NTNU

TMA 4300

Computer Intensive Statistical Methods

Exercise 2

Gunnar Grung Grotmol
Jan Henrik Jahren
Audun Aass Engstrøm

March 11, 2020
Trondheim

Fakulitet for Informasjonsteknologi og elektronikk

Institutt for matematiske fag

Project 2

Problem A

In this problem we look at the coal mining data, and do an analysis using Marcov chain Monte Carlo. The plot of acumulated disasters from the year 1851 to 1962 is given below.

```
library(coda)
library(MASS)
library(boot)
library(MCMCpack)
```

```
## ##
## ## Markov Chain Monte Carlo Package (MCMCpack)
```

```
## ## Copyright (C) 2003-2020 Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park
```

```
## ##
## ## Support provided by the U.S. National Science Foundation
```

```
## ## (Grants SES-0350646 and SES-0350613)
## ##
```

```
library(ggplot2)
library(dae)
library("INLA")
```

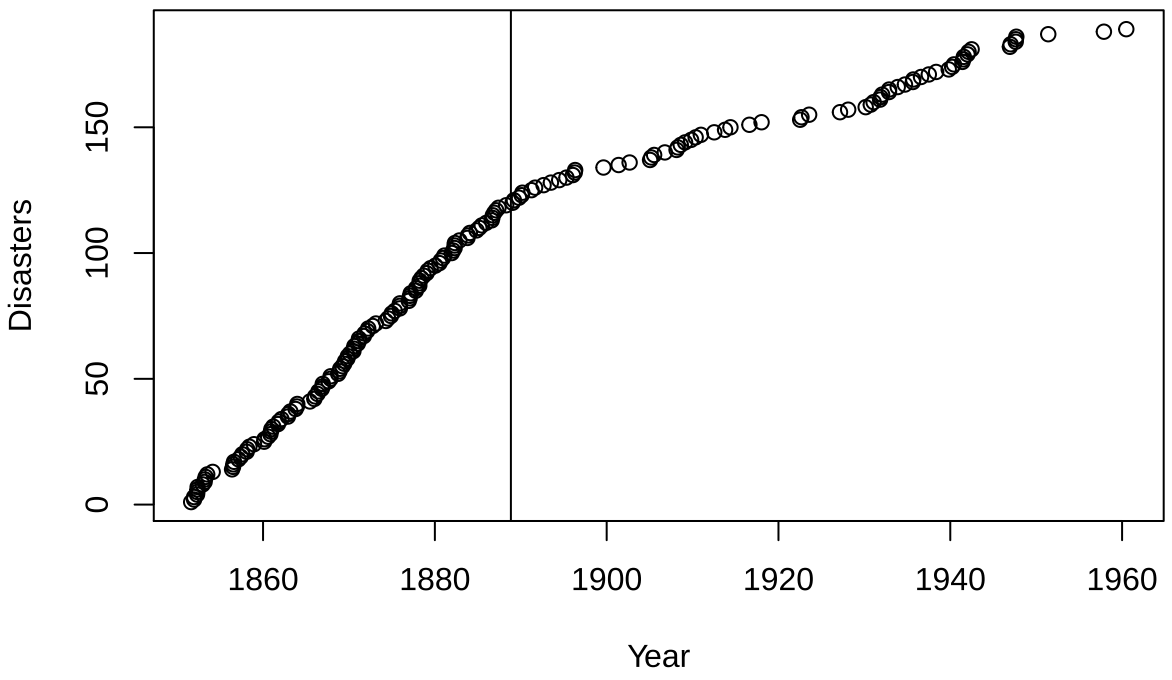
```
## Loading required package: Matrix
```

```
## Loading required package: sp
```

```
## Loading required package: parallel
```

```
## This is INLA_19.09.03 built 2019-09-03 09:07:31 UTC.
## See www.r-inla.org/contact-us for how to get help.
## To enable PARDISO sparse library; see inla.pardiso()
```

```
par(mfrow=c(1,1))
x.all <- coal[[1]]
x <- x.all[2:(length(x.all)-1)]
# Plot coalmine-data
plot(x,1:length(x),ylab = "Disasters",xlab = "Year");  abline(v = mean(x))
```



As seen from the above plot, the number of incidents seem to be greater in the 19'th century compared to the 20'th. The slope to the left of the mean(horisontal line) is greater than to the right. The coal mining disasters are assumed to follow an inhomogenius Poisson process with intensity function $\lambda(t)$, which is assumed to be piece-wize constant with n segments. We let t_0 and t_{n+1} denote the start en end time of the data set. It is possible to show that the likelihood function for the observed data is then

$$f(x|t_1,\ldots,t_n,\lambda_1,\ldots,\lambda_n) = \exp\left(-\int_{t_0}^{t_n+1} \lambda(t)dt\right)\prod_{k=0}^{n}\lambda_k^{y_k} = \exp\left(-\sum_{k=0}^{n}\lambda_k(t_{k+1}-t_k)\right)\prod_{k=0}^{n}\lambda_k^{y_k}$$

Where y_k is the number of incidents during the period t_k to t_{k+1}. We assume that all t_i, for $i \in \{1,\ldots,n\}$, are independent and uniformly distributed between t_0 and t_{n+1}. Apriori, we assume all λ_i to be independent of t_i and λ_j for all $i,j \in \{1,\ldots,n\}$ and $j \neq i$. Furthermore, we assume all λ_i to be i.i.d. from the same gamma distribution with shape parameter $\alpha = 2$ and scale parameter β. In other words

$$f(\lambda_i|\beta) = \frac{1}{\beta^2}\lambda_i e^{-\frac{\lambda_i}{\beta}}.$$

Moreover, the scale hyperparameter is assigned the improper prior

$$f(\beta) \propto \frac{e^{-1/\beta}}{\beta} \quad \text{for } \beta > 0.$$

We now restrict ourselves to look at n = 1, and therefore the model parameterts $\theta = (t_1,\lambda_0,\lambda_1,\beta)$.The posterior distribution can be found (up to a normalizing constant)

$$\begin{aligned} f(\theta|x) &\propto f(x|\theta)f(\theta) = f(x|\theta)f(t_1)f(\lambda_0,\lambda_1|\beta)f(\beta) \\ &= f(x|\theta)f(t_1)f(\lambda_0|\beta)f(\lambda_1|\beta)f(\beta) \\ &\propto \frac{\lambda_0^{y_0+1}\lambda_1^{y_1+1}}{\beta^5}e^{-\lambda_0(t_1-t_0)-\lambda_1(t_2-t_1)-\frac{1+\lambda_0+\lambda_1}{\beta}}. \end{aligned}$$

Furthermore the full conditionals can be derived. The F.C. for t_1 is

$$\begin{aligned} &f(t_1|x,\lambda_0,\lambda_1,\beta) \propto f(t_1)f(x|\lambda_0,\lambda_1,t_1,\beta) \\ &\propto \lambda_0^{y_0}\lambda_1^{y_1}e^{-\lambda_0(t_1-t_0)-\lambda_1(t_2-t_1)} \propto \lambda_0^{y_0}\lambda_1^{y_1}e^{t_1(\lambda_1-\lambda_0)} \end{aligned}$$

which is not a known distribution. Next, the full conditional λ_0 is given by

$$\begin{aligned} f(\lambda_0 \,|\, \lambda_1, t_1, \beta, x) &\propto f(\lambda_0 \,|\beta) f(\lambda_1, t_1, x|\lambda_0, \beta) \\ &\propto f(\lambda_0 \,|\beta) f(x|\lambda_1, \lambda_0, t_1, \beta) \\ &\propto \lambda_0^{y_0+1} e^{-\lambda_0(t_1-t_0)-\lambda_0/\beta} \end{aligned}$$

which is the gamma distribution, $\lambda_0 \,|\, \lambda_1, t_1, \beta, x \sim \text{Gamma}\,(y_0+2, t_1-t_0+1/\beta)$ using shape and rate parameters. Using the exact same resoning, the F.C. for λ_1 is

$$f(\lambda_1 \,|\, \lambda_0, t_1, \beta, x) \propto \lambda_1^{y_1+1} e^{-\lambda_1(t_2-t_1)-\lambda_1/\beta}.$$

Therefore, $\lambda_1 \,|\, \lambda_0, t_1, \beta, x \sim \text{Gamma}\,(y_1+2, t_2-t_1+1/\beta)$ using shape and rate parameters. Lastly, the full conditional of β is

$$\begin{aligned} f(\beta \,|\, \lambda_0, \lambda_1, t_1, x) \propto f(\beta) f(\lambda_0, \lambda_1, t_1, x|\beta) &\propto f(\beta) f(\lambda_0|\beta) f(\lambda_1|\beta) \\ &\propto \frac{1}{\beta^5} e^{-\beta(1+\lambda_0+\lambda_1)}, \end{aligned}$$

which is the inverse gamma distribution. That is, $\beta \,|\, \lambda_0, \lambda_1, t_1, x \sim \text{InvGamma}(4, (1+\lambda_0+\lambda_1))$ with shape and scale parameter respectivly.

Next, we want to sample from the posterior distrubution using a single site MCMC. The variable of interest when using Metropolis-Hastings is t_1. The other variables will be drawn from their known full conditionals. The new values are found in the order: t_1, β, λ_0 and lastly λ_1. We want the limiting distribution of the Metropolis-Hastings step to be

$$f(\theta|x) \propto f(t_1|x, \lambda_0, \lambda_1, \beta) \propto \lambda_0^{y_0} \lambda_1^{y_1} e^{-\lambda_0(t_1-t_0)-\lambda_1(t_2-t_1)}.$$

keeping the other variables constant. Therefore, we intoduce the notation $\pi(t) = f(t|x, \lambda_0, \lambda_1, \beta)$. Let the probability for transitioning from state t_1 to t_1' to be $p(t_1'|t_1)$. In order to assure a unique limiting distribution we need the Markov chain to be aperiodic and reversible. We require

$$\pi(t_1) p(t_1'|t_1) = \pi(t_1') p(t_1, t_1'),$$

otherwize, the Markov chain would not be reversible, as for example a transition from t_1 to t_1' would occur to often. Since this might not be the case, we should only accept a transition from t_1 to t_1' with a probability $\alpha(t_1, t_1')$. We need to construnct α such that

$$\pi(t_1) q(t_1'|t_1) \alpha(t_1, t_1') = \pi(t_1') q(t_1|t_1') \alpha(t_1', t_1),$$

where $p(t_1'|t_1) = q(t_1'|t_1)\alpha(t_1, t_1')$. Here, $q(t_1'|t)$ is the distribution used to propose a new candidate t_1'. A natural choice for α is to pick

$$\alpha(t_1, t_1') = \min\{1\,, \frac{\pi(t_1') q(t_1|t_1')}{\pi(t_1) q(t_1'|t_1)}\}.$$

In our implementation, we choose the random walk Markov chain, with the symmetric transition probability $q(t_1'|t_1) \sim \mathcal{N}(t_1, \sigma^2)$ for some properly chosen tuning parameter σ. Since the transition probability is symmetric, $q(t_1'|t_1) = q(t_1|t_1')$, the acceptence probability becomes

$$\alpha(t_1, t_1') = \min\left\{1\,, \frac{\pi(t_1')}{\pi(t_1)}\right\} = \min\left\{1\,, \frac{\lambda_0^{y_0'} \lambda_1^{y_1'} e^{-\lambda_0\, t_1' + \lambda_1\, t_1'}}{\lambda_0^{y_0} \lambda_1^{y_1} e^{-\lambda_0\, t_1 + \lambda_1\, t_1}}\right\}.$$

In order to implement the algorithm, we introduce the variables and help functions

```r
#global variables
t0 <- x.all[1]
t2 <- x.all[length(x.all)]
x <- x.all[2:(length(x.all)-1)]
#help functions
get.y <- function(x, t) {
  # Get y_0 and y_1, the number of events occuring before and after t
  y_0 <- sum(x <= t)
  y_1 <- length(x) - y_0
  return(c(y_0,y_1))
}
sample.FC.beta <- function(lambda_0,lambda_1) {
  # Sample beta from inverse gamma distribution
  return(rinvgamma(1,shape = 4, scale = (lambda_0 + lambda_1 + 1)))
}
sample.FC.lambda_0 <- function(t_1, beta, x) {
  # Sample lambda_0 from gamma distribution
  return(rgamma(1,shape = get.y(x,t_1)[1] + 2,scale = beta/(beta*(t_1-t0)+1)))
}

sample.FC.lambda_1 <- function(t_1, beta,x) {
  # Sample lambda_1 from gamma distribution
  lambda_1 = rgamma(1,shape = get.y(x,t_1)[2] + 2, scale = beta/(beta*(t2-t_1)+1))
}
```

Since the values of $\lambda_0^{y_0}\lambda_1^{y_1}$ can be very large the calculations are done in logarithmic scale in order to increase stability. The code for the single site Markov chain Monte Carlo is given below.

```r
single_site_MH_MCMC <- function(N,t_1,lambda_0, lambda_1, beta, x, sigma) {
  #create data frame and insert first values
  MCMC.data <- data.frame(matrix(0,N,4))
  colnames(MCMC.data) <- c('t_1','lambda_0','lambda_1','beta')
  MCMC.data <- data.frame(t_1 = double(), lambda_0 = double(), lambda_1 = double(), beta = double())
  MCMC.data[1,] <- c(t_1,lambda_0,lambda_1,beta)

  successes <- 0 # count number of sucessfull new draws
  for (i in 2:N){
    t.new <- rnorm(1,mean = t_1, sd = sigma)#propose new t_1 value
    if( (t.new < t0) | (t.new > t2)){
      # if outside allowed region, trow away new sample
      log_alpha <- -Inf
    }
    else{
      # Find y-values at t_1 and t.new
      y <- get.y(x, t_1)
      y.new <- get.y(x, t.new)

      #calculate numerator and denomenator of acceptance ratio log scale

      numerator <- y.new[1]*log(lambda_0) + y.new[2]*log(lambda_1) + t.new*(lambda_1 - lambda_0)

      denomerator <- y[1]*log(lambda_0) + y[2]*log(lambda_1) + t_1*(lambda_1 - lambda_0)
      log_alpha <- numerator - denomerator    # log Acceptance probability
    }
    # if u is U(0,1), u < alpha, accept new draw,
    if (runif(1) < exp(log_alpha)) {
      t_1 <- t.new
      successes <- successes + 1 # count number of successes
    }
    # if u > alpha, do nothing
    # update data frame
    MCMC.data[i,] <- c(t_1,lambda_0,lambda_1,beta)
    #draw new lambda_0, lambda_1, beta
    beta <- sample.FC.beta(lambda_0,lambda_1)
    lambda_0 <- sample.FC.lambda_0(t_1,beta,x)
    lambda_1 <- sample.FC.lambda_1(t_1,beta,x)
    # do everything again with new parameters
  }
  return(c(MCMC.data,successes/N))
}
```

We next try the algorithm to check burn in and mixing properties. We set the initial value of t_1 to the mean of t_0 and t_2. For convenience we set the initial value of the hyperparameter $\beta = 1$ and sample λ_0 and λ_1 from the their full conditionals above as starting values.

```
# initial values of single site algorithm
t.init = 0.5*(t0+t2)
beta = 1.0
lambda_0 = sample.FC.lambda_0(t.init, beta, x)
lambda_1 = sample.FC.lambda_1(t.init, beta, x)
lambda_0;lambda_1;
```

```
## [1] 2.795728
```

```
## [1] 0.8669295
```

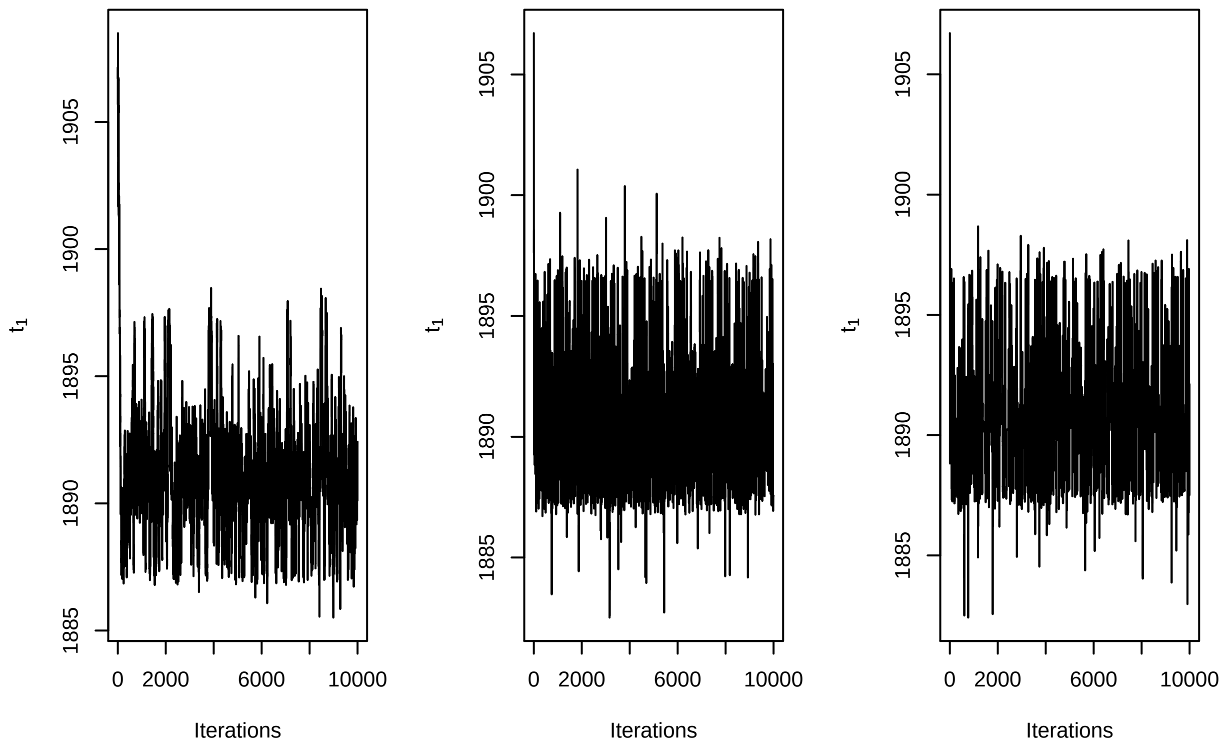
We run for 5000 iterations with different values of σ. Three values are compared. These are $\sigma = 1$, $\sigma = 5$ and $\sigma = 10$.

```
# steps of Markov chain
N <- 10000
# with sigma = 1
sigma = 1
results <- single_site_MH_MCMC(N,t.init,lambda_0,lambda_1,beta,x,sigma)
MCMC.data1 = results[1:4]; success_rate1 = results[5]

# with sigma = 5
sigma = 5
results <- single_site_MH_MCMC(N,t.init,lambda_0,lambda_1,beta,x,sigma)
MCMC.data2 = results[1:4]; success_rate2 = results[5]

# with sigma = 10
sigma = 10
results <- single_site_MH_MCMC(N,t.init,lambda_0,lambda_1,beta,x,sigma)
```

```r
MCMC.data3 = results[1:4]; success_rate3 = results[5]

success_rate1; success_rate2; success_rate3;
```

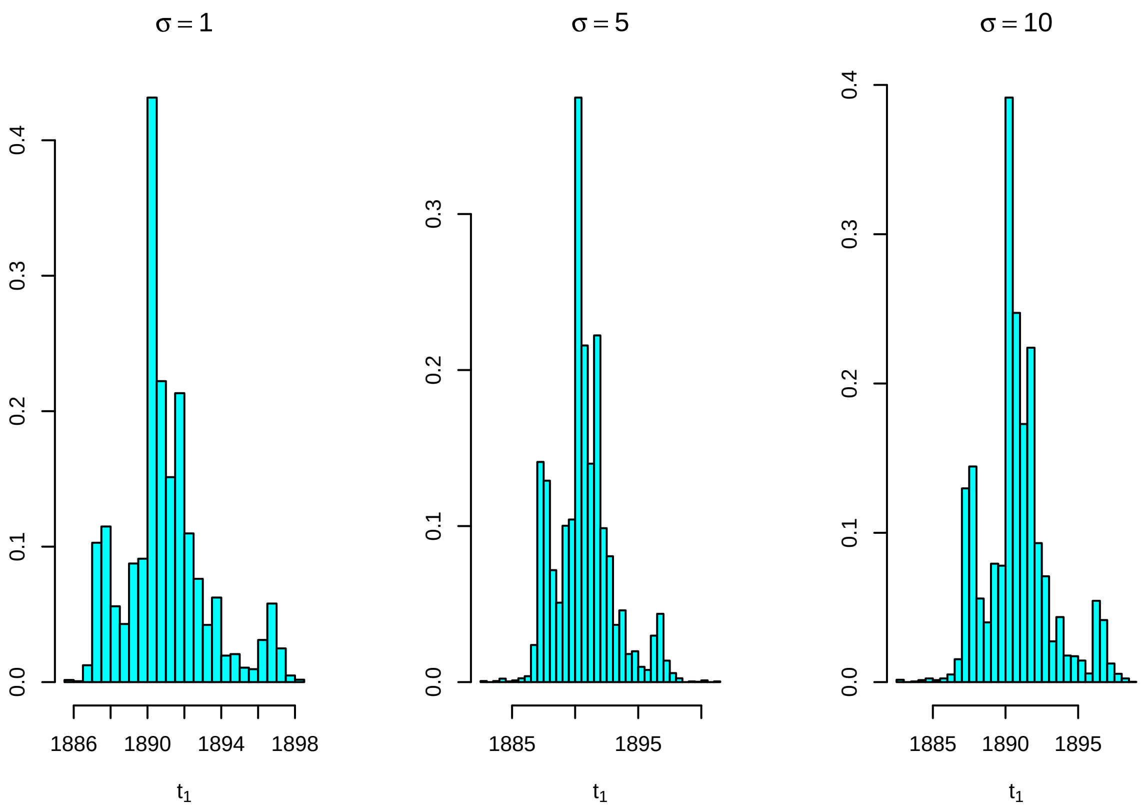
```
## [[1]]
## [1] 0.6064

## [[1]]
## [1] 0.3218

## [[1]]
## [1] 0.1783
```

```r
# Plot results
par(mfrow=c(1,3))
plot(MCMC.data1$t_1, type='l', ylab=expression(t[1]), xlab = "Iterations")
plot(MCMC.data2$t_1, type='l', ylab=expression(t[1]), xlab = "Iterations")
plot(MCMC.data3$t_1, type='l', ylab=expression(t[1]), xlab = "Iterations")
```



As seen from the above plot, the algorithm seems to use somewhere close to 500 iterations to reach the points around the equilibrium state for $\sigma = 1$. For both $\sigma = 5$ and $\sigma = 10$, the sampled values converge to the equilibrium faster. The acceptence rates for the different values of σ are quite different, with the acceptence ratio for $\sigma = 1$ around 0.6, $\sigma = 5$ around 0.3 and $\sigma = 10$ around 0.2. If the value of sigma is set too high, many proposed values will be too extreme, yielding a lower convergence rate. The higher values of sigma in other words introduce smaller acceptence values. However, we see that all Makov chains ocilate around the same line around 1890. In other words, the algorithm does converge to the same distribution for all σ values, but at different rates. This is seen in the plot below, with a burn in of 1000.

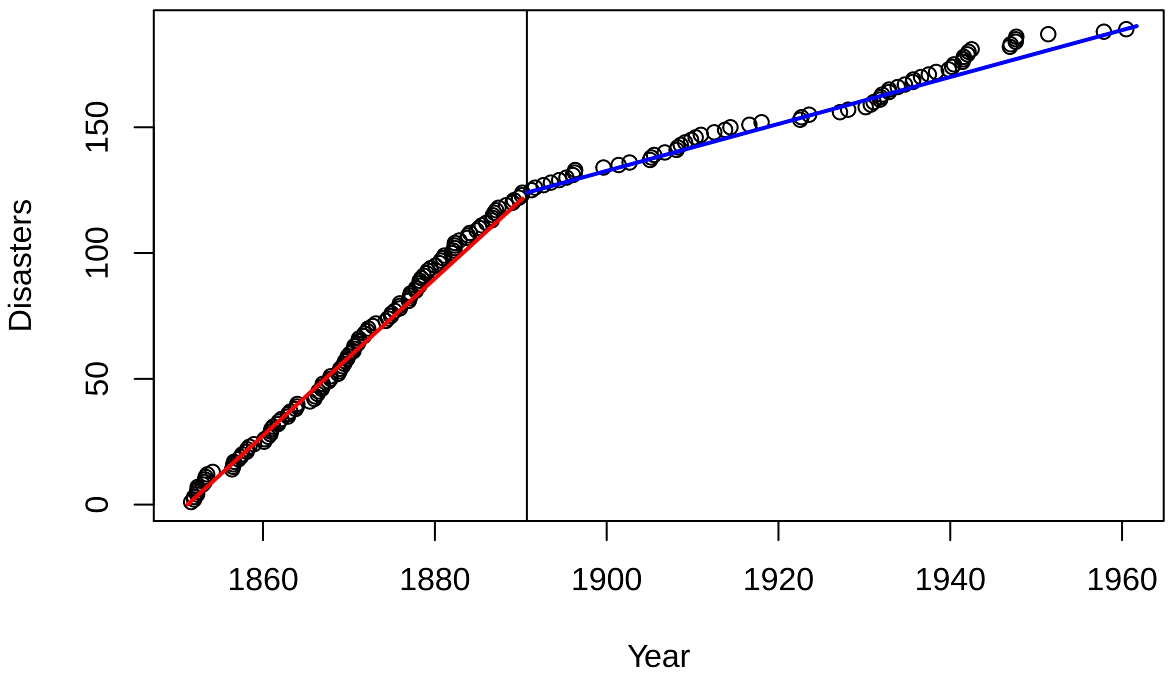
```
Burn.in = 1000
par(mfrow=c(1,3))
truehist(MCMC.data1$t_1[Burn.in:N], xlab = expression(t[1]),main = expression(sigma == 1))
truehist(MCMC.data2$t_1[Burn.in:N], xlab = expression(t[1]),main = expression(sigma == 5))
truehist(MCMC.data3$t_1[Burn.in:N], xlab = expression(t[1]),main = expression(sigma ==10))
```



The ablove plot shows that convergence is achieved using any σ. Furthermore, the limiting distribution is unique. Since $\sigma = 1$ uses around 500 iterations in order to reach a equalibrium, the burn in will be set to be a reasonable 1000. Using this value, we run the algorithm further using $\sigma = 5$, and compare the rusults to the Coal mining disaster data.

```
#Parameters
N <- 50000
sigma <- 5
Burn.in <- 1000
MCMC.data <- single_site_MH_MCMC(N,t.init,lambda_0,lambda_1,beta,x,sigma)[0:4]
```

```
# see if algorithm makes sense
t_avg = mean(MCMC.data$t_1[Burn.in:N])
lambda_0_avg = mean(MCMC.data$lambda_0[Burn.in:N])
lambda_1_avg = mean(MCMC.data$lambda_1[Burn.in:N])
#plot
par(mfrow=c(1,1))
plot(x,1:length(x),xlab = "Year", ylab = "Disasters")
abline(v = mean(MCMC.data$t_1), ylab ="Disasters")
lines(seq(t0, t_avg), lambda_0_avg*(seq(t0, t_avg)-t0), type = "l", col = "red",  lwd=2)
lines(seq(t_avg, t2), lambda_1_avg*(seq(t_avg, t2)-t_avg)+get.y(x,t_avg)[1],type = "l", col = "blue", l
```



```r
# Average values of the rate parameters
sprintf("mean of t_1: %.3f,", t_avg)
```

```
## [1] "mean of t_1: 1890.703,"
```

```r
sprintf("mean of lambda_0: %.3f,", lambda_0_avg)
```

```
## [1] "mean of lambda_0: 3.120,"
```

```r
sprintf("mean of lambda_1: %.3f ", lambda_1_avg)
```

```
## [1] "mean of lambda_1: 0.933 "
```

The above plot shows a vertical line with x-value equal to the mean of sampled t_1. Furthermore, it shows a line with slope equal to the sample mean of λ_0 to the left, and a line with slope mean of λ_1 to the right. These lines closely hug the cumulative distribution of the disaster data. Therefore, the sampled values make sense. Our algorithm is most likely implemented correctly.

We next define a two block Metropolis-Hastings algorithm when sampling from $f(\theta|x)$. The first block proposal is updating the block $(t_1, \lambda_0, \lambda_1)$ keeping β constant. In this block we generate a proposal to a new value for t_1, $\widetilde{t}_1$ using a normal distribution $\widetilde{t}_1 \sim \mathcal{N}(t_1, \sigma_{t_1}^2)$, with mean t_1 and variance $\sigma_{t_1}^2$. Thereafter, we propose the new values of (λ_0, λ_1), $(\widetilde{\lambda}_0, \widetilde{\lambda}_1)$, using their joint full conditional $f(\lambda_0, \lambda_1|x, \widetilde{t}_1, \beta)$ inserted the value $\widetilde{t}_1$. The second block is $(\lambda_0, \lambda_1, \beta)$ keeping t_1 unchanged. A new proposal to β, $\widetilde{\beta}$, is sampled from the normal distribution $\widetilde{\beta} \sim \mathcal{N}(\beta, \sigma_\beta^2)$, with mean β. The proposed values $(\widetilde{\lambda}_0, \widetilde{\lambda}_1)$ are thereafter found using their full conditional $f(\lambda_0, \lambda_1|x, t_1, \widetilde{\beta})$ inserted $\widetilde{\beta}$. Let α_1 and α_2 be the acceptence probability of the first and second block respectively. Focusing on the first block, we again require

$$\alpha_1(t_1, \lambda_0, \lambda_1, \widetilde{t}_1, \widetilde{\lambda}_0, \widetilde{\lambda}_1) = \min\left\{1, \ \frac{f(\widetilde{\theta}|x) q_1(t_1, \lambda_0, \lambda_1|\widetilde{t}_1, \widetilde{\lambda}_0, \widetilde{\lambda}_1)}{f(\theta|x)\, q_1(\widetilde{t}_1, \widetilde{\lambda}_0, \widetilde{\lambda}_1|t_1, \lambda_0, \lambda_1)}\right\}$$

Where

$$q_1(\widetilde{t}_1,\widetilde{\lambda}_0,\widetilde{\lambda}_1|t_1,\lambda_0,\lambda_1,x,\beta)=f(\widetilde{t}_1|t_1,\lambda_0,\lambda_1,x,\beta)f(\widetilde{\lambda}_0,\widetilde{\lambda}_1|\widetilde{t}_1,t_1,\lambda_0,\lambda_1,x,\beta)=f(\widetilde{t}_1|t_1)f(\lambda_0|x,\widetilde{t}_1,\beta)f(\lambda_1|x,\widetilde{t}_1,\beta).$$

Since the value of $\widetilde{t}_1$ is sampled from the symmetric normal distribution with mean t_1, we have $f(\widetilde{t}_1|t_1)=f(t_1|\widetilde{t}_1)$. Therefore, this part of the expression cancels out. Furthermore, the exponential part and the polynomial part of the gamma distribution cancels with the posterior distribution. Therefore the acceptence probability for the block 1 transition becomes

$$\alpha_1(t_1,\lambda_0,\lambda_1,\widetilde{t}_1,\widetilde{\lambda}_0,\widetilde{\lambda}_1)=\min\left\{1,\ \frac{\Gamma(\widetilde{y}_0+2)\Gamma(\widetilde{y}_1+2)}{\Gamma(y_0+2)\Gamma(y_1+2)}\frac{(t_1-t_0+1/\beta)^{y_0+2}(t_2-t_1+1/\beta)^{y_1+2}}{(\widetilde{t}_1-t_0+1/\beta)^{\widetilde{y}_0+2}(t_2-\widetilde{t}_1+1/\beta)^{\widetilde{y}_1+2}}\right\}.$$

we now look at the transition probability of block 2. The transition kernel, q_2, is on the form

$$q_2(\widetilde{\lambda}_0,\widetilde{\lambda}_0,\widetilde{\beta}|\lambda_0,\lambda_1,\beta)=f(\widetilde{\beta}|\lambda_0,\lambda_1,\beta)f(\widetilde{\lambda}_0,\widetilde{\lambda}_1|\widetilde{\beta},\lambda_0,\lambda_1,\beta)=f(\widetilde{\beta}|\beta)f(\widetilde{\lambda}_0,\widetilde{\lambda}_1|x,t_1,\widetilde{\beta})$$

yielding

$$\alpha_2(\widetilde{\lambda}_0,\widetilde{\lambda}_1,\widetilde{\beta},\lambda_0,\lambda_1,\beta)=\min\left\{1,\ \frac{\beta^5}{\widetilde{\beta}^5}e^{1/\beta-1/\widetilde{\beta}}\frac{(t_1-t_0+1/\beta)^{y_0+2}(t_2-t_1+1/\beta)^{y_1+2}}{(t_1-t_0+1/\widetilde{\beta})^{y_0+2}(t_2-t_1+1/\widetilde{\beta})^{y_1+2}}\right\}.$$

The implementation can be seen below. When implementing we set $\sigma_{t_1}=2\sigma_\beta=\sigma$.

```
Block_MH_MCMC <- function(N,t_1, lambda_0, lambda_1, beta, x, sigma){
  #create data frame and insert first values
  MCMC.data <- data.frame(matrix(0,N,4))
  colnames(MCMC.data) <- c('t_1','lambda_0','lambda_1','beta')
  MCMC.data <- data.frame(t_1 = double(), lambda_0 = double(), lambda_1 = double(), beta = double())
  MCMC.data[1,] <- c(t_1,lambda_0,lambda_1,beta)

  for (i in 2:N){
    #block 1
    t.new <- rnorm(1,mean=t_1,sd = sigma)
    #draw new sample if outside allowed region
    while(t.new <t0 | t.new > t2){t.new <- rnorm(1,mean=t_1,sd = sigma)}
    lambda_0.new <- sample.FC.lambda_0(t.new,beta, x)
    lambda_1.new <- sample.FC.lambda_1(t.new,beta, x)
    y <- get.y(x,t_1)
    y.new <- get.y(x,t.new)


    numerator   <- lfactorial(y.new[1]+2) + lfactorial(y.new[2]+2) +(y[1]+2)*log(t_1-t0+1/beta) +

    denomerator <- lfactorial(y[1]+2) + lfactorial(y[2]+2) +(y.new[1]+2)*log(t.new - t0+1/beta) + (y.ne
    alpha_log <- numerator - denomerator
    if (runif(1)< exp(alpha_log)){
      t_1 <- t.new
      lambda_0 <- lambda_0.new
      lambda_1 <- lambda_1.new
    }

    #block 2
    beta.new <- rnorm(1,mean= beta,sd = 0.5*sigma)
```

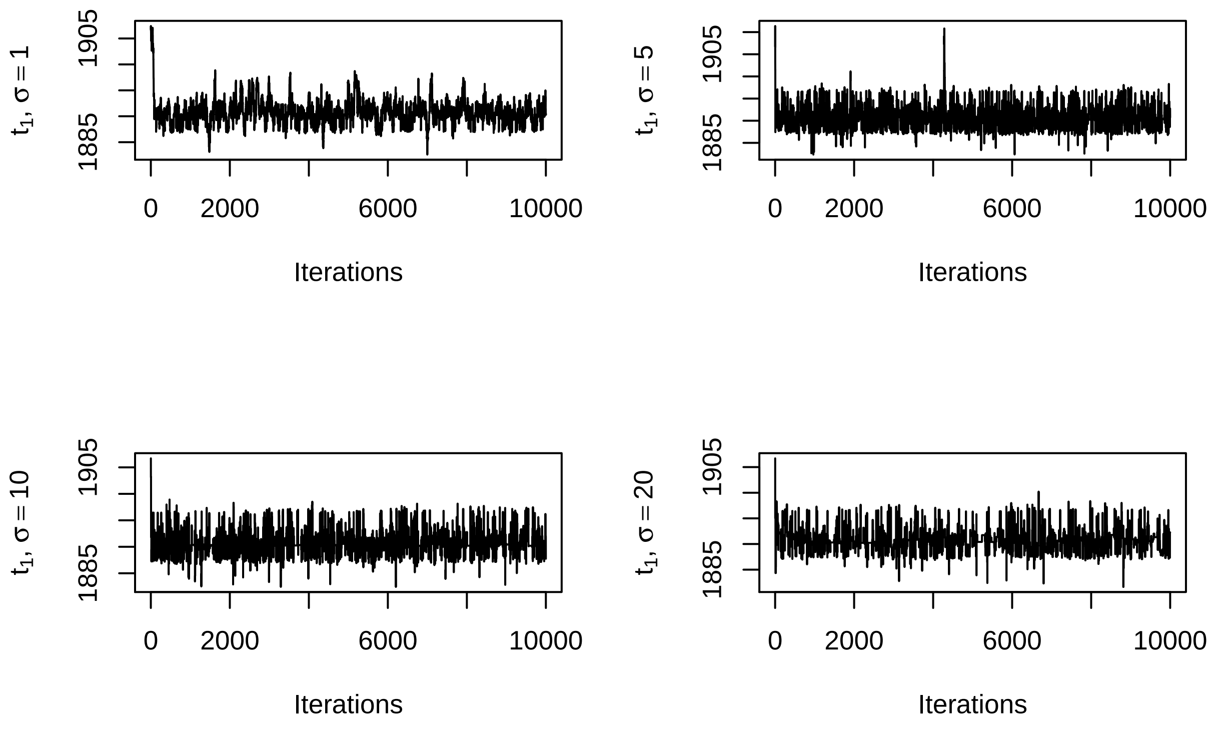
```r
    # draw new sample if beta non-positive
    while(beta.new <= 0){beta.new <- rnorm(1,mean= beta,sd = 0.5*sigma)}
    lambda_0.new <- sample.FC.lambda_0(t_1,beta.new, x)
    lambda_1.new <- sample.FC.lambda_1(t_1,beta.new, x)
    y <- get.y(x,t_1)

    numerator   <- 5*log(beta)+(1/beta-1/beta.new)  +(y[1]+2)*log(t_1-t0+1/beta) + (y[2]+2)*log(t2-t_1+
    denomerator <- 5*log(beta.new) +(y[1]+2)*log(t_1-t0+1/beta.new)+ (y[2]+2)*log(t2-t_1+1/beta.new)

    alpha_log <- numerator - denomerator
    if (runif(1)< exp(alpha_log)){
      lambda_0 <- lambda_0.new
      lambda_1 <- lambda_1.new
      beta <- beta.new
    }
    MCMC.data[i,] <- c(t_1,lambda_0,lambda_1,beta)
  }
  return(MCMC.data)
}
```

We run the algorithm for different choices of σ to evaluate the mixing properties and burn in. The values picked are $\sigma \in \{1, 5, 10, 20\}$.

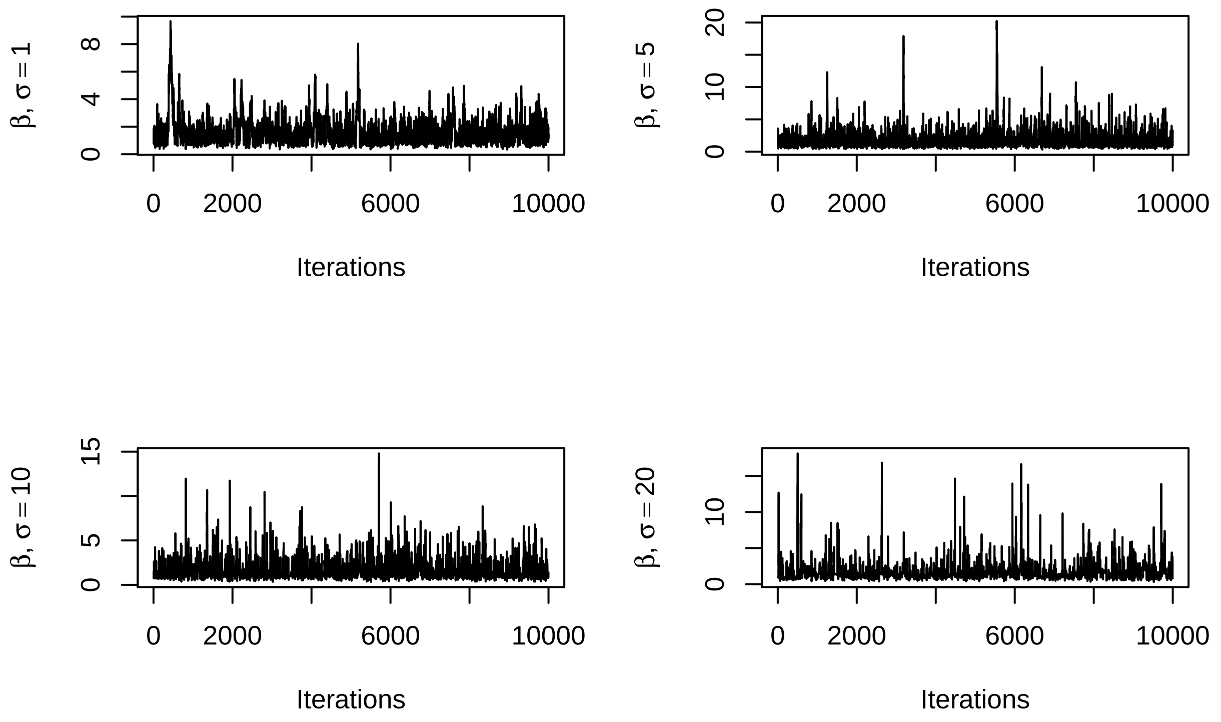
```r
# initial values of single site algorithm
t.init <- 0.5*(t0+t2)
beta <- 1.0
lambda_0 <- sample.FC.lambda_0(t.init, beta, x)
lambda_1 <- sample.FC.lambda_1(t.init, beta, x)
# steps of Markov chain
N <- 10000
# with sigma = 1
sigma = 1
MCMC.data1 <- Block_MH_MCMC(N,t.init,lambda_0,lambda_1,beta,x,sigma)
# with sigma = 5
sigma = 5
MCMC.data2 <- Block_MH_MCMC(N,t.init,lambda_0,lambda_1,beta,x,sigma)
# with sigma = 10
sigma = 10
MCMC.data3 <- Block_MH_MCMC(N,t.init,lambda_0,lambda_1,beta,x,sigma)
# with sigma = 20
sigma = 20
MCMC.data4 <- Block_MH_MCMC(N,t.init,lambda_0,lambda_1,beta,x,sigma)
```

Below are two plots showing the sampled values of t_1 and β.

```r
# plot t_1
par(mfrow=c(2,2))
plot(MCMC.data1$t_1, type='l', ylab=expression(paste(t[1],", ",sigma== 1)), xlab = "Iterations")
plot(MCMC.data2$t_1, type='l', ylab=expression(paste(t[1],", ",sigma== 5)), xlab = "Iterations")
plot(MCMC.data3$t_1, type='l', ylab=expression(paste(t[1],", ",sigma==10)), xlab = "Iterations")
plot(MCMC.data4$t_1, type='l', ylab=expression(paste(t[1],", ",sigma==20)), xlab = "Iterations")
```



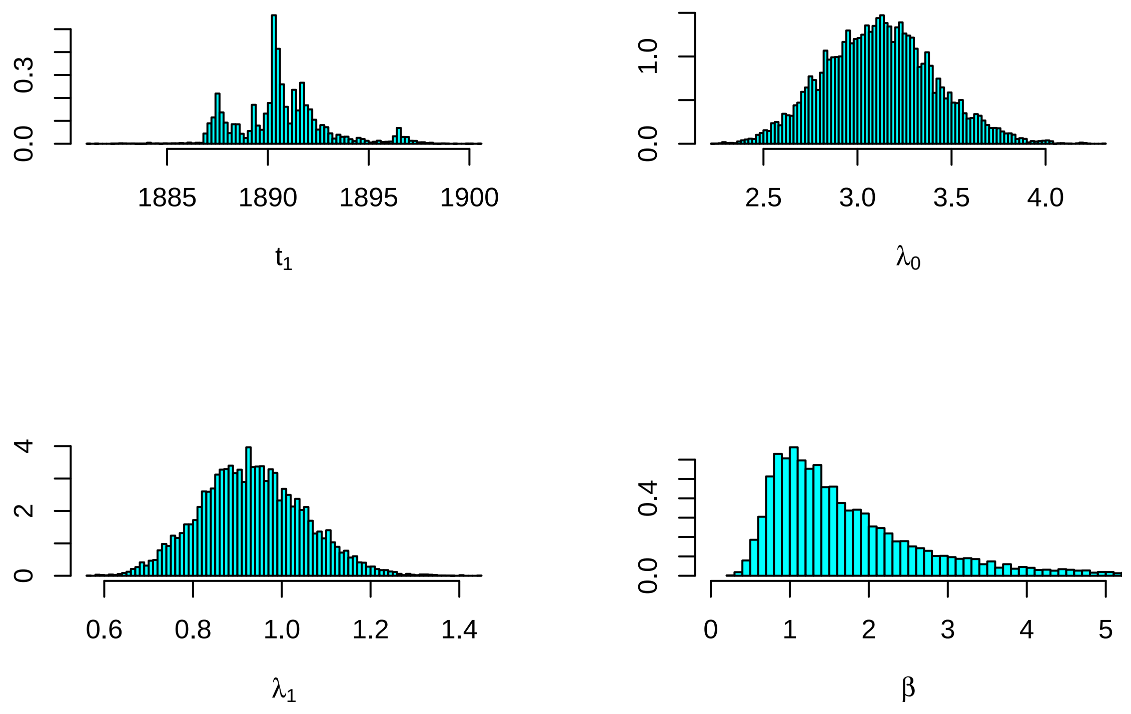
```
#plot beta
par(mfrow=c(2,2))
plot(MCMC.data1$beta, type='l', ylab=expression(paste(beta,", ",sigma== 1)), xlab = "Iterations")
plot(MCMC.data2$beta, type='l', ylab=expression(paste(beta,", ",sigma== 5)), xlab = "Iterations")
plot(MCMC.data3$beta, type='l', ylab=expression(paste(beta,", ",sigma==10)), xlab = "Iterations")
plot(MCMC.data4$beta, type='l', ylab=expression(paste(beta,", ",sigma==20)), xlab = "Iterations")
```



As seen from the above plots, the optimal mixing occors for $\sigma = 5$. For $\sigma = 1$ the acceptence value is high with small movements, leading to slow convergence. For high values of σ, $\sigma = 10$ and $\sigma = 20$ the mixing is slower than for $\sigma = 5$. This is especially clear for $\sigma = 20$, where the line is less noisy, especially for t_1. This implies a low acceptence rate. It is, however, evident that the mixing property of the two block Markov chain Monte Carlo is greater than it is for the single site MCMC. The number of iterations needed to reach equalibrium for the two block seems far less than 500, as was the value for the single site MCMC. We pick $\sigma = 5$ and run the algorithm further. We then apply the a burn in of 1000, and look at the results.

```
N <- 50000
Burn.in <- 1000
MCMC.data <- Block_MH_MCMC(N,t.init,lambda_0,lambda_1,beta,x,sigma)[Burn.in:N,]
```

```
par(mfrow=c(2,2))
truehist(MCMC.data$t_1, xlab = expression(t[1]))
truehist(MCMC.data$lambda_0, xlab = expression(lambda[0]))
truehist(MCMC.data$lambda_1, xlab = expression(lambda[1]))
truehist(MCMC.data$beta, xlim = c(0,5),, xlab = expression(beta))
```



```
sprintf("Mean of t_1: %.2f",mean(MCMC.data$t_1))
```

```
## [1] "Mean of t_1: 1890.66"
```

```
sprintf("Mean of lambda_0: %.3f",mean(MCMC.data$lambda_0))
```

```
## [1] "Mean of lambda_0: 3.120"
```

```
sprintf("Mean of lambda_1: %.3f",mean(MCMC.data$lambda_1))
```

```
## [1] "Mean of lambda_1: 0.934"
```

```
sprintf("Mean of beta: %.3f",mean(MCMC.data$beta))
```

```
## [1] "Mean of beta: 1.854"
```

```
sprintf("Covariance lambda_0, lambda_1 : %.5f",cov(MCMC.data$lambda_0,MCMC.data$lambda_1))
```

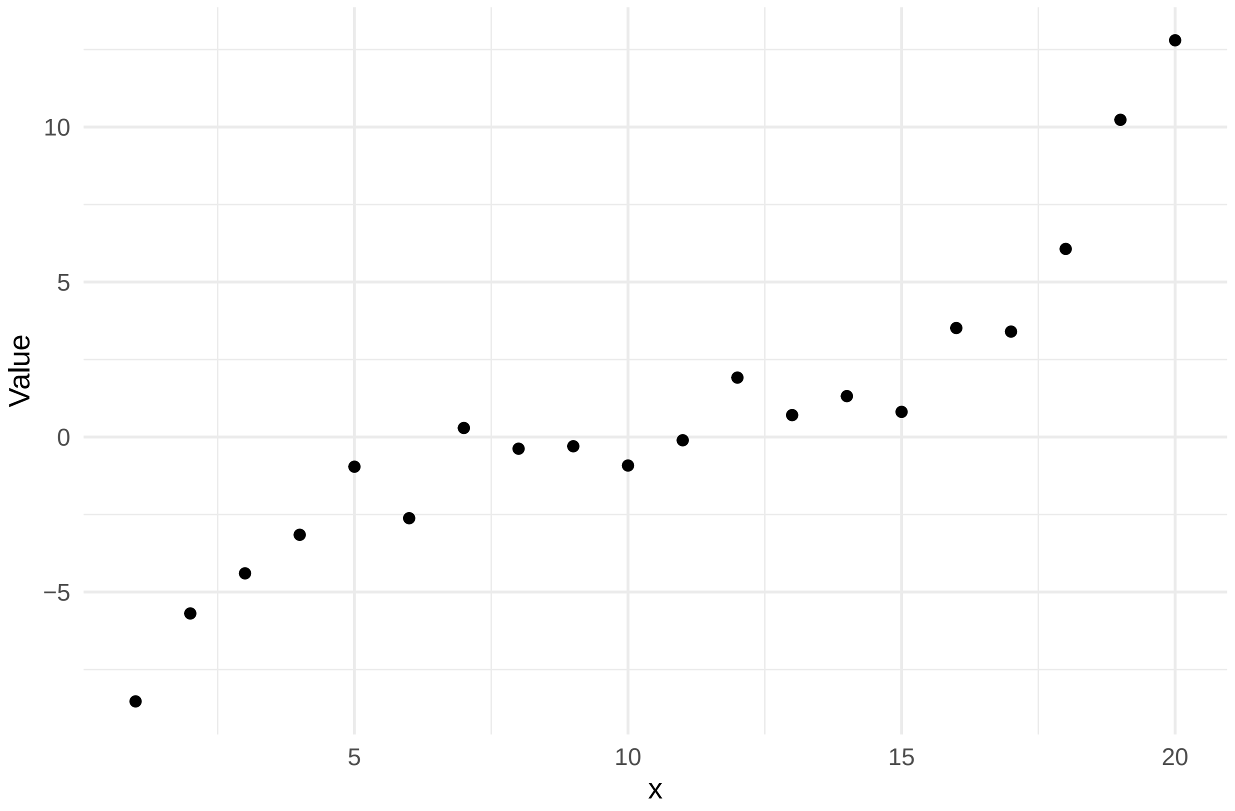
```
## [1] "Covariance lambda_0, lambda_1 : 0.00120"
```

Above are the estimated distributions of $f(t_1|x)$, $f(\lambda_0|x)$, $f(\lambda_1|x)$ and $f(\beta|x)$. In adition, the estimated values of $\mathrm{E}[t_1|x]$, $\mathrm{E}[\lambda_0|x]$, $\mathrm{E}[\lambda_1|x]$, $\mathrm{E}[\beta|x]$ and $\mathrm{Cov}[\lambda_0, \lambda_1|x]$

Problem B

1)

```
#Read in from the website and plot the data
filepath <- "https://www.math.ntnu.no/emner/TMA4300/2020v/Exercise/exercise2/Gaussiandata.txt"


gaus<- read.table(file = filepath, header = FALSE)
x <- seq(1, length(gaus$V1))
gaus <- cbind(x, gaus)
plot <- ggplot(data = gaus, aes(x = x, y = V1)) + geom_point() + theme_minimal() + labs(y = "Value")
plot + labs(caption = "Plot of the observed data")
```



Plot of the observed data

Our model can be written as a combination of the likelihood of the response and the prior distributuon of the latent field (Hierarchical Mode):

$$\mathbf{y} \mid \mathbf{f} \sim \prod_{t=1}^{T} P(y_t \mid \eta_t)$$

$$\mathbf{f} \mid \theta \sim \pi(\mathbf{f} \mid \theta) = N(\mathbf{0}, \mathbf{Q}(\theta)^{-1})$$

Where

$$\theta \sim Gamma(1,1)$$

Since we can write our model in this way its by definition a latent Gaussian model.

Since our model is a LGM, each of our observations y_i only depends on one of the of the elements (linear predictors) in the latent Gaussian field and the dimension of hyperparameters is only 1. (Must be <=9) . It is also desireble to use INLA in this case cause the cause the precision matrix $\mathbf{Q}(\theta)$ is sparse.

2)

We want to implement a block Gibbs sampling using two proposed value for θ and the linear predictor vector η from

$$\pi(\theta \mid \eta, y)$$

and

$$\pi(\eta \mid \theta, y)$$

respectively.

First we look at the posterior

$$\pi(\boldsymbol{\eta}, \theta \mid \mathbf{y}) \propto \pi(\theta)\pi(\boldsymbol{\eta} \mid \theta) \prod_{t=1}^{T} \pi(y_t \mid \eta_t, \theta)$$

Where

$$\pi(\theta) = exp(-\theta)$$

(pdf of gamma(1,1))

$$\pi(\eta \mid \theta)$$

is the second order walk model

$$\prod_{t=1}^{T} \pi(y_t \mid \eta_t, \theta)$$

is the likelihood of the normal distribution. Taking the log likelihood and inserting the known paramteres:

$$\pi(\boldsymbol{\eta}, \theta \mid \mathbf{y}) \propto \frac{\theta^{(T-2)/2}}{(2\pi)^{1/2}} \exp(-\theta - \frac{\theta}{2} \sum_{t=3}^{T} (\eta_t - 2\eta_{t-1} + \eta_{t-2})^2 - \frac{1}{2} \sum_{t=1}^{T} (y_t - \eta_t)^2)$$

The full conditional for theta is then (removed the terms that doesnt depend on theta)

$$\pi(\theta \mid \mathbf{y}, \boldsymbol{\eta}) \propto \theta^{\frac{T}{2}-1} exp(-\theta(1 + \frac{1}{2}\sum_{t=3}^{T}(\eta_t - 2\eta_{t-1} + \eta_{t-2})^2$$

We recognize this as the gamma function with paramter $\alpha = T/2$ and $\beta = 1 + \frac{1}{2}\sum_{t=3}^{T}(\eta_t - 2\eta_{t-1} + \eta_{t-2})^2$

By introducing Q as the precision matrix to account for the "markov term" in the linear predictors we can write the term in matrix notation. We define this as

$$Q = \theta \mathbf{L}\mathbf{L}^{\mathbf{T}}$$

where L is a T X (T-2) matrix that accounts for the coefficents in the "markov term".

$$\mathbf{L} = \begin{bmatrix} 1 & -2 & 1 & 0 & 0 & 0 & \dots \\ 0 & 1 & -2 & 1 & 0 & 0 & \dots \\ \vdots & & \ddots & \ddots & \ddots & & \\ 0 & 0 & 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & -2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

The full conditional in matrix notation is then:

$$\pi(\theta \mid \mathbf{y}, \boldsymbol{\eta}) \propto \theta^{\frac{T}{2}-1} exp(-\theta - \frac{1}{2}\boldsymbol{\eta}^T \mathbf{Q}\boldsymbol{\eta})$$

To get the exact gamma distribution we define $Q* = \frac{1}{\theta}Q$ and we end up with

$$\pi(\theta \mid \mathbf{y}, \boldsymbol{\eta}) \propto \theta^{\frac{T}{2}-1} exp(-\theta(1 + \frac{1}{2}\boldsymbol{\eta}^T \mathbf{Q}^* \boldsymbol{\eta}))$$

Now for the full conditional for $\boldsymbol{\eta}$

$$\pi(\boldsymbol{\eta} \mid \beta, \mathbf{y}) \propto exp(-\frac{\theta}{2}\sum_{t=3}^{T}(\eta_t - 2\eta_{t-1} + \eta_{t-2})^2 - \frac{1}{2}\sum_{t=1}^{T}(y_t - \eta_t)^2))$$

In matrix notation

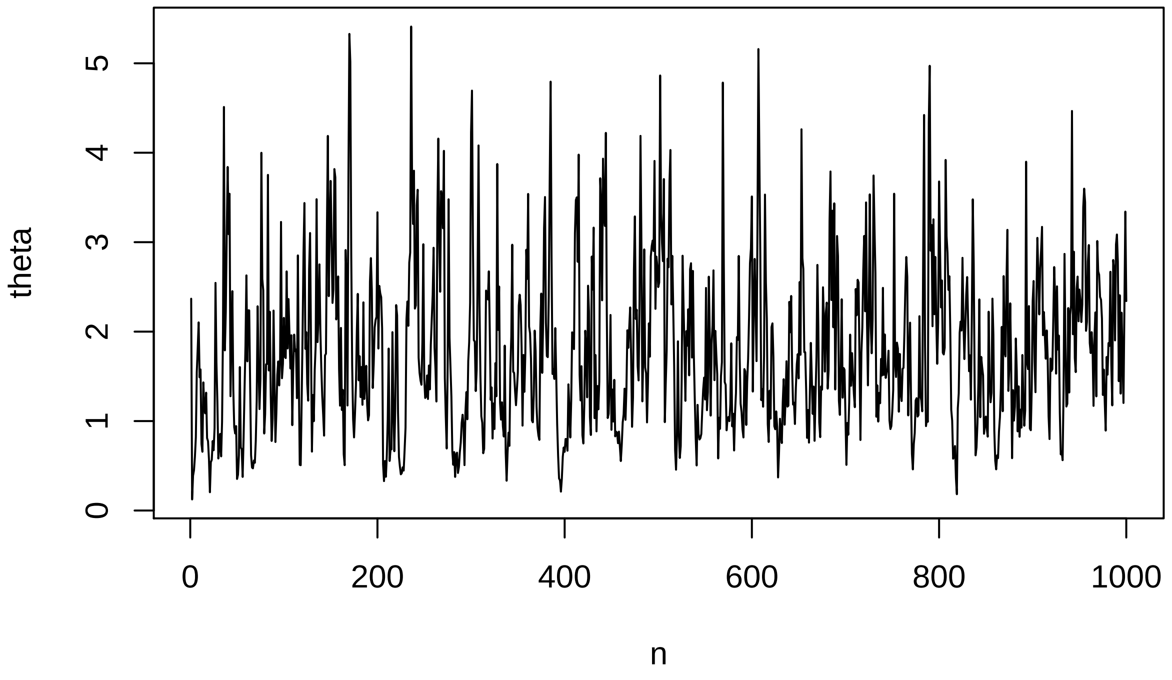
$$= exp(-\frac{1}{2}(\boldsymbol{\eta}^T \mathbf{Q}\boldsymbol{\eta}) + (\mathbf{y} - \boldsymbol{\eta})^T(\mathbf{y} - \boldsymbol{\eta})))$$

$$= exp(-\frac{1}{2}\boldsymbol{\eta}^T(\mathbf{Q} + \mathbf{I})\boldsymbol{\eta} + \mathbf{y}^T\boldsymbol{\eta})$$

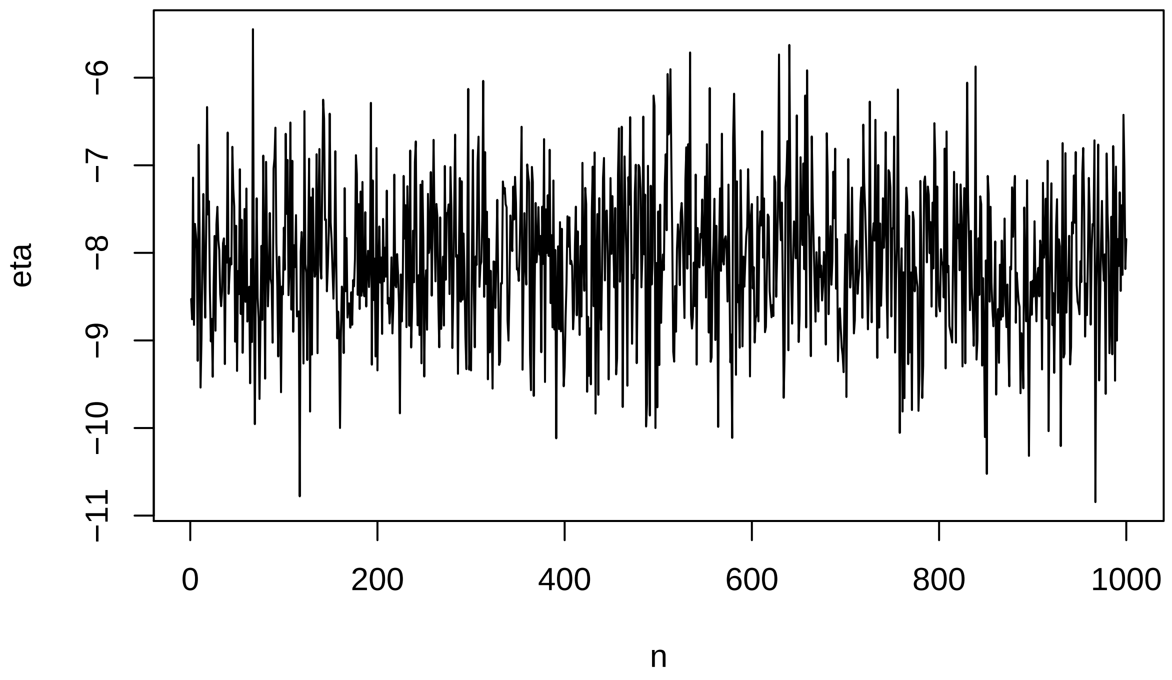
2)

```r
#Code for the Precision matrix. Initialising theta to be 1 so it can generate Q and Q*
precMat <- function(Tdim, theta=1){
  L <-diag(1, Tdim, Tdim-2)
  L[row(L) - col(L) == 1] <- -2
  L[row(L) - col(L) == 2] <- 1
  Q <- theta*L%*%t(L)
  return(Q)
}
```

```r
#Code for Gibbs sampling routine
gibbsSampling <- function(y, n){
  theta <- rep(0,n)
  Tdim <- length(y)
  A <- precMat(Tdim)
  Idmat <- diag(1,Tdim)
  eta <- matrix(0,n,Tdim)
  theta[1] <- rgamma(1,1,1)
  eta[1,] <- y

  #Iterating and calculate the next theta and eta based on the previous and conditional to each other
  for(i in 2:n){
    rate <- 1 + 0.5*t(eta[i-1,]) %*% A %*% eta[i-1,]
    theta[i] <- rgamma(1,shape = Tdim/2, rate = rate)
    var <- solve(A*theta[i] + Idmat , sparse = TRUE)
    mean <- var %*% y
    eta[i,] <- rmvnorm(mean, var)

  }
  samples <- cbind(theta, eta)
  return(samples)
}
test  <- gibbsSampling(gaus$V1, 1000)
#Plot of the theta and first eta.
plot(test[,1], type="l",xlab="n", ylab="theta")
```

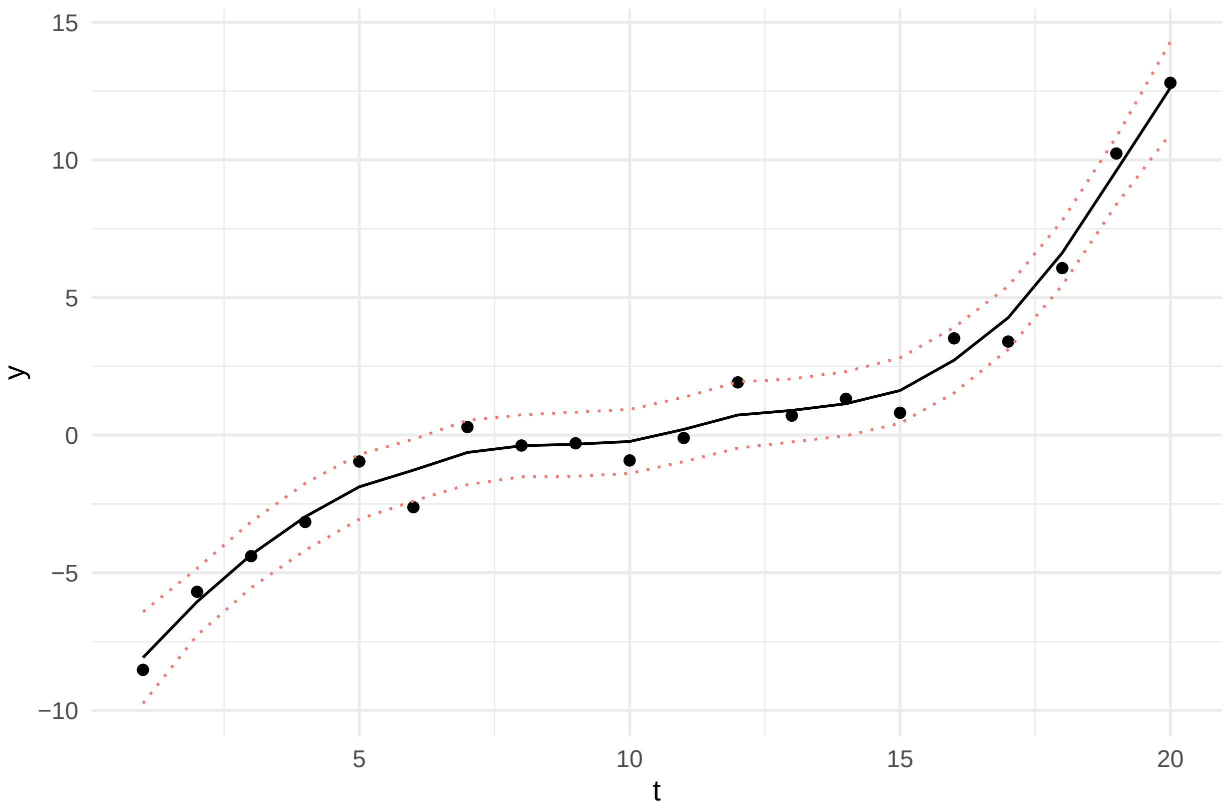


```r
plot(test[,2], type="l",xlab="n", ylab="eta")
```



From these plots we dont notice any burn in period for neither eta og theta. This is because we dont see any patterns that stands out for the first x iterations. However, to account for small patterns that are hard to notice, we choose to omit the first 100 iterations.

```
#Omitting the first 100 samples to get rid of the "burnn" period
newtest <- test[-c(1:100),]

meanvec <- colMeans(newtest)[-1]
varvec <- apply(newtest, 2, sd)[-1]
#Confidence interval
upperlim <- meanvec + qnorm(0.025, lower.tail = FALSE)*varvec
lowerlim <- meanvec - qnorm(0.025, lower.tail = FALSE)*varvec

test3 <- cbind(gaus, lowerlim, upperlim)

#Plot it
plot <- ggplot(data = test3, aes(x = x, y = meanvec)) + geom_point(aes(x = x, y = V1 )) + geom_line() +
plot <-  plot + labs(y = "y", x = "t", caption = "Estimate of the smooth effect, where the dotted lines
plot +  theme(legend.position="none")
```



Estimate of the smooth effect, where the dotted lines represents a 95% confidence interval

3)

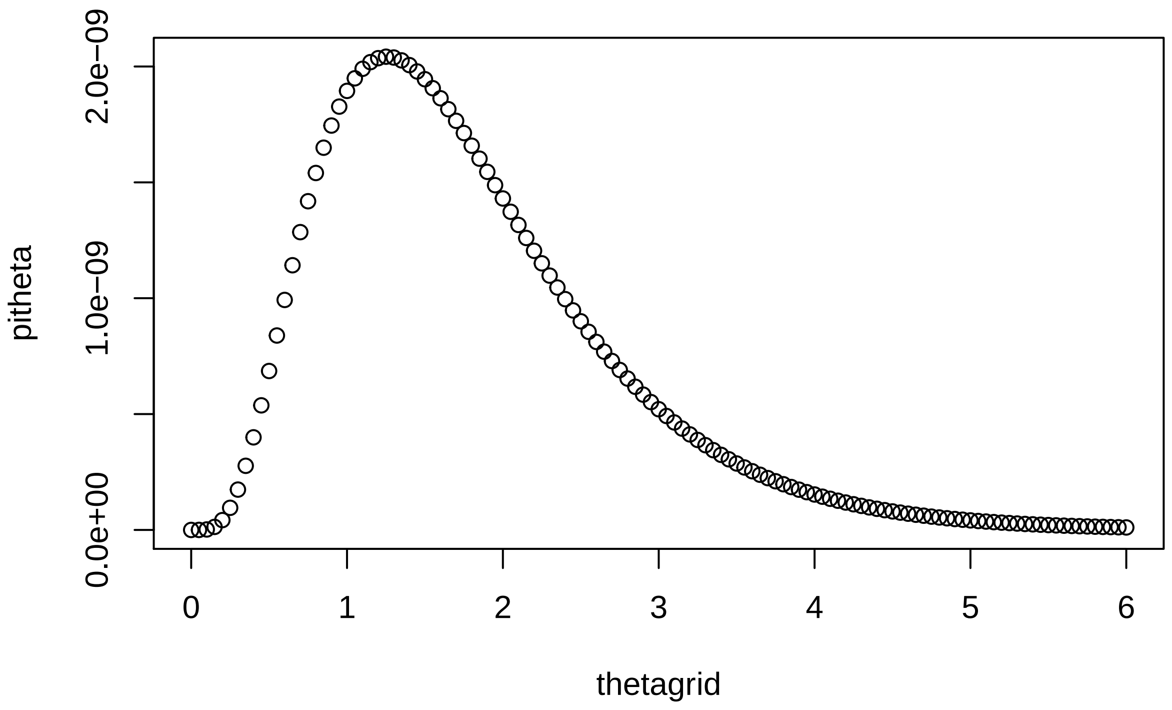
$$\pi(\theta \mid \mathbf{y}) \propto \frac{\pi(\mathbf{y} \mid \boldsymbol{\eta}, \theta)\pi(\boldsymbol{\eta} \mid \theta)\pi(\theta)}{\pi(\boldsymbol{\eta} \mid \theta, \mathbf{y})}$$

$$\propto \frac{\theta^{\frac{T-2}{2}} exp(-\theta - \frac{1}{2}(\boldsymbol{\eta}^T(\mathbf{Q}+\mathbf{I})\boldsymbol{\eta} + \mathbf{y^T y} + \mathbf{y}^T)\boldsymbol{\eta})}{\mid \mathbf{Q}+\mathbf{I} \mid exp(-\frac{1}{2}(\boldsymbol{\eta} - (\mathbf{Q}+\mathbf{I})^{-1}\mathbf{y})^T)(\mathbf{Q}+\mathbf{I})(\boldsymbol{\eta} - (\mathbf{Q}+\mathbf{I})^{-1}\mathbf{y})}$$

The etas cancels out and we get

$$\propto \frac{\theta^{\frac{T-2}{2}}}{\mid \mathbf{Q}+\mathbf{I} \mid^{\frac{1}{2}}} exp(-\theta - \frac{1}{2}\mathbf{y}^T(\mathbf{I} - (\mathbf{Q}+\mathbf{I})^{\mathbf{-1}})\mathbf{y})$$

```
#Function to calulcate the pi of theta given y
piTheta<- function(theta){
  Tdim <- length(gaus$V1)
  thetl <- length(theta)
  pivec <- rep(0,thetl)
  det <- rep(0,thetl)
  Q <- precMat(Tdim)
  Idmat <- diag(Tdim)

  for(i in 1:thetl){
    det <- det(Q*theta[i]+ Idmat)
    pivec[i] <- theta[i]^(Tdim/2 -1)/sqrt(det)* exp(-theta[i]-0.5*t(gaus$V1) %*% (Idmat-solve(Q*theta[i
  }
  return(pivec)
}

#Defining the grid
delta <- 0.05
thetagrid <- seq(0,6,delta)
#smiple plot
pitheta <- piTheta(thetagrid)
plot(thetagrid, pitheta)
```



The plot shows the the values of $\pi(\theta \mid y)$ for values of theta from 0-6.

Comparing with the MCMC estimate it looks pretty similar as we see the highest frequency of theta is between 0.5 and 2.5 in both plots. (Note that this graph is not normalized)

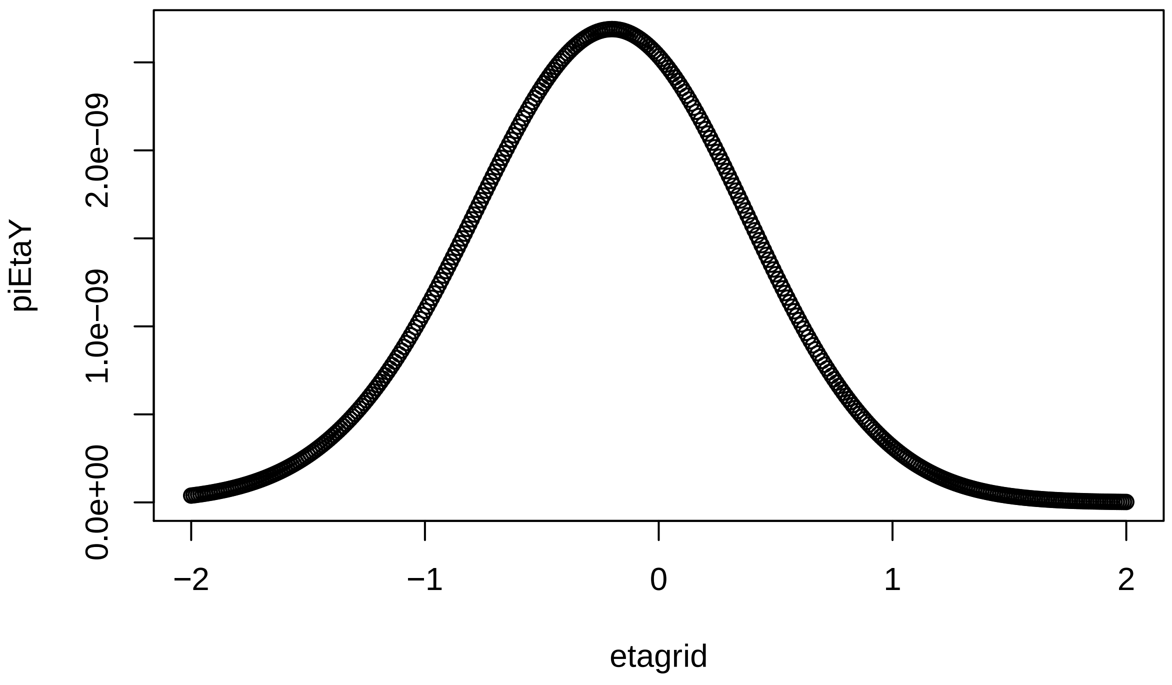
4)

By making a grid and sum up we can make a approcimation of the integral (numerical integration) as follows:

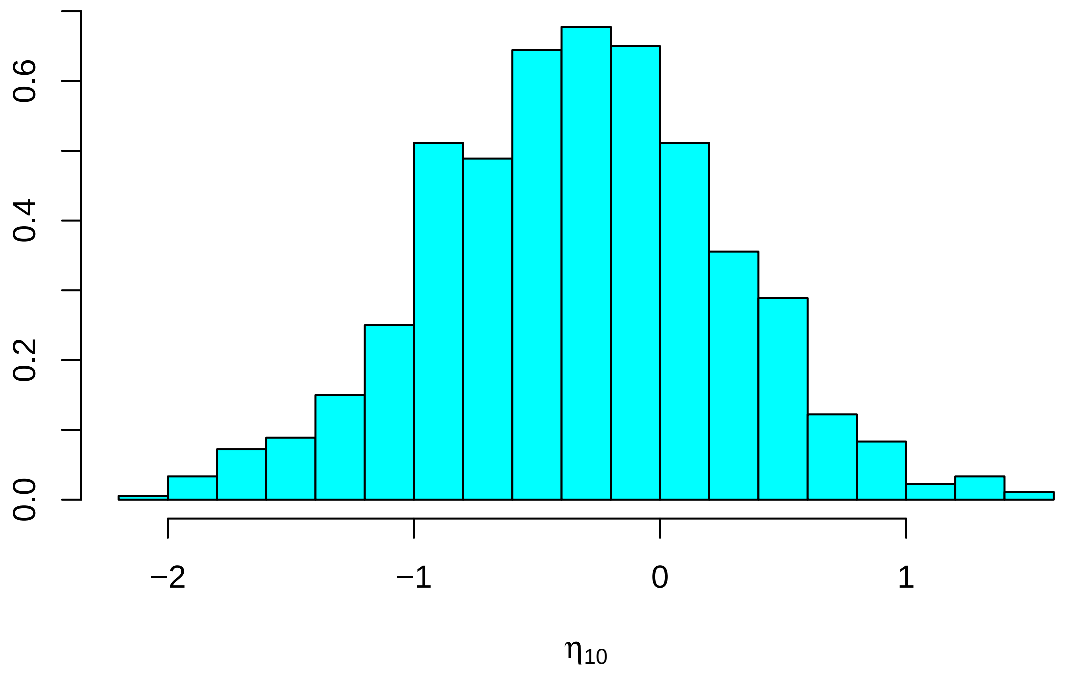
$$\pi(\eta_i \mid y) = \int \pi(\eta_i \mid y, \theta)\pi(\theta \mid y)d\theta \approx \sum_{j=1}^{J} \pi(\eta_i \mid y, \theta^j)\pi(\theta^j \mid y)\Delta j$$

Where Δj is the steplength for the grid we made for theta in the previous task.

```
#Function to calculate pi of eta given y and theta for eta10

piEtaYTheta <- function(etagrid, theta, i = 10){
  Tdim <- length(gaus$V1)
  Q <- precMat(Tdim, theta)
  A <- solve(Q + diag(Tdim))
  mean <- (A%*% gaus$V1)[i]
  var <- A[i,i]
  pi <- dnorm(etagrid, mean, sqrt(var))
  return(pi)
}

#Approximation of the integral for i = 10 using numerical integration with the previous functions
piEtaY <- function(thetagrid, etagrid){
  sum <- rep(0, length(etagrid))
  steplength <- thetagrid[2]-thetagrid[1]
  thetay <- piTheta(thetagrid)
  for (j in 1:length(thetagrid)){
    theta <- thetagrid[j]
    sum <- sum + piEtaYTheta(etagrid, theta) * thetay[j]*steplength
  }
  return(sum)
}

#COnstructing the etagrid
etagrid <- seq(-2,2,0.01)
piEtaY<- piEtaY(thetagrid, etagrid)

plot(etagrid, piEtaY)
```



```
est2 <- newtest[,10]

#Simple plot
truehist(est2, xlab = bquote(eta[10]))
```



The first plot is the plot from INLA method and the second from the Gibbs sampling we obtained in task 2. Both of them show the marginal posterior distribution for the smooth effect for η_{10} As we can see the mean is shifted slightly to the left of zero for both and they look quite similar.

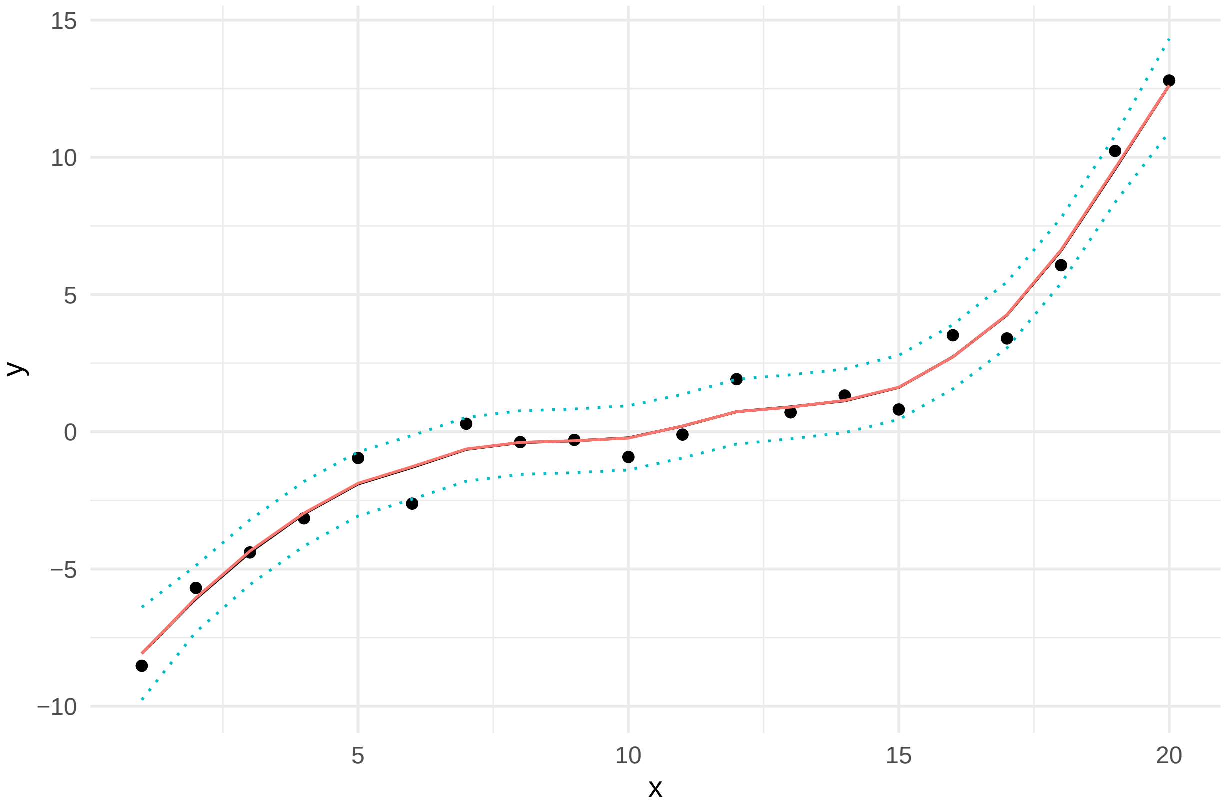
5)

```
y <- gaus$V1
x <- gaus$x

#Using the built in INLA
thetapri <- list(theta = list(prior = "log.gamma", param = c(1,1)))

formula <- y ~ f(x, model = "rw2", hyper = thetapri, constr = FALSE)  - 1

fit <- inla(formula = formula, family = "gaussian", data = gaus, control.family = list(hyper=list(prec
```

Comparison with results obtined in part 2:

```
#Extracting mean and variance from the INLA model.
mean <- fit$summary.random$x$mean
sd<- fit$summary.random$x$sd
lower <- mean - qnorm(0.025, lower.tail = FALSE)*sd
upper <- mean + qnorm(0.025, lower.tail = FALSE)*sd

inladf <- cbind(x, mean, lower, upper)
inladf <- as.data.frame(inladf)

#GGplot
plot <- ggplot(data = inladf, aes(x = x, y = mean)) + geom_point(aes(x = x, y = gaus$V1 )) + geom_line(
```

```r
plot <-  plot + labs(y = "y", x = "x", caption = "Estimate of the smooth effect, where the dotted lines
plot +  theme(legend.position="none") + geom_line(aes(x = x, y = meanvec, color = "blue"))
```

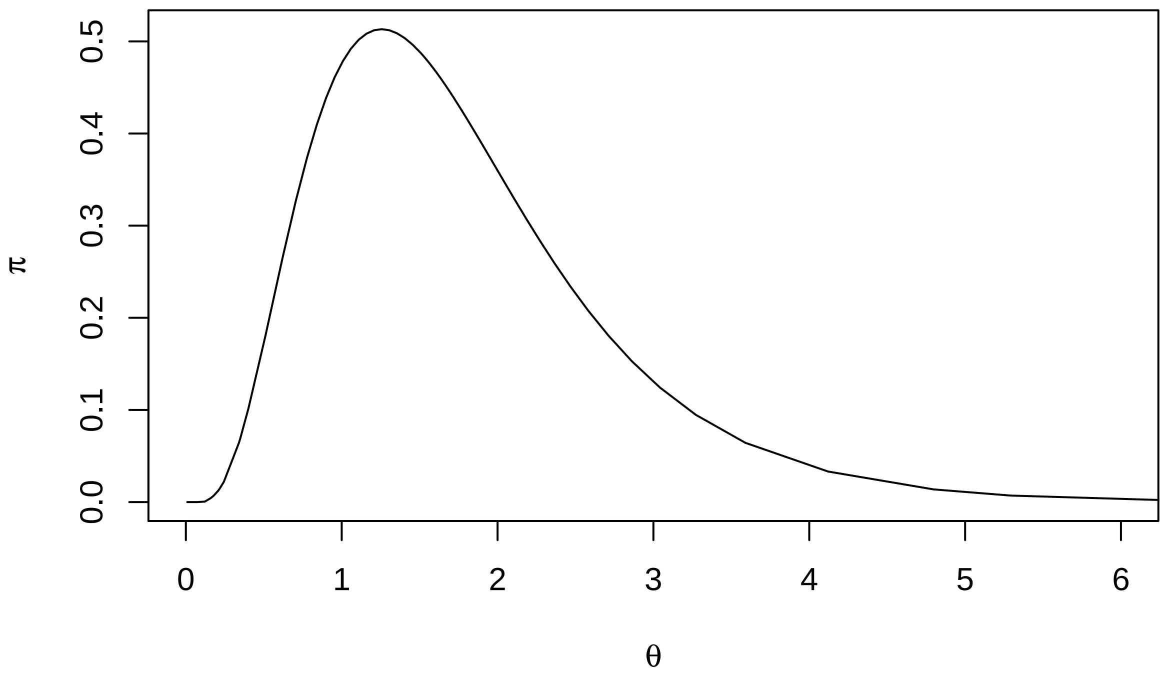


Estimate of the smooth effect, where the dotted lines represents a 95% confidence interval

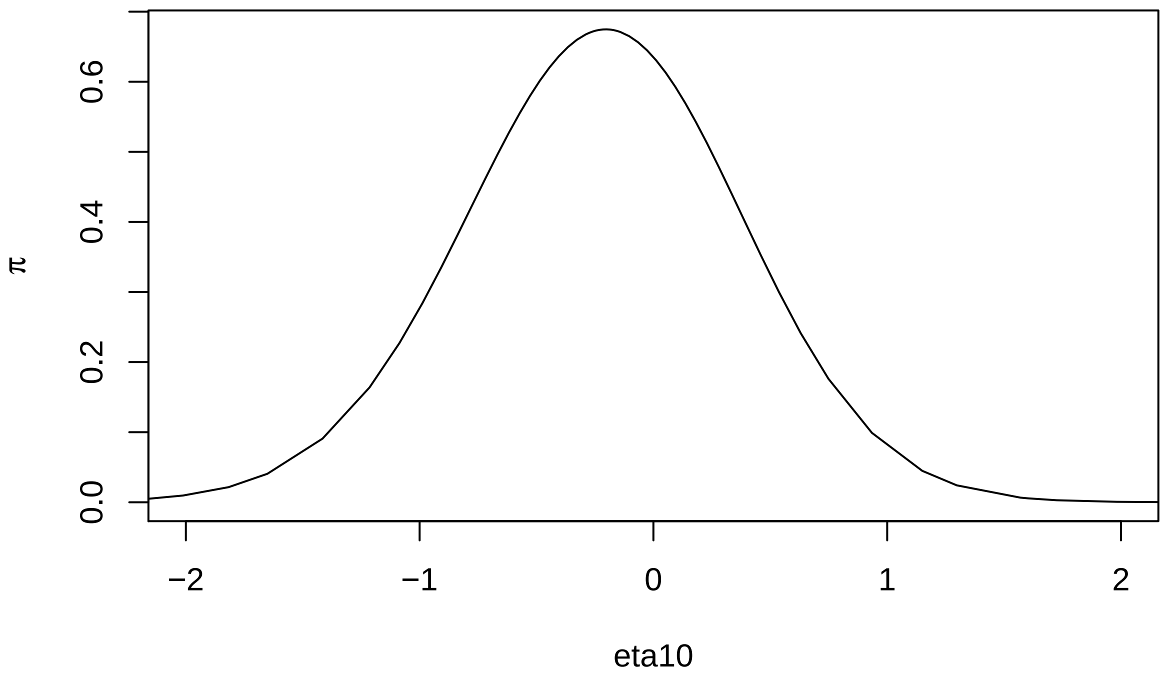
The graph compares the INLA method and the one obtained by MCMC in task 2. We see that the lines that represents the mean are overlapping to a degree that we can barely see that there are two different lines.

Comparison with results obtined in part 3 and 4:

```r
#Extracting theta and eta from the model and make simple plots.
theta3 <- fit$marginals.hyperpar$`Precision for x`
eta3 <- fit$marginals.random$x$index.10

plot(theta3, xlim = c(0,6), type = "l", xlab = bquote(theta), ylab = bquote(pi))
```



```r
plot(eta3, xlim = c(-2,2), type = "l", xlab = bquote(eta10), ylab = bquote(pi))
```



The first plot shows the the values of $\pi(\theta \mid y)$ for values of theta from 0-6 with the built in INLA scheme.

The second plot shows the marginal posterior distribution for the smooth effect for η_{10} with the built in INLA scheme.

Wee see that both the plots looks alike to the corresponding plots obtained in the previous tasks.