAOral = inla(formula = Y ~ f(Region.struct, model = "besag", graph = g,
    hyper = list(prec = list(param = c(alpha_u, beta_u))), constr = FALSE) +
    f(Region, model = "iid", hyper = list(prec = list(param = c(alpha_v,
        beta_v)))) - 1, family = "poisson", E = E, data = OralData, control.compute = list(dic = TRUE))
```

```r
# Marginal posteriors and histograms
kappa_u_marginal = INLAOral$marginals.hyperpar$`Precision for Region.struct`
kappa_v_marginal = INLAOral$marginals.hyperpar$`Precision for Region`

kappa_u_df = as.data.frame(kappa_u_marginal[-(74:75), ])
kappa_v_df = as.data.frame(kappa_v_marginal[-(70:75), ])

ggplot(data.frame(kappa = samples$kappa_u[-burnins]), aes(kappa)) + geom_histogram(aes(y = ..density..)
    binwidth = 0.5, boundary = 0) + geom_path(aes.inherit = FALSE, data = kappa_u_df,
    aes(x = x, y = y), col = "red") + ggtitle("") + theme(plot.title = element_text(hjust = 0.5)) +
    xlab("x")
```



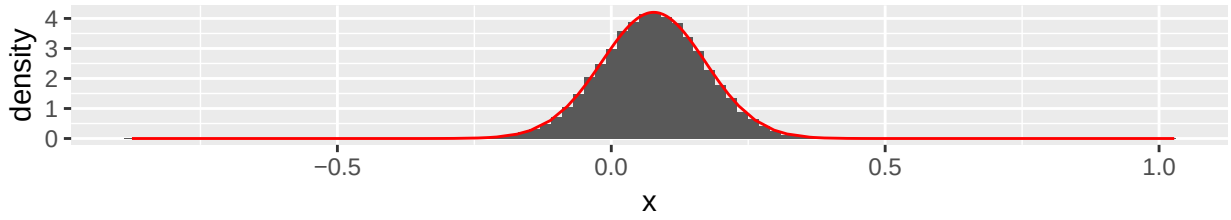

```
ggplot(data.frame(kappa = samples$kappa_v[-burnins]), aes(kappa)) + geom_histogram(aes(y = ..density..)
    binwidth = 10, boundary = 0) + geom_path(aes.inherit = FALSE, data = kappa_v_df,
    aes(x = x, y = y), col = "red") + ggtitle("") + theme(plot.title = element_text(hjust = 0.5)) +
    xlab("x")
```

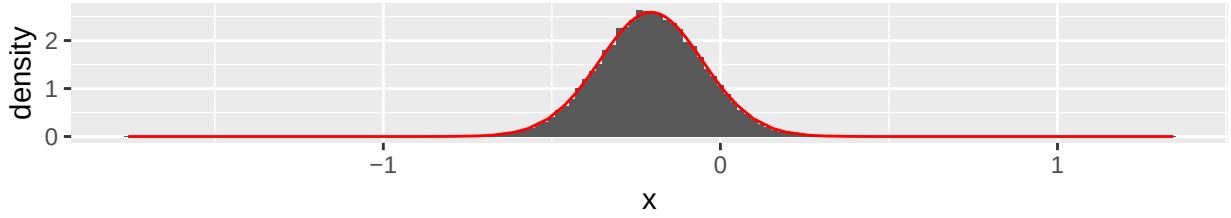


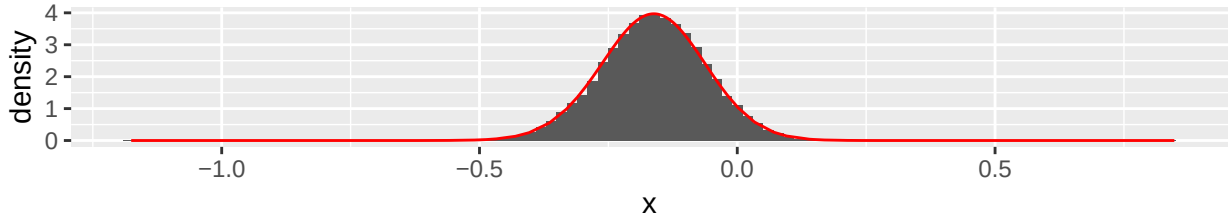

```
# samma_det = inla.hyperpar(INLAOral)
```

```
u_inla = INLAOral$marginals.random$Region.struct[comp]
v_inla = INLAOral$marginals.random$Region[comp]
v_MCMC = samples$eta - samples$u

u_inla_df = as.data.frame(u_inla[[1]])
v_inla_df = as.data.frame(v_inla[[1]])

u_plots = list()
v_plots = list()
binw = 0.02

u_plots[[1]] = ggplot(data.frame(samp = samples$u[comp[1], -burnins]),
    aes(samp)) + geom_histogram(aes(y = ..density..), binwidth = binw) +
    geom_path(aes.inherit = FALSE, data = u_inla_df, aes(x = x, y = y),
        col = "red") + ggtitle("") + theme(plot.title = element_text(hjust = 0.5)) +
    xlab("x")

v_plots[[1]] = ggplot(data.frame(samp = v_MCMC[comp[1], -burnins]), aes(samp)) +
    geom_histogram(aes(y = ..density..), binwidth = binw) + geom_path(aes.inherit = FALSE,
    data = v_inla_df, aes(x = x, y = y), col = "red") + ggtitle("") +
    theme(plot.title = element_text(hjust = 0.5))

for (i in 2:3) {
    u_inla_df = as.data.frame(u_inla[[i]])
    v_inla_df = as.data.frame(v_inla[[i]])

    u_plots[[i]] = ggplot(data.frame(samp = samples$u[comp[i], -burnins]),
```

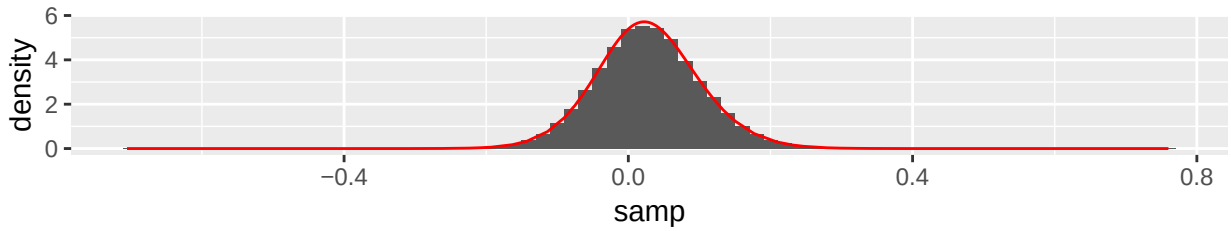
```
        aes(samp)) + geom_histogram(aes(y = ..density..), binwidth = binw) +
        geom_path(aes.inherit = FALSE, data = u_inla_df, aes(x = x, y = y),
            col = "red") + ggtitle("") + theme(plot.title = element_text(hjust = 0.5)) +
        xlab("x")

    v_plots[[i]] = ggplot(data.frame(samp = v_MCMC[comp[i], -burnins]),
        aes(samp)) + geom_histogram(aes(y = ..density..), binwidth = binw) +
        geom_path(aes.inherit = FALSE, data = v_inla_df, aes(x = x, y = y),
            col = "red") + ggtitle("") + theme(plot.title = element_text(hjust = 0.5)) +
        xlab("x")
}

library(gridExtra)
grid.arrange(grobs = u_plots, ncol = 1)
```

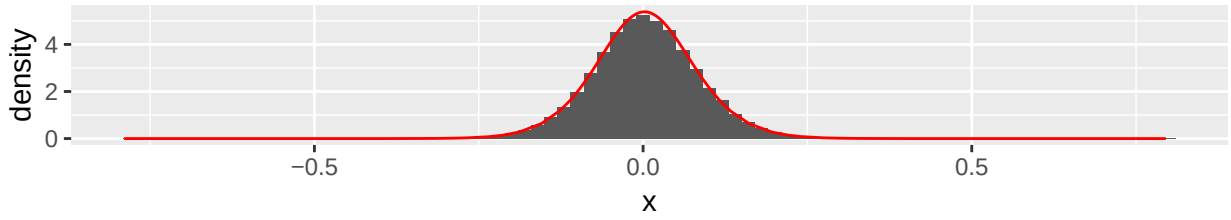


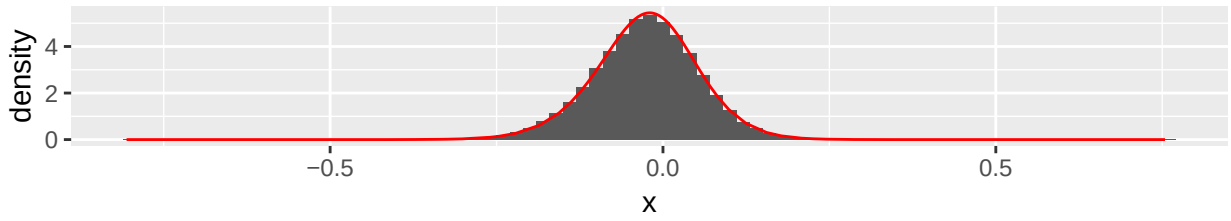




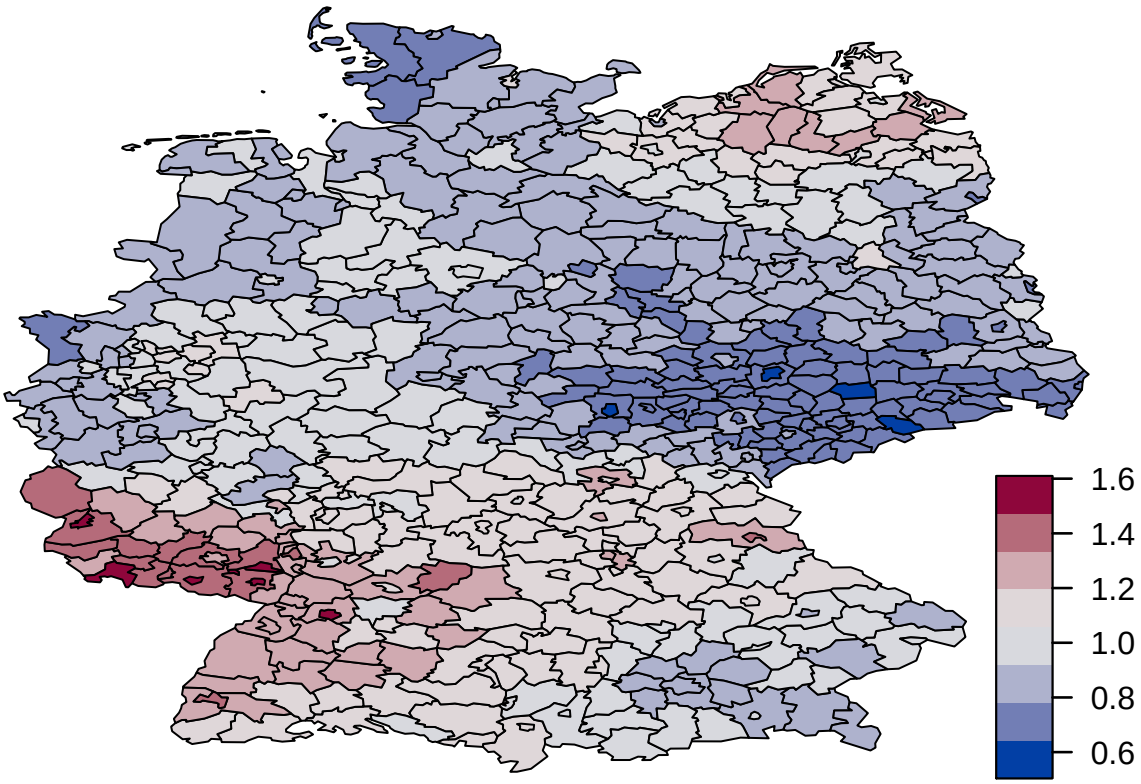

```
grid.arrange(grobs = v_plots, ncol = 1)
```








```
u_median = INLAOral$summary.random$Region.struct$`0.5quant`
germany.plot(exp(u_median), col = col, legend = TRUE)
```



b)

We will now add some explanatory factors for getting oral cancer.

```
smoking = read.table("additionalFiles/smoking.dat")

OralData["smoking"] = smoking
```

```
INLASmoking1 = inla(formula = Y ~ f(Region.struct, model = "besag", graph = g,
    hyper = list(prec = list(prior = "loggamma", param = c(alpha_u, beta_u))),
    constr = FALSE) + f(Region, model = "iid", hyper = list(prec = list(param = c(alpha_v,
    beta_v)))) + smoking - 1, family = "poisson", E = E, data = OralData,
    control.compute = list(dic = TRUE))
```

```
# Constr = FALSE?
INLASmoking2 = inla(formula = Y ~ -1 + f(Region.struct, model = "besag",
    graph = g, hyper = list(prec = list(param = c(alpha_u, beta_u))),
    constr = FALSE) + f(Region, model = "iid", hyper = list(prec = list(prior = "loggamma",
    param = c(alpha_v, beta_v)))) + f(smoking, model = "rw2", hyper = list(prec = list(prior = "loggamma
    constr = FALSE), family = "poisson", E = E, data = OralData, control.compute = list(dic = TRUE))
```

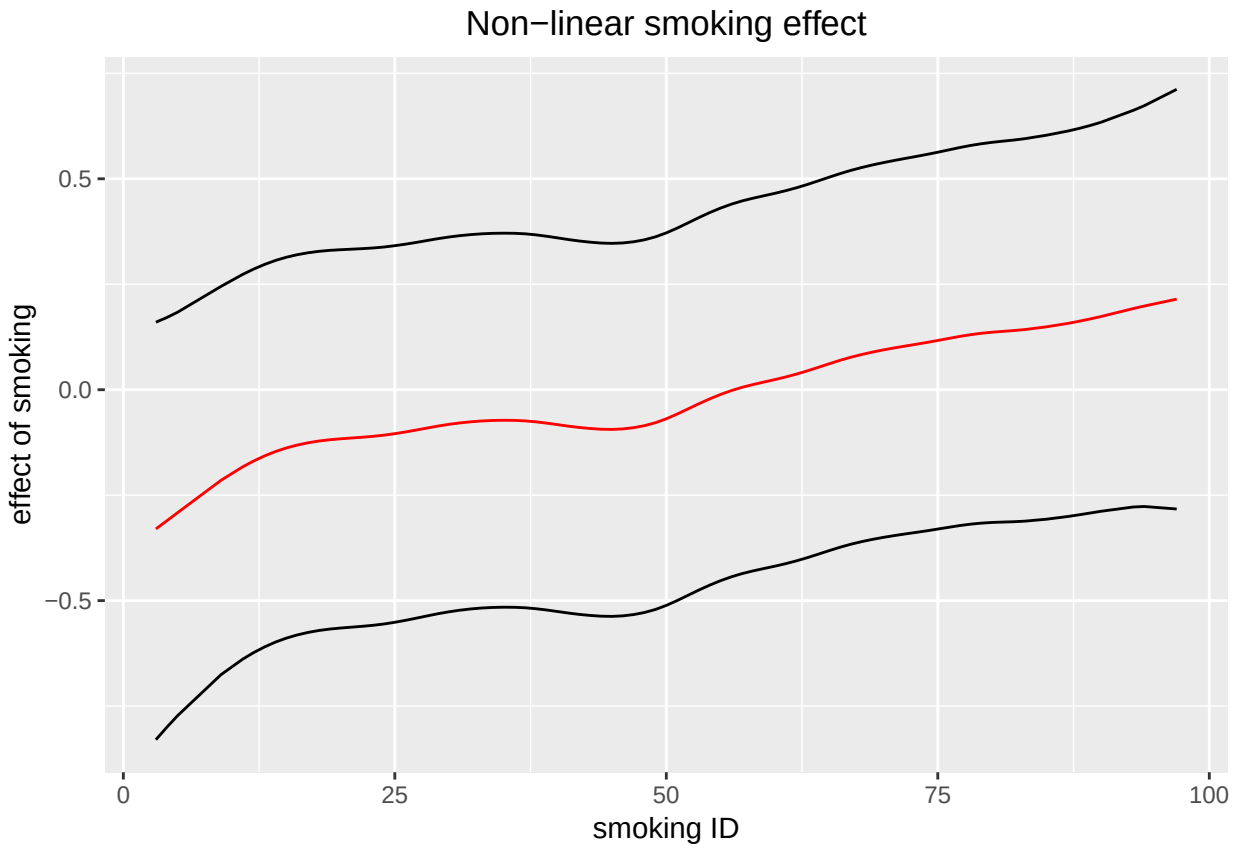
```
# Finding the DICs
DIC_spatial = INLAOral$dic$dic
DIC_smoking_linear = INLASmoking1$dic$dic
DIC_smoking_rw2 = INLASmoking2$dic$dic
```

```
## DIC for the pure spatial model: 3286.903
```

```
## DIC for the linear smoking effect model: 3276.857
```

```
## DIC for the rw2 smoking model: 3277.117
```

```
summary_smoking = INLASmoking2$summary.random$smoking
median_smoking = summary_smoking$`0.5quant`
lower_quant_smoking = summary_smoking$`0.025quant`
upper_quant_smoking = summary_smoking$`0.975quant`
ID_smoking = summary_smoking$ID

smoking_df = data.frame(index = ID_smoking, median = median_smoking,
    lower = lower_quant_smoking, upper = upper_quant_smoking)

ggplot(data = smoking_df, aes(x = index)) + geom_path(aes(y = median),
    col = "red") + geom_path(aes(y = lower)) + geom_path(aes(y = upper)) +
    xlab("smoking ID") + ylab("effect of smoking") + ggtitle("Non-linear smoking effect") +
    theme(plot.title = element_text(hjust = 0.5))
```


Non−linear smoking effect
effect of smoking
0.5
0.0
−0.5
0
25
50
75
100
smoking ID