Project2

Martine Middelthon and Helene Behrens

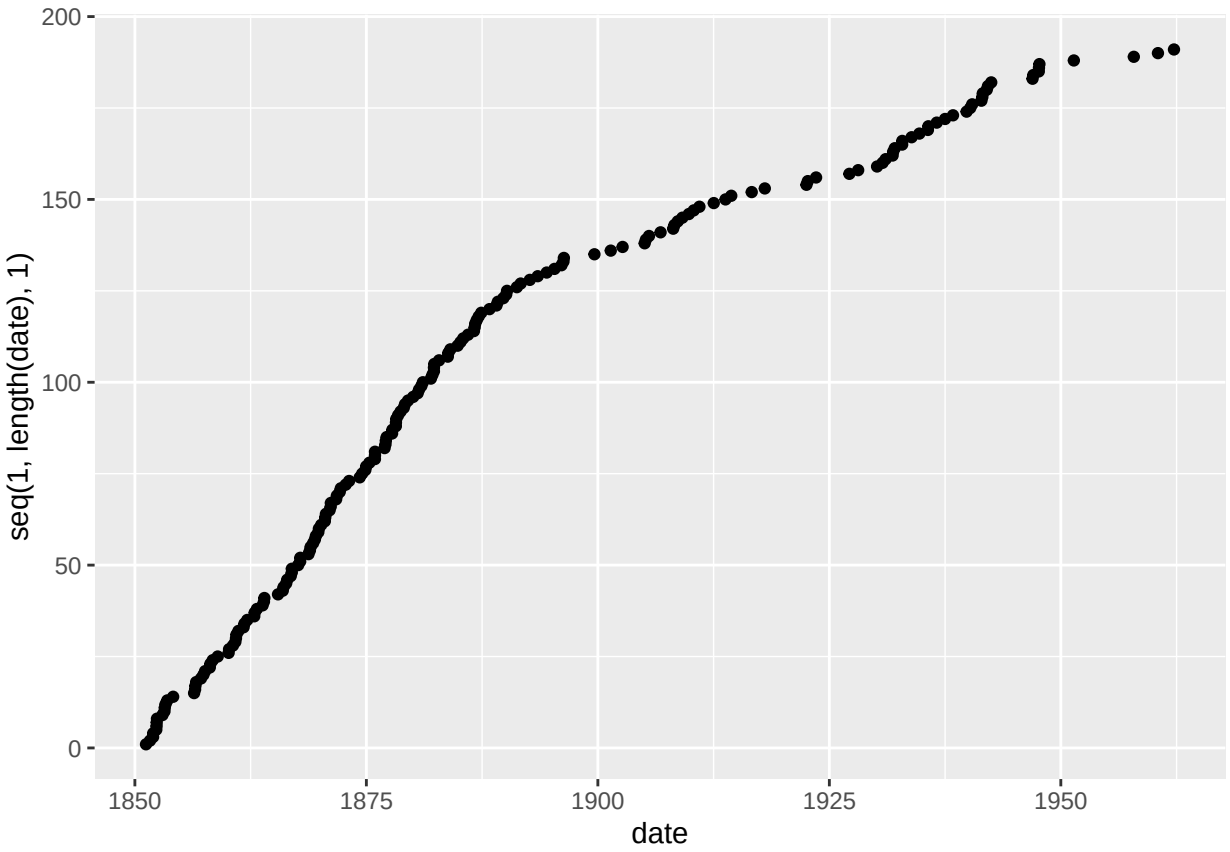
Problem A

We consider the built-in coal-mining data set in R.

1.

We adopt a hierarchical Bayesian model to analyse the data set. Below, we see a visualization of the data:

```
set.seed(124)
data(coal)
visualize <- ggplot() + geom_point(data=coal, aes(x=date,y=seq(1,length(date),1)))
visualize
```



We assume that the coal-mining disasters follow an inhomogeneous Poisson process with intensity function $\lambda(t)$. Furthermore, we assume that $\lambda(t)$ is piecewise constant with n break points, where t_0 and t_{n+1} are

the start- and end-times of the dataset, respectively, and t_k; $k = 1, \dots, n$ are break-points in the intensity function:

$$\lambda = \begin{cases} \lambda_{k-1} \text{ for } t \in [t_{k-1}, t_k), \ k = 1, \dots, n \\ \lambda_n \text{ for } [t_n, t_{n-1}]. \end{cases}$$

Our model parameters are then $(\beta, t_1, \dots, t_n, \lambda_0, \dots, \lambda_n)$. The likelihood function of the data can be derived to be

$$f(x \mid t_1, \dots, t_n, \lambda_1, \dots, \lambda_n) = \exp(-\int_{t_0}^{t_{n+1}} \lambda(t)\mathrm{d}t) \prod_{k=0}^{n} \lambda_k^{y_k} = \exp(-\sum_{k=0}^{n} \lambda_k(t_{k+1} - t_k)) \prod_{k=0}^{n} \lambda_k^{y_k},$$

where x is the observed data and y_k is the number of observed disasters in the corresponding interval. We assume that the t_i are apriori uniformly distributed on the allowed values, and that the λ_i are apriori independent of the t_i and of each other. Apriori, we assume all λ_i to be distributed from the same gamma distribution with shape parameter $\alpha = 2$ and scale parameter β;

$$f(\lambda_i \mid \beta) = \frac{1}{\beta^2} \lambda_i e^{-\frac{\lambda_i}{\beta}}, \quad \lambda_i \geq 0.$$

For β we use the improper prior

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

From now on, we assume $n = 1$, and the model parameters are then $\theta = (t_1, \lambda_0, \lambda_1, \beta)$.

2.

We wish to find an expression for the posterior distribution for θ given x, $f(\theta \mid x)$.

$$f(\theta \mid x) = f(x \mid \theta) f(\theta) f(x)^{-1}$$

$$\begin{aligned} f(\theta \mid x) &\propto f(x \mid \theta) f(\theta) \\ &\propto f(x \mid \theta) f(t_1) f(\lambda_0, \lambda_1, \beta) \\ &\propto f(x \mid \theta) f(t_1) f(\lambda_0, \lambda_1 \mid \beta) f(\beta) \\ &\propto f(x \mid \theta) f(t_1) f(\lambda_0 \mid \beta) f(\lambda_1 \mid \beta) f(\beta). \end{aligned}$$

For $n = 2$ we have that

$$f(t_1) = \begin{cases} \frac{1}{t_2 - t_1} & t_1 \in [t_0, t_2] \\ 0 & \text{otherwise} \end{cases}$$

and

$$f(x \mid \theta) = \exp(-\lambda_0(t_1 - t_0) - \lambda_1(t_2 - t_1)) \lambda_0^{y_0} \lambda_1^{y_1}.$$

Inserting for the remaining distributions, we get

$$f(\theta \mid x) \propto \frac{1}{\beta^5} \lambda_0^{y_0+1} \lambda_1^{y_1+1} \exp(-\lambda_0(t_1 - t_1 + 1/\beta)) \exp(-\lambda_1(t_2 - t_1 + 1/\beta)).$$

3.

We find the full conditionals for each of the elements in θ. In general, to find the full conditional of an element θ_i given x and the remaining parameters θ^{-i} we have

$$f(\theta_i \mid x, \theta^{-i}) = \frac{f(\theta, x)}{f(x, \theta^{-i})} = \frac{f(\theta, x)}{\int f(\theta, x) \mathrm{d}\theta_i}$$

$$= \frac{f(\theta \mid x)f(x)}{\int f(x)f(\theta \mid x)\mathrm{d}\theta_i} = \frac{f(\theta \mid x)}{\int f(\theta \mid x)\mathrm{d}\theta_i}.$$

For λ_0 we get:

$$f(\lambda_0 \mid x, \beta, \lambda_1, t_1) = \frac{\lambda_0^{y_0+1}\exp(-\lambda_0()t_1 - t_0 + 1/\beta)}{\int_0^\infty \lambda_0^{y_0+1}\exp(-\lambda_0()t_1 - t_0 + 1/\beta)\mathrm{d}\lambda_0}$$

$$= \frac{\lambda_0^{y_0+1}(t_1 - t_0 + 1/\beta)^{y_0+2}}{\Gamma(y_0+2)} \cdot e^{-\lambda_0(t_1-t_0+1/\beta)} \sim \text{Gamma}(y_0+2, 1/(t_1 - t_0 + 1/\beta)).$$

Similarily, we get for λ_1:

$$f(\lambda_1 \mid x, \beta, \lambda_0, t_1) = \frac{\lambda_1^{y_1+1}(t_2 - t_1 + 1/\beta)^{y_1+2}}{\Gamma(y_1+2)} \cdot e^{-\lambda_0 1 t_2 - t_1 + 1/\beta)} \sim \text{Gamma}(y_1+2, 1/(t_2 - t_1 + 1/\beta)).$$

For β we get:

$$f(\beta \mid x, \lambda_0, \lambda_1, t_1) = \frac{1/\beta^5 \cdot e^{-(\lambda_0+\lambda_1+1)/\beta}}{\int_0^\infty 1/\beta^5 \cdot e^{-(\lambda_0+\lambda_1+1)/\beta}\mathrm{d}\beta}$$

$$= \frac{(\lambda_0+\lambda_1+1)^4 \cdot e^{-(\lambda_0+\lambda_1+1)/\beta}}{6\beta^5} \sim \text{invGamma}(4, 1/(\lambda_0+\lambda_1+1)).$$

Finally, we have for t_1:

$$f(t_1 \mid x, \lambda_0, \lambda_1, \beta) = \frac{\lambda_0^{y_0+1}\lambda_1^{y_1+1}e^{t_1(\lambda_1-\lambda_0)}}{\int \lambda_0^{y_0+1}\lambda_1^{y_1+1}e^{t_1(\lambda_1-\lambda_0)}\mathrm{d}t_1}$$

$$\propto \lambda_0^{y_0+1}\lambda_1^{y_1+1}e^{t_1(\lambda_1-\lambda_0)}.$$

We recall that y_0 and y_1 are dependent on t_1, and so this does not follow an exponential distribution.

4.

A single-site MCMC algoritm is implemented below. We have used Gibbs sampling for β, λ_0 and λ_1, meaning that we have used their conditional distributions as proposal distributions. This gives us an acceptance probability of exactly one for all samples of β, λ_0 and λ_1. As we do not know how to sample directly from the full conditional of t_1, we choose a random-walk approach for this. For each iteration we sample the proposed t_1^i from a normal distribution with mean t_1^i and variance σ^2. As the normal distribution is symmetric, we get that the proposal ratio is one:

$$\frac{Q(t^{i-1} \mid t')}{Q(t' \mid t^{i-1})} = \frac{\exp(-\frac{1}{2}(\frac{t^{i-1}-t'}{\sigma})^2)}{\exp(-\frac{1}{2}(\frac{t'-t^{i-1}}{\sigma})^2)} = \exp(0) = 1.$$

Thus, the acceptance ratio is

$$\alpha = \min(1, \frac{\pi(t', ...)}{\pi(t^{i-1}, ...)}) = \min\left(1, \frac{\lambda_0^{y_0'+1} \cdot \lambda_1^{y_1'+1} \cdot e^{t'(\lambda_1-\lambda_0)}}{\lambda_0^{y_0^{i-1}+1} \cdot \lambda_1^{y_1^{i-1}+1} \cdot e^{t^{i-1}(\lambda_1-\lambda_0)}}\right)$$

$$= \min\left(1, \lambda_0^{y_0'-y_0^{i-1}}\lambda_1^{y_1'-y_1^{i-1}}e^{(\lambda_1-\lambda_0)(t'-t^{i-1})}\right),$$

where y_0^{i-1}, y_1^{i-1} and t^{i-1} are the values of y_0, y_1 and t_1 from the last iteration, respectively.

```r
Alg.4 <- function(n,t1.0,l0.0,l1.0,beta.0, obs, sigma){
  # initializing parameters with prior estimates

  theta <- matrix(0L, nrow=7, ncol=n)
  rownames(theta) <- c("t1", "l0", "l1", "beta", "y0", "y1","accept")

  t0 <- coal$date[1]
  t2 <- tail(coal$date,1)

  theta["t1",1] <- t1.0
  # disasters before t1, subtracting the first element which is not a disaster
  theta["y0",1] <- max(which(coal$date < theta["t1",1])) - 1
  # disasters at and after t1, subtracting start time and end time element
  theta["y1",1] <- length(coal$date) - theta["y0",1] - 2

  theta["l0",1] <- l0.0
  theta["l1",1] <- l1.0
  theta["beta",1] <- beta.0

  for(i in 2:n){

    # MH-step; sample t1 from normal with mean t1
    y <- rnorm(1,theta["t1",i-1],sigma)                # proposal for t1
    if (y > t2 | y < t0){
      # reject
      theta["t1",i] <- theta["t1",i-1]
      theta["beta",i] <- theta["beta",i-1]
      theta["l0",i] <- theta["l0",i-1]
      theta["l1",i] <- theta["l1",i-1]
      theta["accept",i] <- 0
    }
    else {
      y0_prop <- max(which(coal$date <= y)) - 1
      y1_prop <- length(coal$date) - y0_prop - 2
      u <- runif(1)
      # acceptance probability:
      alpha <- min(1, theta["l0",i-1]^(y0_prop - theta["y0",i-1])
                   *theta["l1",i-1]^(y1_prop  - theta["y1",i-1])
                   *exp((theta["l1",i-1]-theta["l0",i-1])*(y - theta["t1",i-1])))

      if (u < alpha){
        # accept
        theta["t1",i] <- y
        theta["accept", i] <- 1
      }
      else{
        # reject
        theta["t1",i] <- theta["t1",i-1]
        theta["accept", i] <- 0
      }
      # sample beta from inverse gamma distribution
      theta["beta",i] <- rinvgamma(1,shape=4,
                                   scale=1/(theta["l0",i-1]+ theta["l1",i-1] + 1))
```

```
      # sample l0 from gamma
      theta["l0",i] <- rgamma(1,shape=theta["y0",i-1] + 2,
                              scale=1/(theta["t1",i-1] - t0 + 1/theta["beta",i-1]))
      # sample l1 from gamma
      theta["l1",i] <- rgamma(1,shape = theta["y1",i-1] + 2,
                              scale=1/(t2 - theta["t1",i-1] + 1/theta["beta",i-1]))

    }
    # update y0 based on current t1 and obs
    theta["y0",i] <- max(which(coal$date <= theta["t1",i])) - 1
    # update y1 based on current t1 and obs
    theta["y1",i] <- length(coal$date) - theta["y0",i] - 2
  }
  return(theta)
}

relevant.plots <- function(theta.df,coal,burn.in){

  t1.mean <- mean(theta.df[burn.in:length(theta.df$t1),]$t1)

  hist <- ggplot(theta.df,aes(x=t1)) + geom_histogram(binwidth = 1)
  hist <- hist + geom_vline(xintercept = t1.mean, col="red") +
    labs(x="t1",y="occurrances")

  time.proc <- ggplot(theta.df,aes(x=seq(1,length(t1),1),y=t1)) +
    geom_line() + labs(x="iterations",y="t1")

  l0.proc <- ggplot(theta.df,aes(x=seq(1,length(t1),1),y=l0)) +
    geom_line() + labs(x="iterations",y="lambda0")

  l1.proc <- ggplot(theta.df,aes(x=seq(1,length(t1),1),y=l1)) +
    geom_line() + labs(x="iterations",y="lambda 1")

  beta.proc <- ggplot(theta.df,aes(x=seq(1,length(t1),1),y=beta)) +
    geom_line() + labs(x="iterations", y="beta")

  l0.mean <- mean(theta.df[burn.in:length(theta.df$t1),]$l0)

  l1.mean <- mean(theta.df[burn.in:length(theta.df$t1),]$l1)

  #values in helplines:
  x0 <- coal$date[1]
  yend0 <- l0.mean*(t1.mean - x0)
  xend1 <- tail(coal$date,1)
  yend1 <- l1.mean*(xend1 - t1.mean) + yend0

  compare <- ggplot() + geom_point(data=coal, aes(x=date,y=seq(1,length(date),1)))
  compare <- compare + geom_segment(aes(x=x0, xend=t1.mean, y = 0, yend = yend0))
  compare <- compare + geom_segment(aes(x = t1.mean, xend=xend1, y = yend0, yend = yend1)) +
    labs(x="time",y="disasters")

  g.arr <- ggarrange(ggarrange(hist,compare,ncol=2),
            ggarrange(l0.proc,l1.proc,ncol=2),
```

```
            ggarrange(time.proc,beta.proc,ncol=2),
            nrow=3)
  return(g.arr)
}
```

We plot the result for some different initial values and values of tuning parameters to evaluate the performance of the algorithm. First, we compare the results for different values of σ:

```
th1 <- Alg.4(40000,1940,2.5,1,0.2,coal,1)
th1.df <- as.data.frame(t(th1))
th5 <- Alg.4(40000,1940,2.5,1,0.2,coal,5)
th5.df <- as.data.frame(t(th5))
th10 <- Alg.4(40000,1940,2.5,1,0.2,coal,10)
th10.df <- as.data.frame(t(th10))
th20 <- Alg.4(40000,1940,2.5,1,0.2,coal,20)
th20.df <- as.data.frame(t(th20))

# acceptance probabilities:
accept.1 <- mean(th1.df$accept)
cat("Acceptance propability with sigma = 1: ",accept.1,"\n")
```

```
## Acceptance propability with sigma = 1:  0.46445
```

```
accept.5 <- mean(th5.df$accept)
cat("Acceptance propability with sigma = 5: ",accept.5,"\n")
```

```
## Acceptance propability with sigma = 5:  0.319875
```

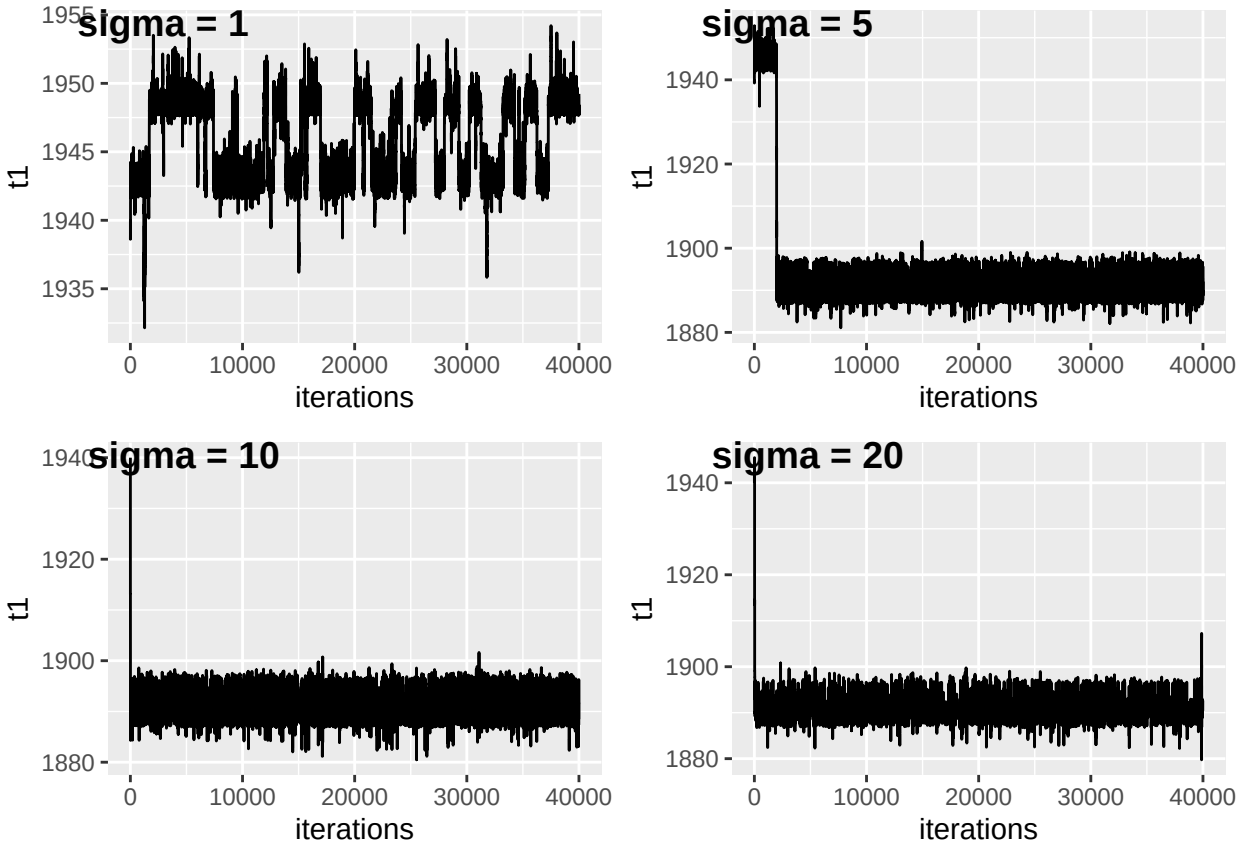
```
accept.10 <- mean(th10.df$accept)
cat("Acceptance propability with sigma = 10: ",accept.10,"\n")
```

```
## Acceptance propability with sigma = 10:  0.19415
```

```
accept.20 <- mean(th20.df$accept)
cat("Acceptance propability with sigma = 20: ",accept.20,"\n")
```

```
## Acceptance propability with sigma = 20:  0.0997
```

```
sig.plot.1 <- ggplot(th1.df,aes(x=seq(1,length(t1),1),y=t1)) + geom_line() +
  labs(y="t1",x="iterations")
sig.plot.5 <- ggplot(th5.df,aes(x=seq(1,length(t1),1),y=t1)) + geom_line() +
  labs(y="t1",x="iterations")
sig.plot.10 <- ggplot(th10.df,aes(x=seq(1,length(t1),1),y=t1)) + geom_line() +
  labs(y="t1",x="iterations")
sig.plot.20 <- ggplot(th20.df,aes(x=seq(1,length(t1),1),y=t1)) + geom_line() +
  labs(y="t1",x="iterations")

ggarrange(ggarrange(sig.plot.1,sig.plot.5,ncol=2,labels=c("sigma = 1","sigma = 5")),
          ggarrange(sig.plot.10,sig.plot.20,ncol=2,labels=c("sigma = 10", "sigma = 20")),
          nrow = 2)
```



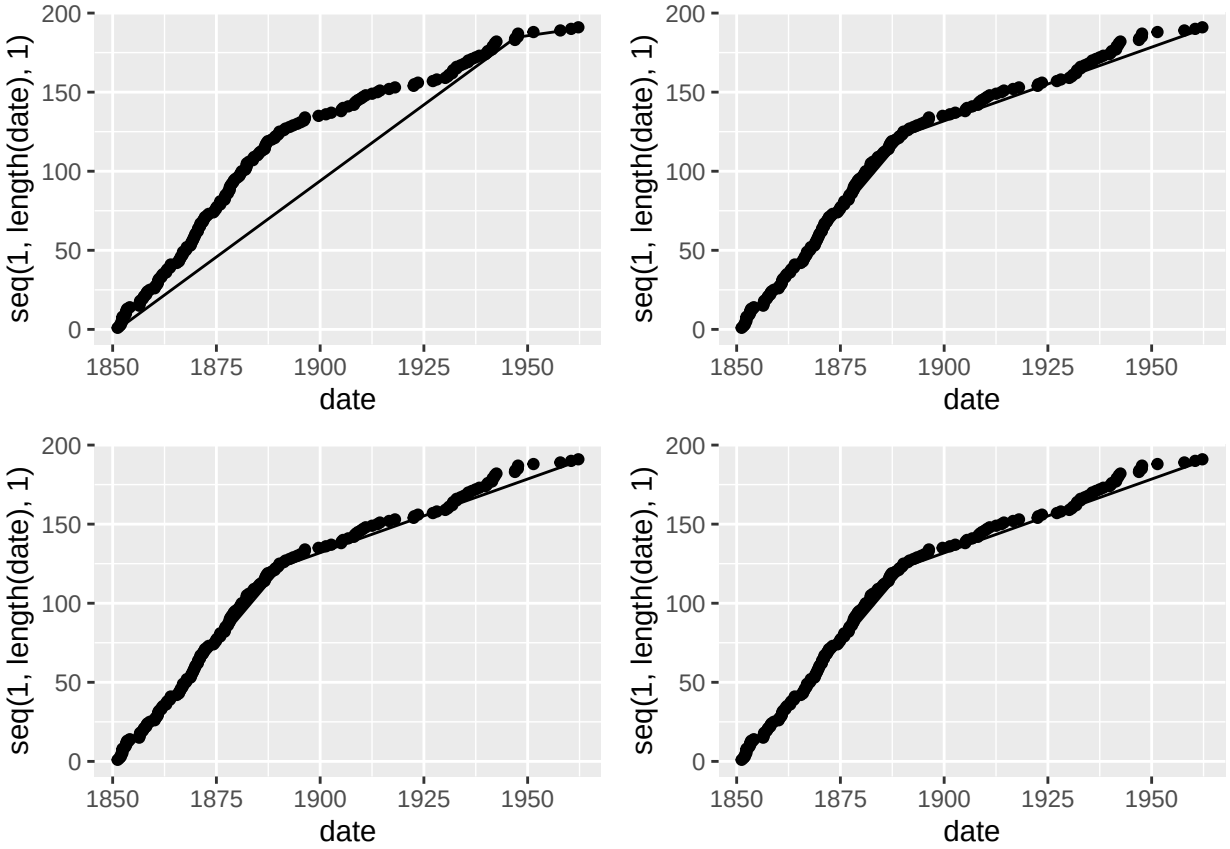
5. and 6.

Above, we ran the algorithm for $\sigma = 1, 5, 10, 20$ for an initial $t_1 = 1940$, which is quite far from the correct value. We set the intial value so far off to more clearly see the relation between the burn-in time and the value of σ. We observe that the burn-in period, i.e. how many iterations the alogrithm use to reach the correct domain, is significantly higher for lower values of σ. From the results so far, it seems like a σ somewhere between 5 and 10 is ideal, as $\sigma = 5$ had an acceptance probability in the desired domain (between 20% and 50% for random-walk proposals), while $\sigma = 10$ had a lower acceptance probability, but a shorter burn-in period.

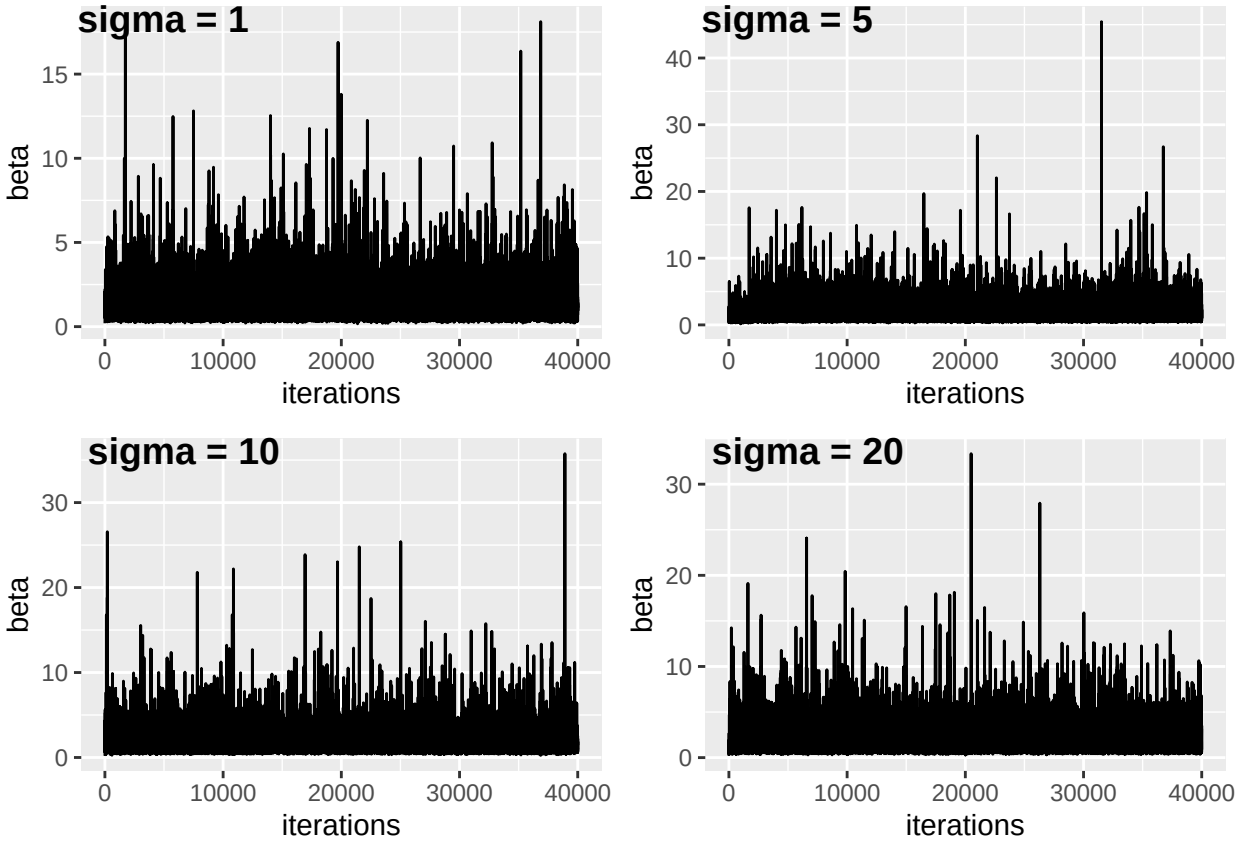
We set the burn-in periods to $[35000, 7000, 1000, 1000]$ respectively, and look at the fit of the resulting values:

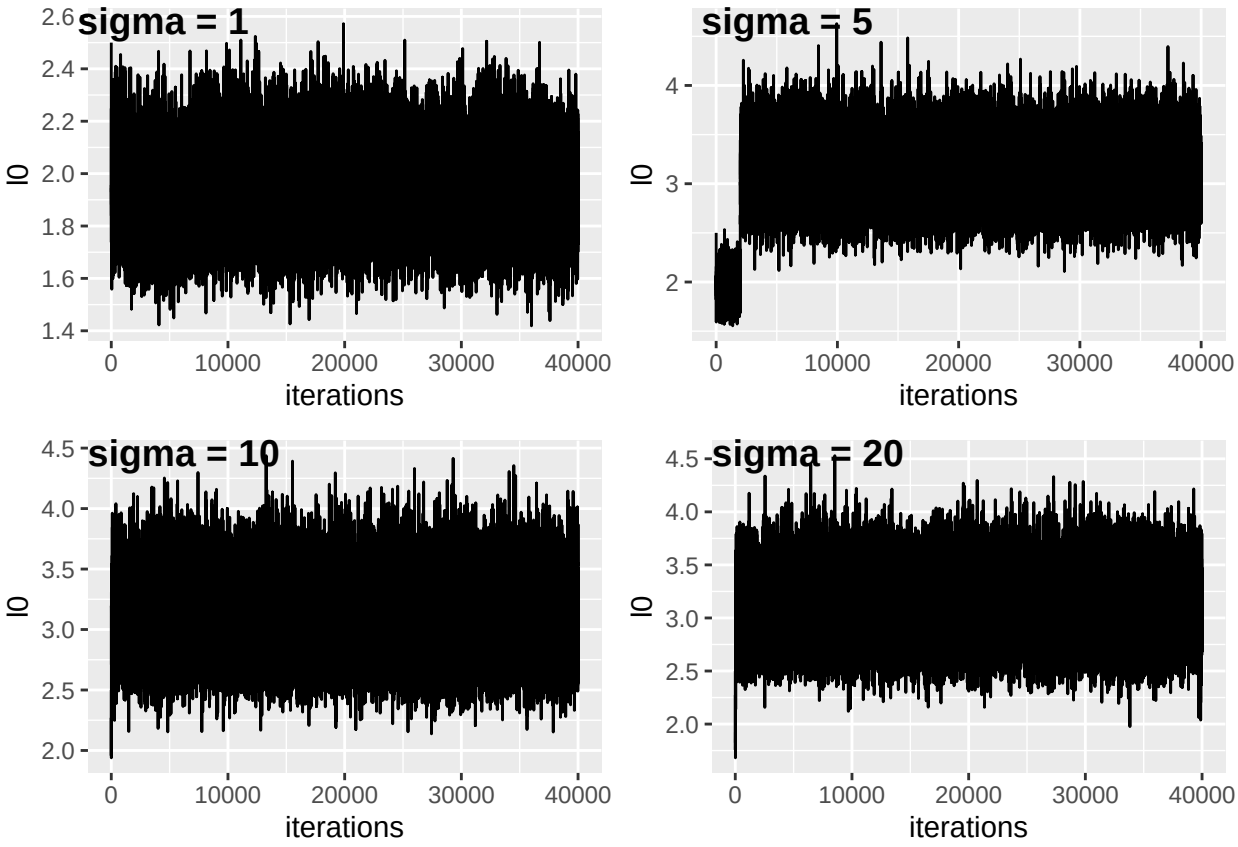
```
compare.fit <- function(df,burnin){
  t1.mean <- mean(df[burnin:length(df$t1),]$t1)

  l0.mean <- mean(df[burnin:length(df$t1),]$l0)

  l1.mean <- mean(df[burnin:length(df$t1),]$l1)

  #values in helplines:
  x0 <- coal$date[1]
  yend0 <- l0.mean*(t1.mean - x0)
  xend1 <- tail(coal$date,1)
  yend1 <- l1.mean*(xend1 - t1.mean) + yend0
```

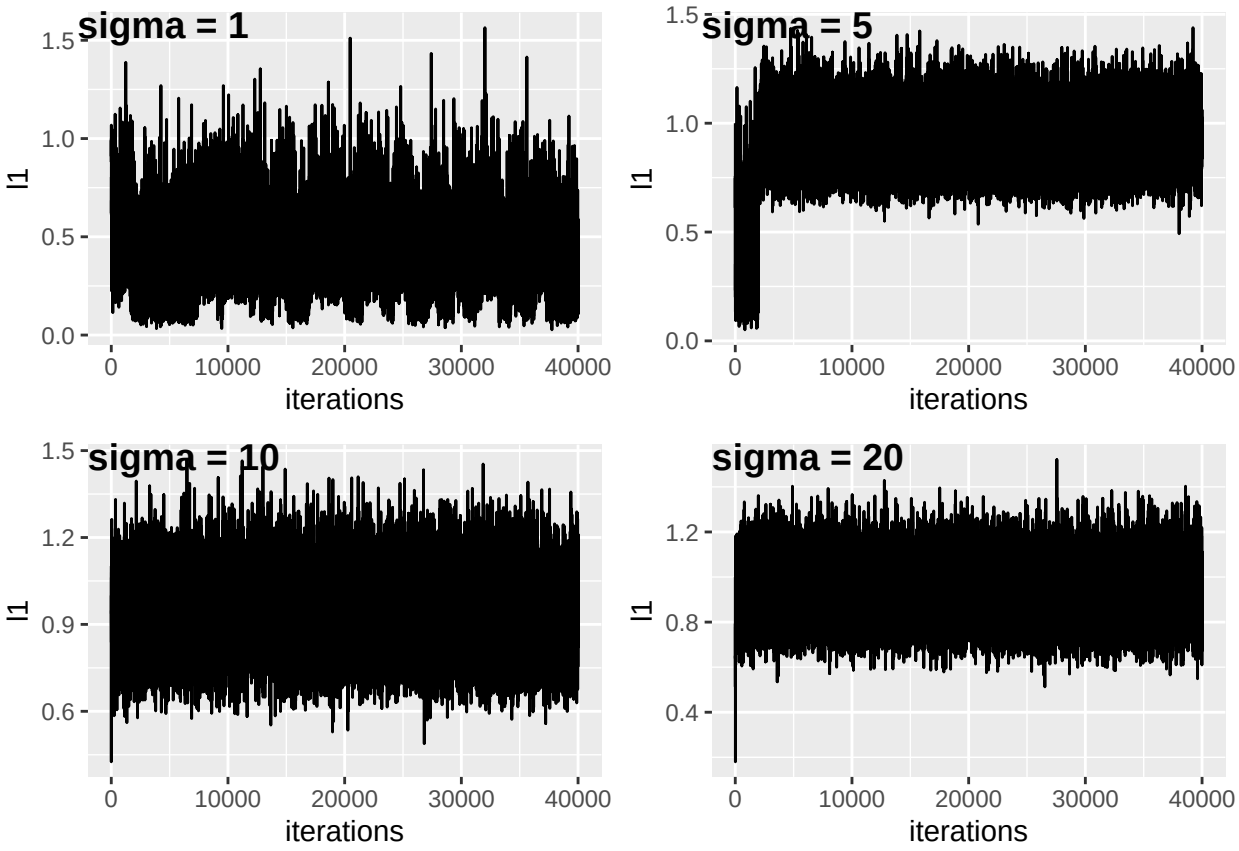
```
  compare <- ggplot() + geom_point(data=coal, aes(x=date,y=seq(1,length(date),1)))
  compare <- compare + geom_segment(aes(x=x0, xend=t1.mean, y = 0, yend = yend0))
  compare <- compare + geom_segment(aes(x = t1.mean, xend=xend1, y = yend0, yend = yend1))
  compare
}
cf.1 <- compare.fit(th1.df,35000)
cf.5 <- compare.fit(th5.df,5000)
cf.10 <- compare.fit(th10.df,1000)
cf.20 <- compare.fit(th20.df,1000)

ggarrange(ggarrange(cf.1,cf.5,ncol=2),ggarrange(cf.10,cf.20,ncol=2),nrow = 2)
```



We observe that when we take the burn-in into consideration, the algoritm seems to produce a good fit with all values of σ, with an exception of when $\sigma = 1$, where we observe that the algorithm did still not reach good convergence. To evaluate the mixing properties of the algortim, we look at how well the algoritm explores the range of possible values for the parameters. In particular, we look at how the tuning parameter influence the mixing properties of the simulated Markov Chain with respect to β, λ_0 and λ_1:


sigma = 1
sigma = 5
sigma = 10
sigma = 20
beta
iterations
0
10000
20000
30000
40000
0
5
10
15
20
30
40


sigma = 1
sigma = 5
sigma = 10
sigma = 20
l0
iterations
0
10000
20000
30000
40000



The sampled parameters are plotted along the number of iterations. The algorithm seems to mix relatively well for all values of σ for all parameters except λ_1. Here we can observe that the algorithm moves more slowly between the different values of λ_1 when $\sigma = 1$, i.e. quite low.

7.

We now implement the MCMC using block proposals for:

- $(t_1, \lambda_0, \lambda_1)$ keeping β constant
- $(\beta, \lambda_0, \lambda_1)$ keeping t_1 constant

We alternate between these two block updates for every iteration, and we have an implementation of the algorithm below.

In the first block, we sample $\tilde{t_1}$ from a normal distribution, as in section 4), and λ_0 and λ_1 are sampled from their full conditionals conditioned on the proposed $\tilde{t}_t$. We note that sampling λ_0 and λ_1 from their joint conditional $f(\lambda_1, \lambda_0 \mid t_1, \beta, x)$ is equivalent to sampling them from their marginal conditionals, as the joint conditional is the product of the marginal conditionals:

$$
\begin{aligned}
f(\lambda_0, \lambda_1 \mid x, \beta, t_1) &= \frac{\lambda_0^{y_0+1}\lambda_1^{y_1+1}e^{-\lambda_0(t_1-t_0+\frac{1}{\beta})}e^{-\lambda_1(t_2-t_1+\frac{1}{\beta})}}{\int_0^\infty\int_0^\infty \lambda_0^{y_0+1}\lambda_1^{y_1+1}e^{-\lambda_0(t_1-t_0+\frac{1}{\beta})}e^{-\lambda_1(t_2-t_1+\frac{1}{\beta})}\mathrm{d}\lambda_0\mathrm{d}\lambda_1} \\
&\propto \lambda_0^{y_0+1}\lambda_1^{y_1+1}e^{-\lambda_0(t_1-t_0+\frac{1}{\beta})}e^{-\lambda_1(t_2-t_1+\frac{1}{\beta})} \\
&\propto f(\lambda_0 \mid \lambda_1, x, \beta, t_1) \cdot f(\lambda_1 \mid \lambda_0, \beta, t_1, x).
\end{aligned}
$$

This gives us the following acceptance probability α_1:

$$\alpha_1 = \min(1, \frac{\pi(\tilde{t_1}, \tilde{\lambda_0}, \tilde{\lambda_1} \mid \beta)}{\pi(t_1, \lambda_0, \lambda_1 \mid \beta)} \times \frac{Q(t_1, \lambda_0, \lambda_1 \mid \tilde{t_1}, \tilde{\lambda_0}, \tilde{\lambda_1}, \beta)}{Q(\tilde{t_1}, \tilde{\lambda_0}, \tilde{\lambda_1} \mid t_1, \lambda_0, \lambda_1, \beta)}).$$

We look more closely at these expressions:

$$Q(\tilde{t_1}, \tilde{\lambda_0}, \tilde{\lambda_1} \mid t_1, \lambda_0, \lambda_1, \beta) = Q(\tilde{\lambda_0}, \tilde{\lambda_1} \mid \tilde{t_1}, \beta) Q(\tilde{t_1} \mid t_1, \beta)$$

$$= f(\tilde{\lambda_0} \mid x, \tilde{t_1}, \beta) f(\tilde{\lambda_1} \mid x, \tilde{t_1}, \beta) \mathrm{N}(t_1, \sigma)$$

$$= \frac{\tilde{\lambda_0}^{\tilde{y_0}+1} (\tilde{t_1} - t_0 + \frac{1}{\beta})^{\tilde{y_0}+2} e^{-\tilde{\lambda_0}(\tilde{t_1}-t_0+\frac{1}{\beta})}}{\Gamma(\tilde{y_0}+2)} \frac{\tilde{\lambda_1}^{\tilde{y_1}+1} (t_1 - \tilde{t_1} + \frac{1}{\beta})^{\tilde{y_1}+2} e^{-\tilde{\lambda_1}(t_2-\tilde{t_1}+\frac{1}{\beta})}}{\Gamma(\tilde{y_1}+2)} \cdot \mathrm{N}(t_1, \sigma)$$

$$\pi(\tilde{t_1}, \tilde{\lambda_0}, \tilde{\lambda_1} \mid \beta) \propto \pi(\tilde{t_1}, \tilde{\lambda_0}, \tilde{\lambda_1}, \beta)$$

$$\propto \tilde{\lambda}_0^{\tilde{y}_0+1} \cdot \tilde{\lambda}_1^{\tilde{y}_1+1} \cdot e^{-\tilde{\lambda}_0(\tilde{t}_1-t_0+\frac{1}{\beta})} \cdot e^{-\tilde{\lambda}_1(t_2-\tilde{t}_1+\frac{1}{\beta})}.$$

$\pi(t_1, \lambda_0, \lambda_1 \mid \beta)$ and $Q(t_1, \lambda_0, \lambda_1 \mid \tilde{t}_1, \tilde{\lambda}_0, \tilde{\lambda}_1, \beta)$ will be on the same form. When inserting this into the expression for α we see that many of the factors cancel, and we are left with the expression

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

In the second block, we keep t_1 constant while sampling $\tilde{\beta}$ from a normal distribution centered at the last value of β with variance σ_2^2, and then sample $\tilde{\lambda}_0$ and $\tilde{\lambda}_1$ from their full conditionals conditioned on $\tilde{\beta}$. With this approach, we get the expression for the acceptance probability α_2:

$$\alpha_2 = \min\left(1, \frac{\pi(\tilde{\lambda}_0, \tilde{\lambda}_1, \tilde{\beta} \mid t_1)}{\pi(\lambda_0, \lambda_1, \beta \mid t_1)} \times \frac{Q(\lambda_0, \lambda_1, \beta \mid \tilde{\lambda}_0, \tilde{\lambda}_1, \tilde{\beta}, t_1)}{Q(\tilde{\lambda}_0, \tilde{\lambda}_1, \tilde{\beta} \mid \lambda_0, \lambda_1, \beta, t_1)}\right),$$

where, similarily to the case in the first block

$$Q(\tilde{\lambda}_0, \tilde{\lambda}_1, \tilde{\beta} \mid \lambda_0, \lambda_1, \beta, t_1) = Q(\tilde{\lambda}_0, \tilde{\lambda}_1 \mid \tilde{\beta}, t_1) \cdot \mathrm{N}(\beta, \sigma_2),$$

$$Q(\tilde{\lambda}_0, \tilde{\lambda}_1 \mid \tilde{\beta}, t_1) = \frac{\tilde{\lambda_0}^{y_0+1} (t_1 - t_0 + \frac{1}{\tilde{\beta}})^{y_0+2} e^{-\tilde{\lambda}_0(t_1-t_0+\frac{1}{\tilde{\beta}})}}{\Gamma(y_0+2)} \times \frac{\tilde{\lambda}_1^{y_1+1} (t_2 - t_1 + \frac{1}{\tilde{\beta}})^{y_1+2} e^{-\tilde{\lambda}_1(t_2-t_1+\frac{1}{\tilde{\beta}})}}{\Gamma(y_1+2)}$$

and

$$\pi(\tilde{\lambda}_0, \tilde{\lambda}_1, \tilde{\beta} \mid t_1) \propto \frac{1}{\tilde{\beta}^5} \cdot \tilde{\lambda}_0^{y_0+1} \cdot \tilde{\lambda}_1^{y_1+1} \cdot e^{-\tilde{\lambda}_0(t_1-t_0+\frac{1}{\tilde{\beta}})} \cdot e^{-\tilde{\lambda}_1(t_2-t_1+\frac{1}{\tilde{\beta}})}.$$

Inserting these, and the corresponding expressions for $\pi(\lambda_0, \lambda_1, \beta \mid t_1)$ and $Q(\lambda_0, \lambda_1, \beta \mid \tilde{\lambda}_0, \tilde{\lambda}_1, \tilde{\beta}, t_1)$ into our expression for α_2 we get

$$\alpha_2 = \min\left(1, \frac{\beta^5}{\tilde{\beta}^5} \cdot \frac{(t_1 - t_0 + \frac{1}{\beta})^{y_0+2} (t_1 - t_1 + \frac{1}{\beta})^{y_1+2}}{(t_1 - t_0 + \frac{1}{\tilde{\beta}})^{y_0+2} (t_2 - t_1 + \frac{1}{\tilde{\beta}})^{y_1+2}}\right).$$

```
block.1 <- function(theta,i,sigma,t0,t2){
  # block 1 update
  th <- theta

  #help vars:
  t1 <- th["t1",i-1]
  y1 <- th["y1",i-1]
```

```
y0 <- th["y0",i-1]
beta <- th["beta",i-1]

# keep beta constant
th["beta",i] <- th["beta",i-1]

# sample new t1 from normal distribution
t1.n <- rnorm(1,t1,sigma)
if(t1.n > t2 | t1.n < t0){
  t1.n <- t1
}

#find corresponding y0.n and y1.n
y0.n <- max(which(coal$date <= t1.n)) - 1
y1.n <- length(coal$date) - y0.n - 2

# sample lambda0 from Gamma distirbution with new t1
l0.n <- rgamma(1,shape = y0.n + 2, scale = 1/(t1.n - t0 + 1/beta))

# sample lambda1 from Gamma distribution with new t1
l1.n <- rgamma(1,shape = y1.n + 2, scale = 1/(t2 - t1.n + 1/beta))

# find alpha
g.fact <- sum(log(1:y0.n + 1)) + sum(log(1:y1.n + 1)) -
  sum(log(1:y0+1)) - sum(log(1:y1+1))

t.fact <- (y0 + 2)*log(t1 - t0 + 1/beta) +
  (y1 + 2)*log(t2 - t1 + 1/beta) -
  (y0.n + 2)*log(t1.n - t0 + 1/beta) -
  (y1.n + 2)*log(t2 - t1.n + 1/beta)

alpha <- exp(t.fact + g.fact)
alpha <- min(1,alpha)


# accept or reject
u <- runif(1)
if(u < alpha){
  # accept
  th["t1",i] <- t1.n
  th["l0",i] <- l0.n
  th["l1",i] <- l1.n
  th["y0",i] <- y0.n
  th["y1",i] <- y1.n
  th["accept",i] <- 1
}
else{
  # reject
  th["t1",i] <- th["t1",i-1]
  th["l0",i] <- th["l0",i-1]
  th["l1",i] <- th["l1",i-1]
  th["y0",i] <- th["y0",i-1]
  th["y1",i] <- th["y1",i-1]
```

```
    th["accept",i] <- 0
  }
  return(th)
}

block.2 <- function(theta,i,sigma, t0, t2){
  # block 2 update
  th <- theta

  #help vars:
  t1 <- th["t1",i-1]
  y1 <- th["y1",i-1]
  y0 <- th["y0",i-1]
  beta <- th["beta",i-1]

  # keep t1 constant
  th["t1",i] <- th["t1",i-1]
  th["y0",i] <- th["y0",i-1]
  th["y1",i] <- th["y1",i-1]

  # sample beta from normal distribution
  beta.n <- rnorm(1,beta,sigma)
  if(beta.n <= 0){
    beta.n <- beta  # if beta out of bounds, reject
  }

  # sample l0 from Gamma distribution with new beta
  l0.n <- rgamma(1,shape = y0 + 2, scale = 1/(t1 - t0 + 1/beta.n))

  # sample l1 from Gamma distribution with new beta
  l1.n <- rgamma(1,shape = y1 + 2, scale = 1/(t2 - t1 + 1/beta.n))

  alpha.lg <- 5*log(beta) - 5*log(beta.n) +
    (y0 + 2)*log(t1 - t0 + 1/beta) +
    (y1 + 2)*log(t2 - t1 + 1/beta) -
    (y0 + 2)*log(t1 - t0 + 1/beta.n) -
    (y1 + 2)*log(t2 - t1 + 1/beta.n)
  alpha <- exp(alpha.lg)
  alpha <- min(1, alpha)

  #accept or reject
  u <- runif(1)
  if(u < alpha){
    # accept
    th["beta",i] <- beta.n
    th["l0",i] <- l0.n
    th["l1",i] <- l1.n
    th["accept",i] <- 1
  }
  else{
    # reject
    th["beta",i] <- th["beta",i-1]
    th["l0",i] <- th["l0",i-1]
```

```
    th["l1",i] <- th["l1",i-1]
    th["accept",i] <- 0
  }
  return(th)
}

MH <- function(n,t1.0,l0.0,l1.0,beta.0, obs, sigma1, sigma2){
  # initializing parameters with prior estimates

  theta <- matrix(0L, nrow=7, ncol=n)
  rownames(theta) <- c("t1", "l0", "l1", "beta", "y0", "y1", "accept")

  t0 <- coal$date[1]
  t2 <- tail(coal$date,1)

  theta["t1",1] <- t1.0
  # disasters before t1, subtracting the first element which is not a disaster
  theta["y0",1] <- max(which(coal$date < theta["t1",1])) - 1
  # disasters at and after t1, subtracting start time and end time element
  theta["y1",1] <- length(coal$date) - theta["y0",1] - 2

  theta["l0",1] <- l0.0
  theta["l1",1] <- l1.0
  theta["beta",1] <- beta.0

  for(i in 2:n){
    # alternating block updates
    if(i%%2 == 0){
      # block 1 update
      theta <- block.1(theta, i, sigma1, t0, t2)
    }
    else{
      # block 2 update
      theta <- block.2(theta, i, sigma2, t0, t2)
    }
  }
  return(theta)
}
```

In the second block, we update β by sampling from a normal distibution with mean β_{i-1} and variance σ_2^2. We show the results for $\sigma_2^2 = [0.1, 1, 10]$ while keeping the variance in the sampling of t_1, $\sigma_1^2 = 10$, as we found this to be a good value in section A4).

```
# var2 = 0.1
blc.01 <- MH(n=40000, t1.0=1940,l0.0 = 3,l1.0 = 1,
             beta.0  = 1, obs = coal, sigma1 = 10, sigma2 = 0.1)
blc.01.df <- as.data.frame(t(blc.01))

# var2 = 1
blc.1 <- MH(n=40000, t1.0=1940,l0.0 = 3,l1.0 = 1,
            beta.0  = 1, obs = coal, sigma1 = 10, sigma2 = 1)
blc.1.df <- as.data.frame(t(blc.1))
```

```r
# var2  = 10
blc.10 <- MH(n=40000, t1.0=1940,l0.0 = 3,l1.0 = 1,
             beta.0  = 1, obs = coal, sigma1 = 10, sigma2 = 10)
blc.10.df <- as.data.frame(t(blc.10))

acc.01 <- mean(blc.01.df$accept); cat("acceptante 0.1: ", acc.01, "\n")
```

```
## acceptante 0.1:  0.561625
```

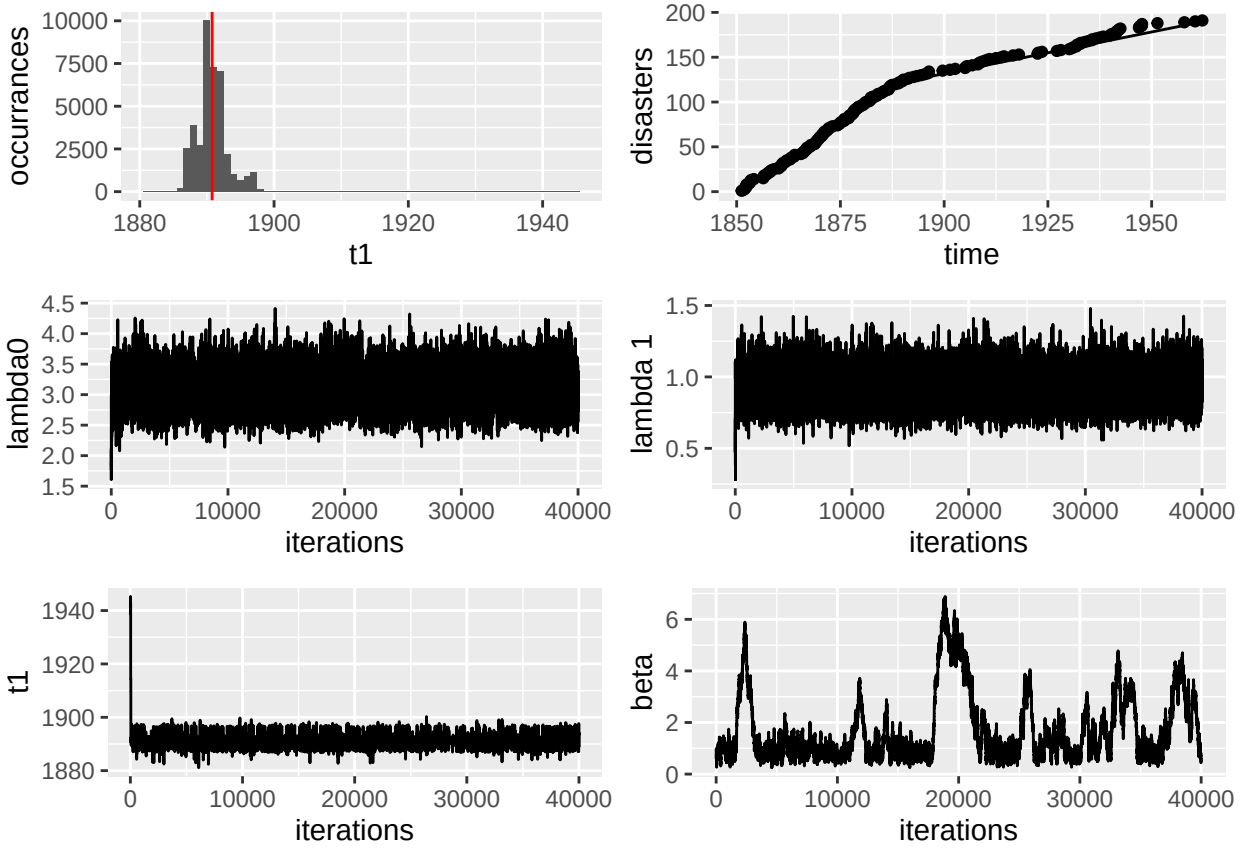
```r
acc.1 <- mean(blc.1.df$accept); cat("acceptance rate 1: ", acc.1, "\n")
```

```
## acceptance rate 1:  0.418375
```

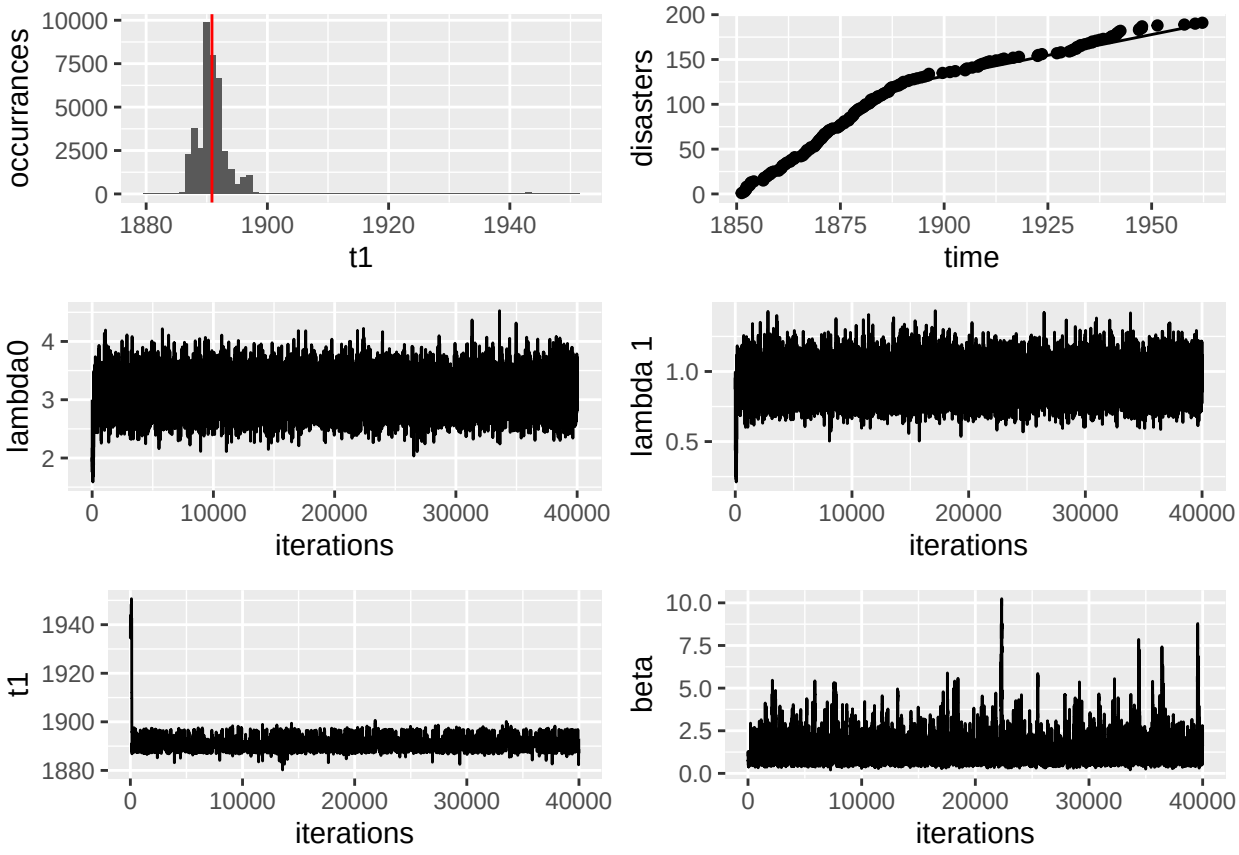
```r
acc.10 <- mean(blc.10.df$accept); cat("acceptance rate 10: ", acc.10, "\n")
```

```
## acceptance rate 10:  0.360225
```

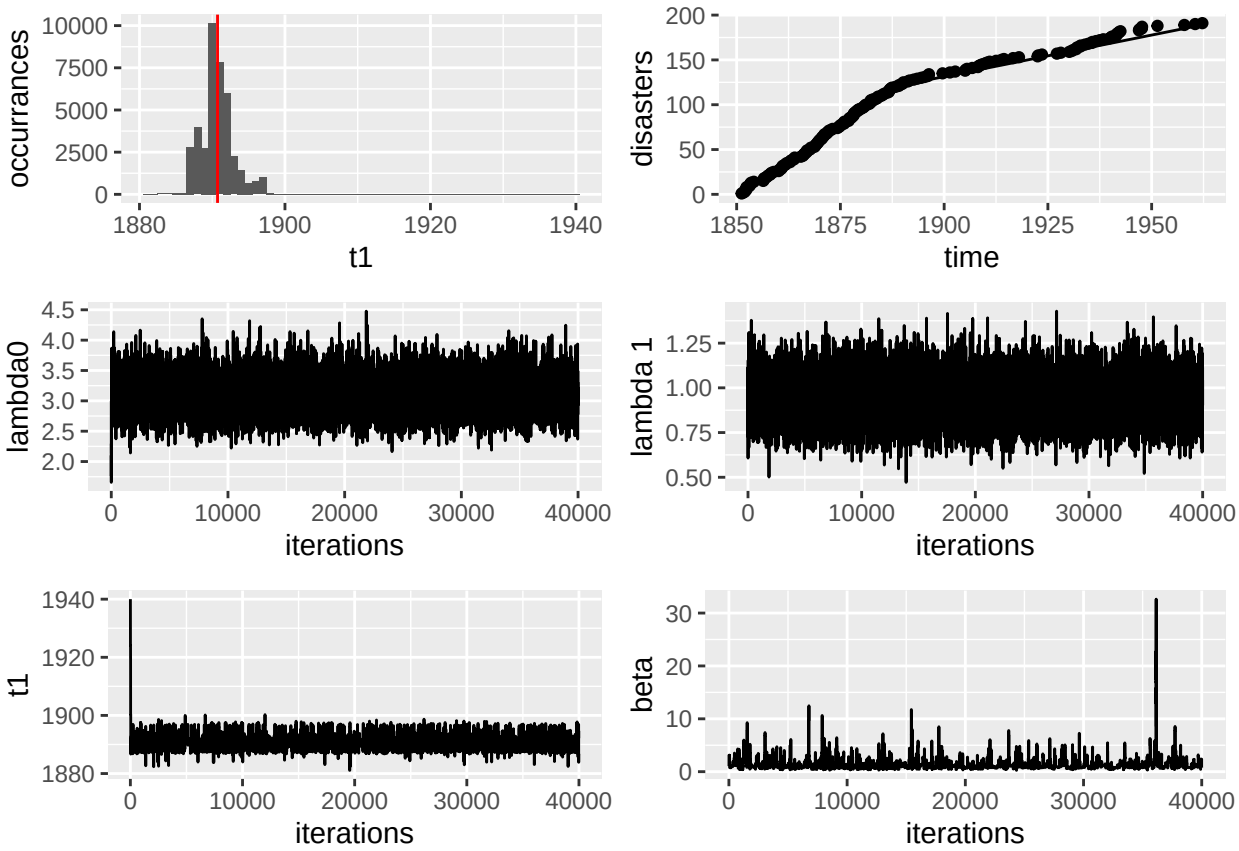
```r
# plotting results for the different sigmas:

relevant.plots(blc.01.df,coal,1000)
```



```
relevant.plots(blc.1.df,coal,1000)
```




```
relevant.plots(blc.10.df,coal,1000)
```



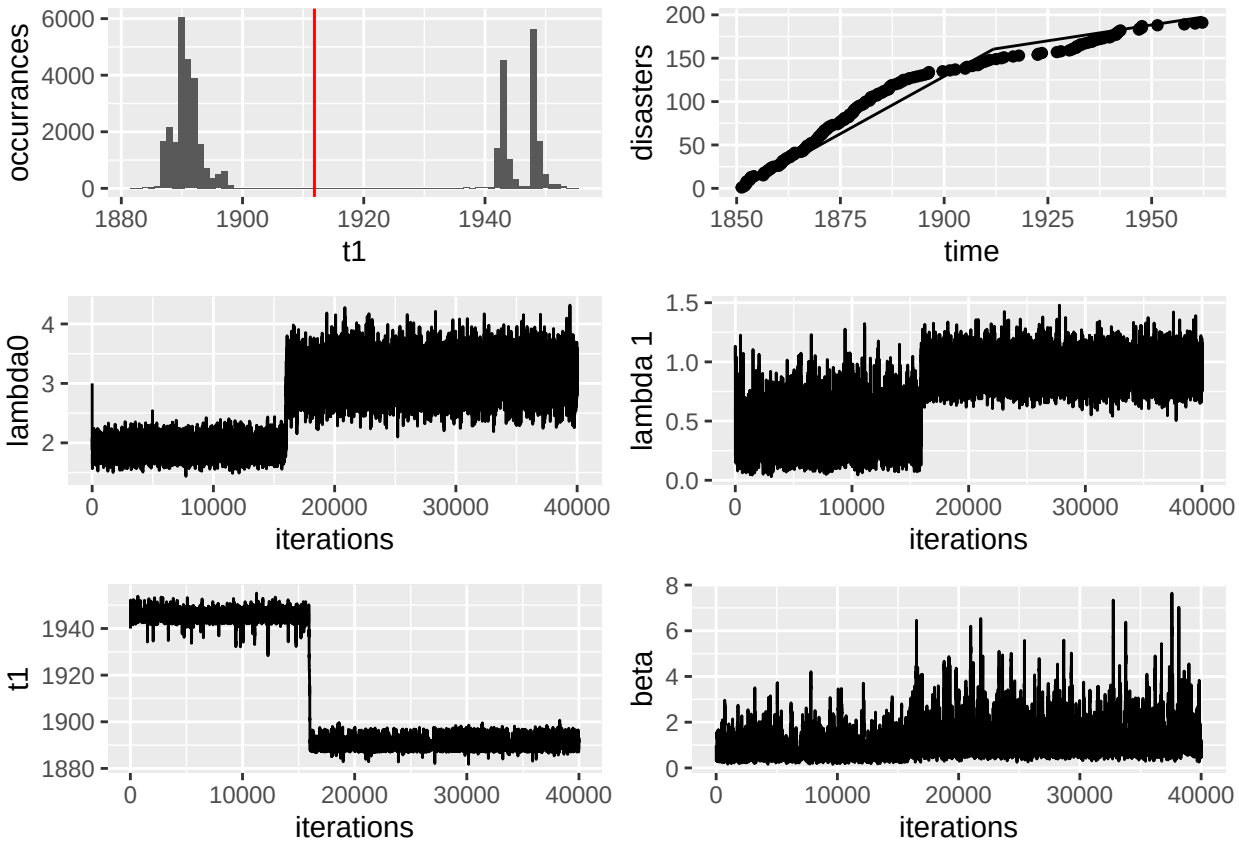
Above the simulated Markov Chain is plotted for $\sigma_2^2 = 0.1$ (top), $\sigma_2^2 = 1$ (middle) and $\sigma_2^2 = 10$ (bottom). We observe that while all values of σ_2^2 seem to produce reasonable results, $\sigma_2^2 = 0.1$ makes the algoritm explore the β values more slowly. For $\sigma_2^2 = 0.1$, the acceptance rate is also slightly higher than optimal, we usually want an acceptance rate between 20% and 50% for random-walk proposals. None of the values for σ_2 seemed to produce very long burn-in periods for either of the parameters.

To be able to compare the performance of the block-update algoritm versus the single-site algorthm, we run the block algorithm with $\sigma_1 = 5$, for which the single-site algorithm produced a quite long burn-in period.

```
# var2  = 10
blc.1.1 <- MH(n=40000, t1.0=1940,l0.0 = 3,l1.0 = 1,
              beta.0  = 1, obs = coal, sigma1 = 5, sigma2 = 1)
blc.1.1.df <- as.data.frame(t(blc.1.1))

acc.1.1 <- mean(blc.1.1.df$accept); cat("acceptance rate sigma1 = 1: ", acc.1.1, "\n")
```

```
## acceptance rate sigma1 = 1:  0.458175
```

```
relevant.plots(blc.1.1.df,coal,1000)
```



From the plots of the samples of t_1 and λ_0 we observe that there is a significantly longer burn-in period compared to the corresponding realization in the single-site algoritm. For the block-update, a burn-in period of 15000 iterations seems to be sufficient for t_1, compared to about 2500 iterations for the single-site algorithm. This indicates that the single-site algorithm converges faster for less optimal values of the tuning parameter σ_1 than the block-update algorithm does.

We want to use the simulation results to estimate the marginal posterior distributions $f(t_1 \mid x), f(\lambda_0 \mid x), f(\lambda_1 \mid x)$ and $f(\beta \mid x)$. For $f(\lambda_0 \mid x)$ and $f(\lambda_1 \mid x)$ we use the simulation results from the block-update algorithm with $\sigma_1 = 10$ and $\sigma_2 = 1$. For $f(t_1 \mid x)$ and $f(\beta \mid x)$ we use the single-site algorithm with $\sigma_1 = 10$, as this seemed to provide better properties for these parameters. To find the simulated marginal distributions we have used the full conditionals for the parameters together with the sample mean of the remaining parameters from the simulation. The results are seen below:

```
# mean values:
t1.mean <- mean(th10.df[1000:length(th10.df$t1),]$t1)
cat("Expected value of t1: ",t1.mean, "\n")
```

```
## Expected value of t1:  1890.784
```

```
y0.mean <- mean(th10.df[1000:length(th10.df$t1),]$y0)
y1.mean <- mean(th10.df[1000:length(th10.df$t1),]$y1)
beta.mean <- mean(th10.df[1000:length(th10.df$t1),]$beta)
cat("Expected value of beta: ", beta.mean, "\n")
```

```
## Expected value of beta:  1.674551
```

```
l0.mean <- mean(blc.1.df[1000:length(blc.1.df$t1),]$l0)
cat("Expected value of lambda 0: ", l0.mean, "\n")
```

```
## Expected value of lambda 0:  3.099295
```

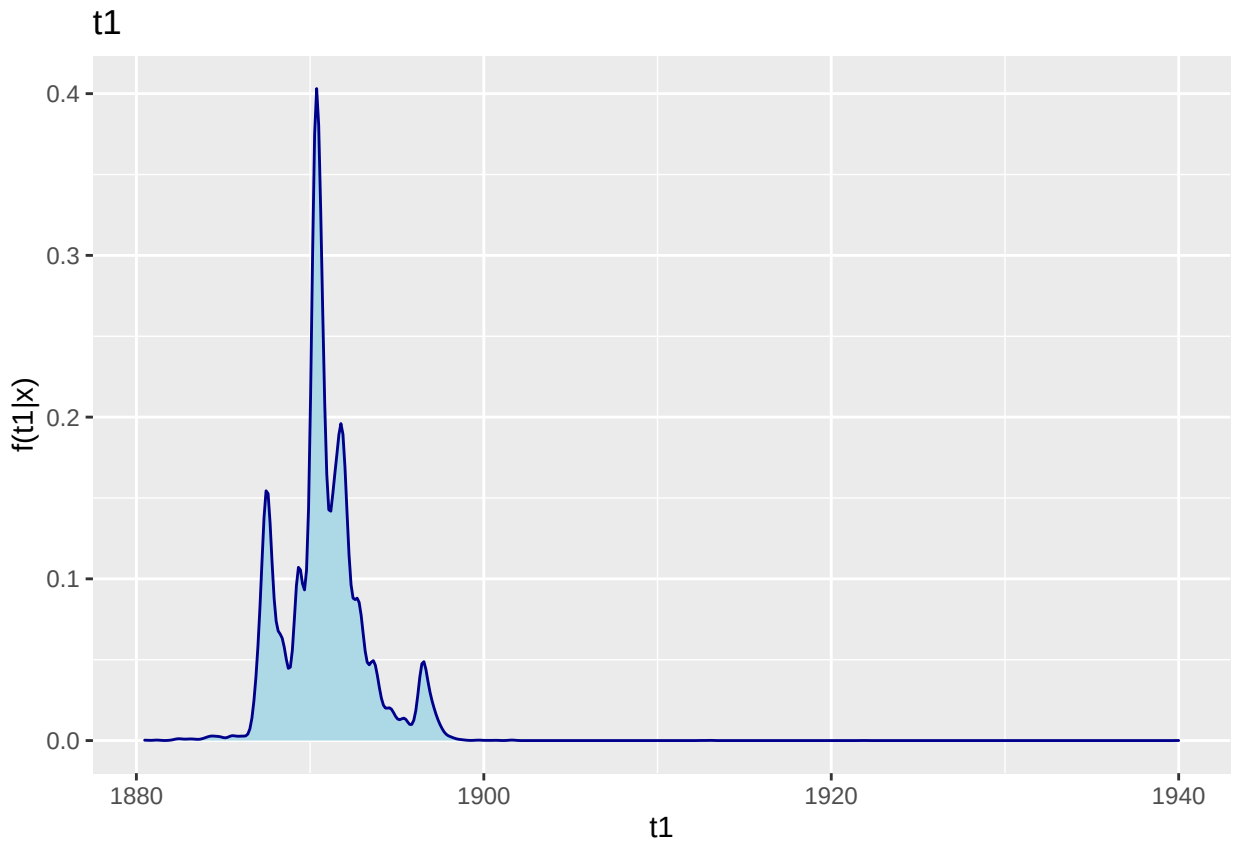
```
l1.mean <- mean(blc.1.df[1000:length(blc.1.df$t1),]$l1)
cat("Expected value of lambda 1: ", l1.mean, "\n")
```

```
## Expected value of lambda 1:  0.9287363
```

```
# covariance of lambda 1 and lambda 0:
cov.l0.l1 <- cov(blc.1.df[1000:length(blc.1.df$t1),]$l0,blc.1.df[1000:length(blc.1.df$t1),]$l1)
cat("Covariance of lambda 0 and lambda 1: ",cov.l0.l1)
```

```
## Covariance of lambda 0 and lambda 1:  0.002303135
```

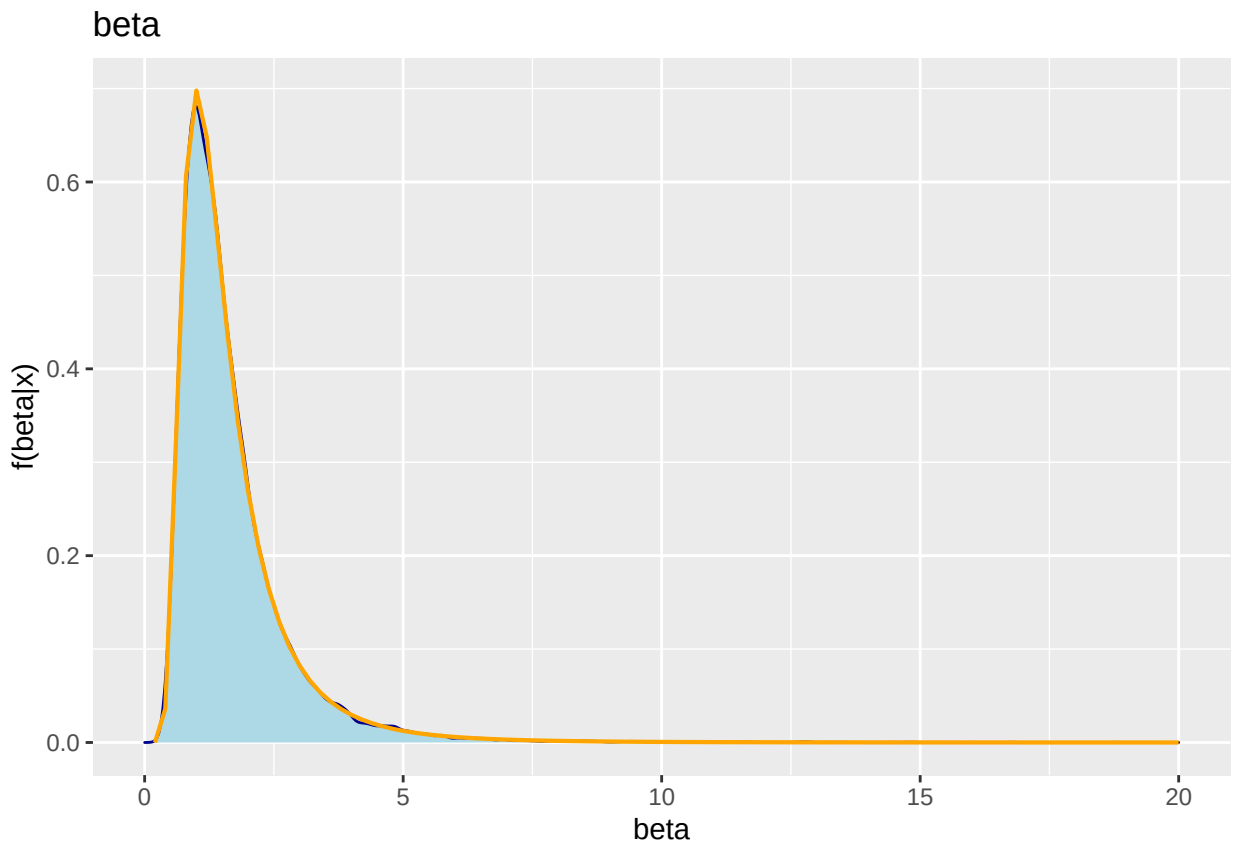
```
f.t1 <- ggplot(th10.df,aes(x=t1)) + geom_density(color="darkblue", fill="lightblue") +
  ggtitle(label="t1") + labs(y="f(t1|x)",x="t1")
f.t1
```



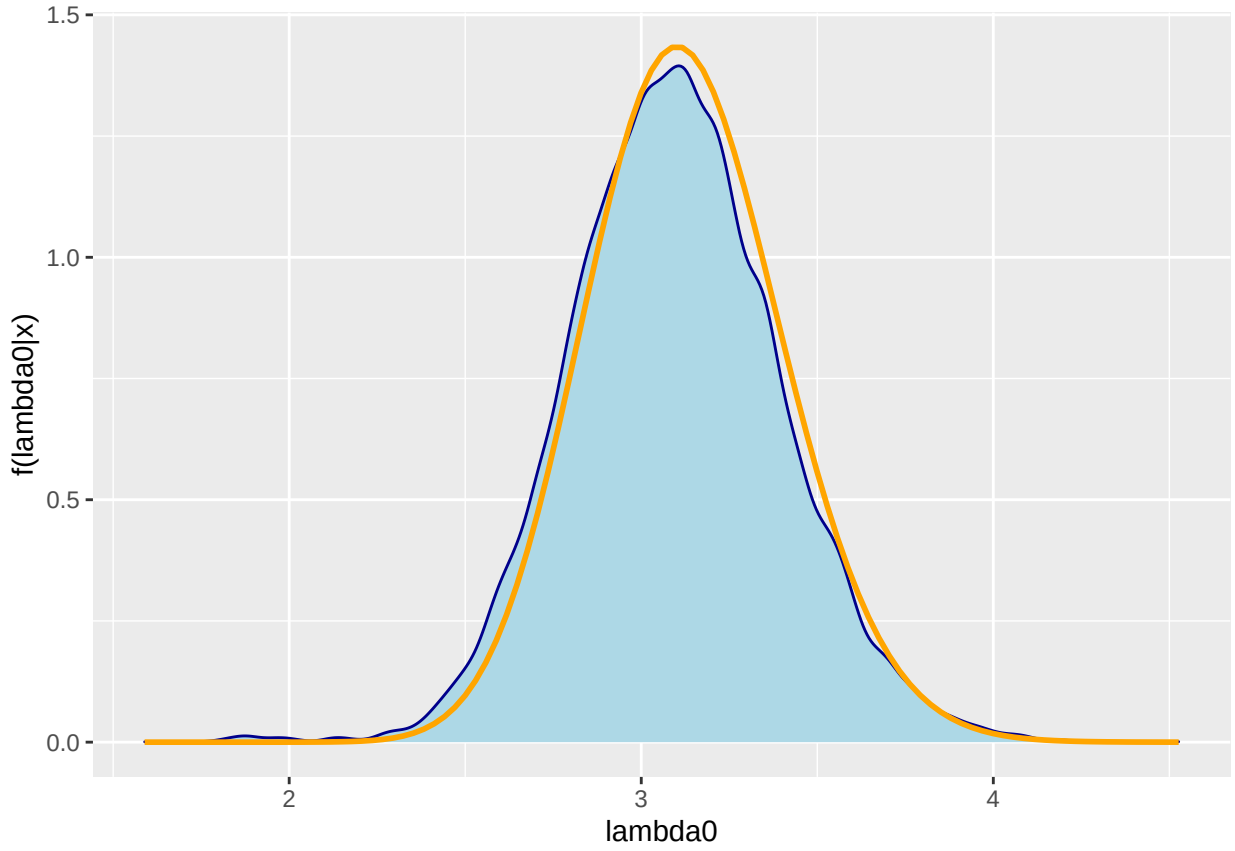

```
f.beta <- ggplot(th10.df,aes(x=beta)) + geom_density(color="darkblue", fill="lightblue")
f.beta <- f.beta + stat_function(fun=dinvgamma, args=list(shape=4, scale=1/(l0.mean + l1.mean + 1)),
                                 color = "orange", size = 0.75) +
  xlim(0,20) + ggtitle(label="beta") + labs(x="beta",y="f(beta|x)")
f.beta
```

```
## Warning: Removed 8 rows containing non-finite values (stat_density).
```

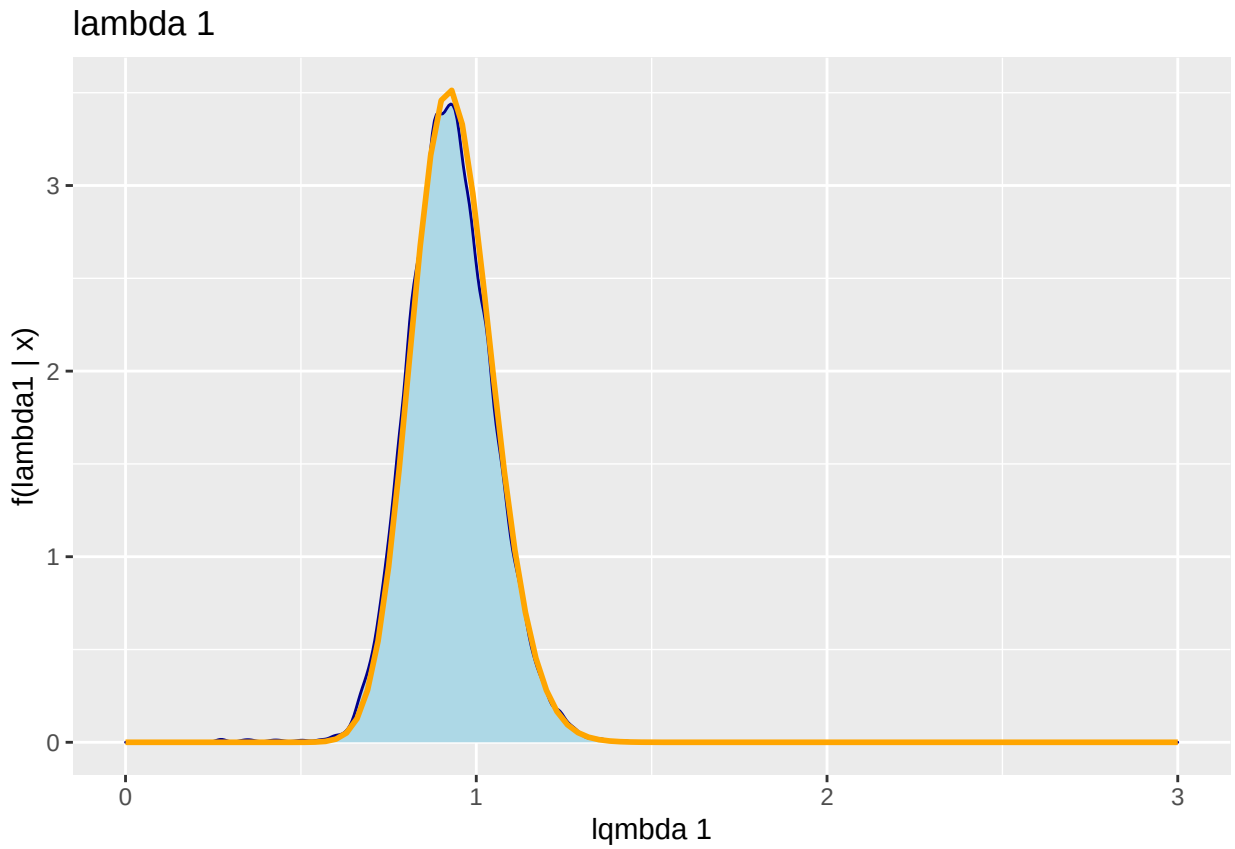
```
## Warning: Removed 1 row(s) containing missing values (geom_path).
```




```
f.l0 <- ggplot(blc.1.df,aes(x=l0)) + geom_density(color="darkblue", fill="lightblue")
f.l0 <- f.l0 + stat_function(fun=dgamma,
                             args=list(shape=y0.mean + 2,
                                       scale=1/(t1.mean - coal$date[1] + 1/beta.mean)),
                             color="orange",size=1) + labs(x="lambda0",y="f(lambda0|x)")
f.l0
```




```
f.l1 <- ggplot(blc.1.df,aes(x=l1)) + geom_density(color="darkblue", fill="lightblue")
f.l1 <- f.l1 + stat_function(fun=dgamma,
                             args=list(shape=y1.mean + 2,
                                       scale=1/(tail(coal$date,1) - t1.mean + 1/beta.mean)),
                             color="orange",size=1) + xlim(0,3) + ggtitle(label="lambda 1") +
  labs(x="lqmbda 1",y="f(lambda1 | x)")
f.l1
```

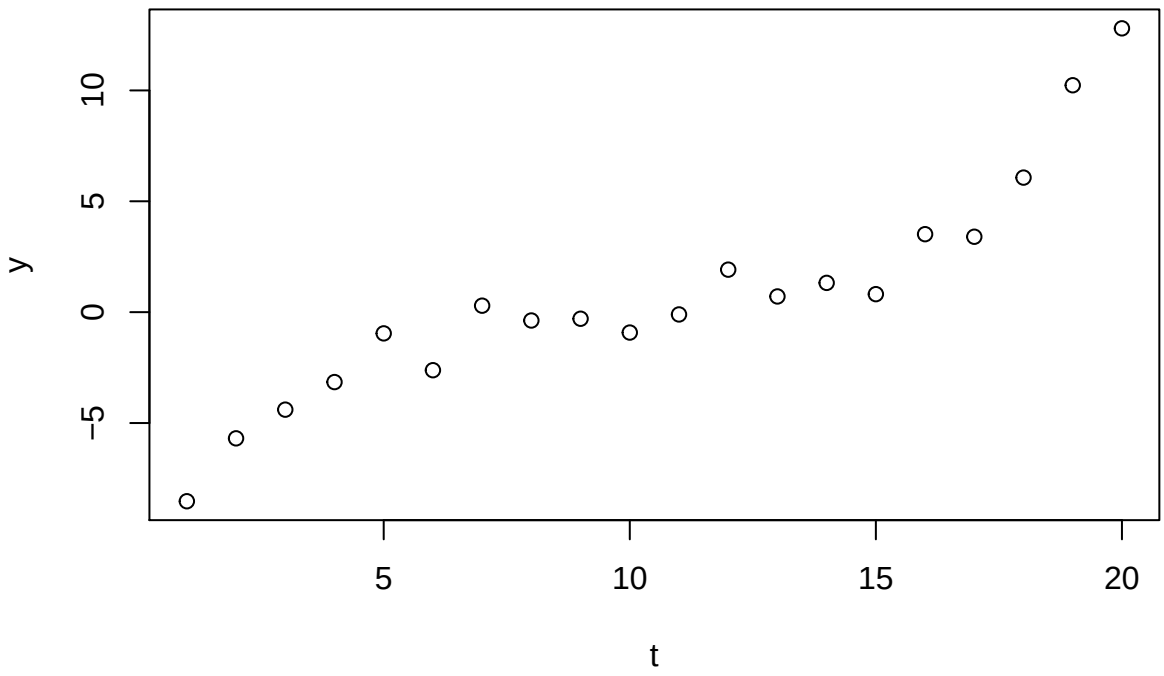


Here, we see the sample density (blue) plotted together with the marginal conditionals (orange) where the mean of the remaining parameters are used as model parameters. The execption is $f(t_1 \mid x)$ where we do not have a simple expression for the density, as the conditional marginal of t_1 is a function of the non-continous $y_0(t_1)$ and $y_1(t_1)$. We observe that the gamma distribution fit well with $f(\lambda_0 \mid x)$ and $f(\lambda_1 \mid x)$ and that the inverse-gamma distribution is a good fit for $f(\beta \mid x)$.

The covariance of λ_0 and λ_1 is found to be 0.002, which is low and indicates that λ_1 and λ_0 are independent. This is in line with our previous reasoning in section 7), where we argued that λ_1 and λ_0 are independent because their joint conditional is the product of their marginal conditionals.

Problem B

```
# Read and plot data
gaussiandata = read.delim("gaussiandata.txt")
y = gaussiandata[,1]
t = seq(from=1,to=length(y),by=1)
plot(t,y)
```



1.

We consider the problem of smoothing the time series that is plotted above. We assume that given the vector of linear predictors $\eta = (\eta_1, \dots, \eta_T)$, where in this case $T = 20$, the observations y_t are independent and distributed according to

$$y_t \mid \eta_t \sim \mathcal{N}(\eta_t, 1) \quad ,$$

for $t = 1, \dots, T$. The linear predictor for time t is $\eta_t = f_t$, where f_t is the smooth effect for time t. For the prior distribution of $\mathbf{f} = (f_1, \dots, f_T)$ we have a second order random walk model, that is,

$$\pi(\mathbf{f} \mid \theta) \propto \theta^{(T-2)/2} \exp\Big\{ -\frac{\theta}{2} \sum_{t=3}^{T} (f_t - 2f_{t-1} + f_{t-2}^2) \Big\} = \mathcal{N}(\mathbf{0}, \mathbf{Q}(\theta)^{-1}) \quad ,$$

where $\mathbf{Q}$ is the precision matrix and θ is the precision parameter that controls the smoothness of $\mathbf{f}$. We assume that the $Gamma(1, 1)$-distribution is the prior for θ.

The model described here can be written as the hierarchichal model:

$$\begin{aligned} \mathbf{y} \mid \mathbf{f} &\sim \prod_{t=1}^{T} P(y_t \mid \eta_t) \\ \mathbf{f} \mid \theta &\sim \pi(\mathbf{f} \mid \theta) = \mathcal{N}(\mathbf{0}, \mathbf{Q}(\theta)^{-1}) \\ \theta &\sim Gamma(1, 1) \end{aligned}$$

Here, the first line is the likelihood of the response $\mathbf{y} = (y_1, \dots, y_T)$, the second line gives the prior distribution of the latent field, and the third line gives the prior distribution of the hyperparameter θ. Since our model has this particular structure, it is a latent Gaussian model. INLA can be used to estimate the parameters

because we have a latent gaussian model where each data point y_t depends only on the one element f_t in the latent field, the dimension of the hyperparameter is one and the precision matrix $\mathbf{Q}(\theta)$ of the latent field is sparse.

2.

Here, we implement a block Gibbs sampling algorithm for $f(\eta, \theta \mid \mathbf{y})$, where we propose a new value for θ from the full conditional $\pi(\theta \mid \eta, \mathbf{y})$ and a new value for η from the full conditional $\pi(\eta \mid \theta, \mathbf{y})$. Thus, we need to find these distributions. We start with the posterior

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

Then we find the full conditional for θ to be

$$\begin{aligned} \pi(\theta \mid \mathbf{y}, \eta) &\propto \theta^{T/2-1} \exp\left\{ -\theta \left(1 + \frac{1}{2} \sum_{t=3}^{T} (\eta_t - 2\eta_{t-1} + \eta_{t-2})^2 \right) \right\} \\ &\propto Gamma\left(\frac{T}{2}, 1 + \frac{1}{2} \sum_{t=3}^{T} (\eta_t - 2\eta_{t-1} + \eta_{t-2})^2 \right) \end{aligned}.$$

The full conditional for η is

$$\begin{aligned} \pi(\eta \mid \theta, \mathbf{y}) &\propto \exp\left\{ -\frac{\theta}{2} \sum_{t=3}^{T} (\eta_t - 2\eta_{t-1} + \eta_{t-2})^2 - \frac{1}{2} \sum_{t=1}^{T} (y_t - \eta_t)^2 \right\} \\ &= \exp\left\{ -\frac{1}{2} \left(\eta^T \mathbf{Q} \eta + (\mathbf{y} - \eta)^T (\mathbf{y} - \eta) \right) \right\} \\ &= \exp\left\{ -\frac{1}{2} \eta^T (\mathbf{Q} + \mathbf{I}) \eta + \mathbf{y}^T \eta \right\} \end{aligned}.$$

Here, $\mathbf{Q}(\theta) = \theta \mathbf{L}\mathbf{L}^T$ is the precision matrix, where $\mathbf{L}$ is the $T \times (T-2)$ matrix

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

By looking at the last line in the above expression for $\pi(\eta \mid \theta, \mathbf{y})$, we recognize that the canonical parametrization is $\mathcal{N}_C(\mathbf{y}, \mathbf{Q} + \mathbf{I})$, and find that $\pi(\eta \mid \theta, \mathbf{y}) \propto \mathcal{N}((\mathbf{Q} + \mathbf{I})^{-1}\mathbf{y}, (\mathbf{Q} + \mathbf{I})^{-1})$. In the algorithm we sample the new proposals for the parameters from these two distributions that we have found for the full conditionals. We always use the last updated parameters.

```
library(Matrix)
```

```
##
## Attaching package: 'Matrix'
```

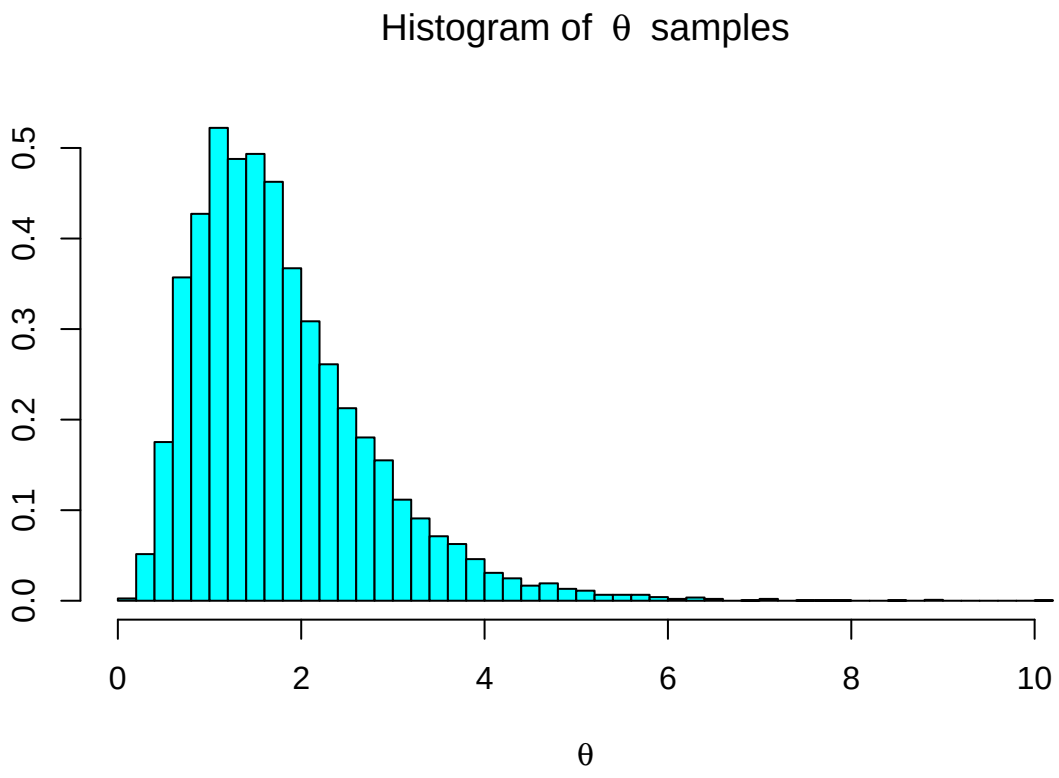
```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```r
library(mvtnorm)
library(MASS)
```

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

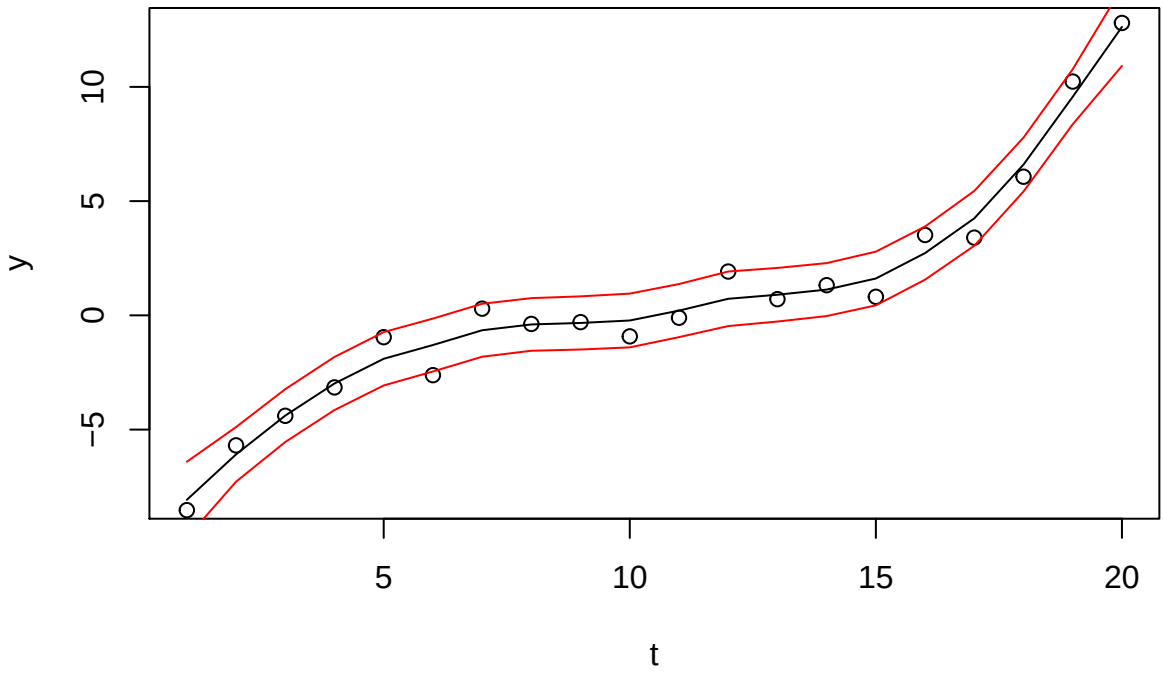
```r
# Function to make the precision matrix
make.Q = function(T, theta) {
  # Make the matrix L as described in the text
  L = diag(T)
  d1 = rep(-2,T-1)
  d2 = rep(1, T-2)
  L[row(L)-col(L)==1] = d1
  L[row(L)-col(L)==2] = d2
  L = L[,-c(T-1,T)]
  # Compute Q(theta)
  Q = theta * L %*% t(L)
  return(Q)
}

set.seed(0)
# Function for block Gibbs sampling
# n is the number of samples including the inital value
sample.Gibbs = function(n, theta.init, f.init, y) {
  T = length(f.init)
  # Make vector and matrix for storing the samples
  theta.vec = rep(0,n)
  f.matrix = matrix(1:T*n, nrow = T, ncol = n)
  # Initialize
  theta.vec[1] = theta.init
  f.matrix[, 1] = f.init
  # Iterations
  for(i in 2:n) {
    # Sample theta
    summ = 0
    for(t in 3:T) {
      summ = summ + (f.matrix[t, i-1] - 2*f.matrix[t-1, i-1] + f.matrix[t-2, i-1])^2
    }
    theta.vec[i] = rgamma(1, shape = T/2, rate = 1 + 0.5*summ)
    # Sample f
    Q = make.Q(T, theta.vec[i])          # Use the last updated theta
    f.mean = solve(Q+diag(T)) %*% y
    f.sigma = solve(Q+diag(T))
    f.matrix[, i] = rmvnorm(1, f.mean, f.sigma)
  }
  return(rbind(f.matrix, theta.vec))    # Return concatenated matrix with f and theta samples
}

# Set values
```

```r
n = 10000
T = length(y)
theta.init = 1
f.init = rep(2,T)
# Sample
result = sample.Gibbs(n, theta.init, f.init, y)
# Extracting the theta samples, excluding the first 100 values
result.theta = result[length(result[,1]), -c(1:100)]
# Extracting the f samples, excluding the first 100 values
result.f = result[-length(result[,1]), -c(1:100)]

# Estimate for the posterior marginal for theta
truehist(result.theta, xlab = bquote(theta), main = bquote("Histogram of " ~theta~ " samples"))
```

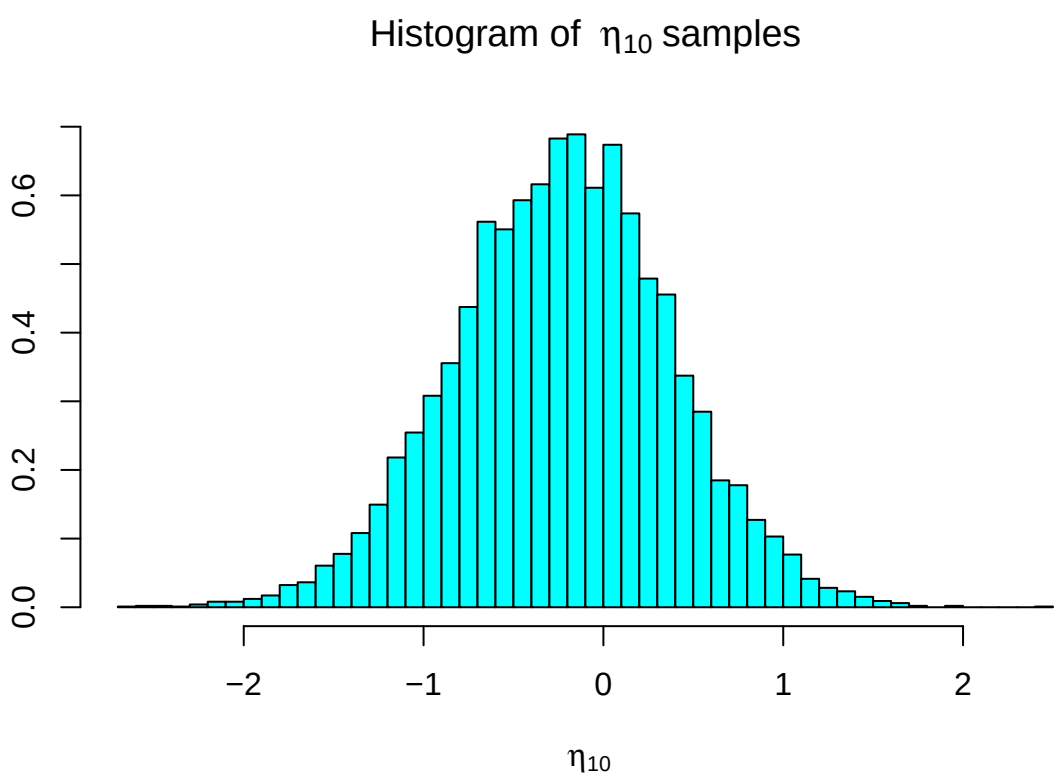



```r
# Vectors for storing the mean, variance and confidence bounds
f.mean = rep(0,T)
f.var = rep(0,T)
conf.upper = rep(0,T)
conf.lower = rep(0,T)
# Calculate the mean and variance
for(t in 1:T) {
  f.mean[t] = mean(result.f[t,])
  f.var[t] = var(result.f[t,])
}
# Calculate 95% confidence bounds
```

```r
for(t in 1:T) {
  z = qnorm(0.025)
  conf.upper[t] = f.mean[t] + z * sqrt(f.var[t])
  conf.lower[t] = f.mean[t] - z * sqrt(f.var[t])
}
# Plotting
t = seq(from = 1, to = T, by = 1)
plot(t, f.mean, type = "l", ylab = "y")
points(t, y)
lines(t, conf.lower, col = "red")
lines(t, conf.upper, col = "red")
```




```r
# Estimate of pi(eta_10|y)
f_10 = result.f[10,]
truehist(f_10, xlab = bquote(~eta[10]), main = bquote("Histogram of " ~eta[10]~"samples"))
```



The first histogram shows an estimate for $\pi(\theta \mid \mathbf{y})$. \ In the second figure, the data points are plotted as circles. The black line is plotted using the estimates of the smooth effects. The red lines are the 95% confidence bounds. Almost all the data points are within the bounds. \ The last histogram of the η_{10} samples provides an estimate of $\pi(\eta_{10} \mid \mathbf{y})$.

3.

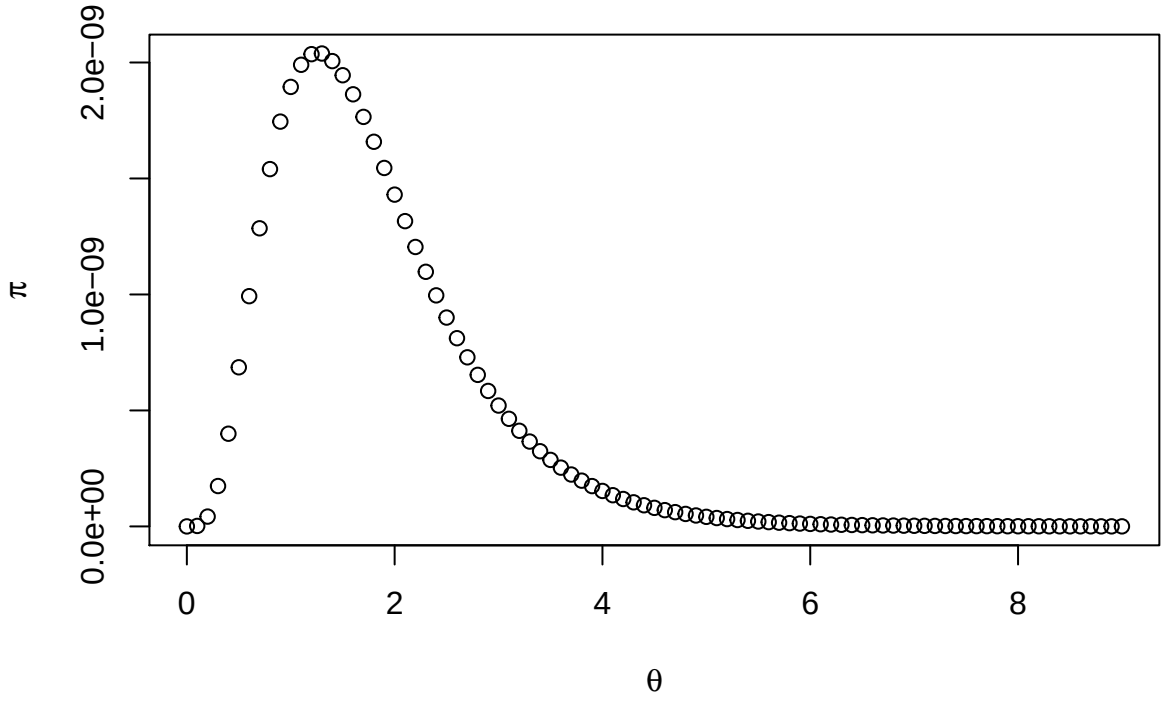
We want to approximate $\pi(\theta \mid \mathbf{y})$ using the INLA scheme. Since we found that $\pi(\eta \mid \theta, \mathbf{y}) \propto \mathcal{N}((\mathbf{Q}+\mathbf{I})^{-1}\mathbf{y}, (\mathbf{Q}+\mathbf{I})^{-1})$, we can calculate

$$\begin{aligned}
\pi(\theta \mid \mathbf{y}) &\propto \frac{\pi(\mathbf{y} \mid \eta, \theta)\pi(\eta \mid \theta)\pi(\theta)}{\pi(\eta \mid \theta, \mathbf{y})} \\
&\propto \frac{\exp(-\frac{1}{2}(\mathbf{y}-\eta)^T(\mathbf{y}-\eta))\theta^{(T-2)/2}\exp(-\frac{1}{2}\eta^T\mathbf{Q}\eta)\exp(-\theta)}{|\mathbf{Q}+\mathbf{I}|^{1/2}\exp(-\frac{1}{2}(\eta-(\mathbf{Q}+\mathbf{I})^{-1}\mathbf{y})^T(\mathbf{Q}+\mathbf{I})(\eta-(\mathbf{Q}+\mathbf{I})^{-1}\mathbf{y}))} \\
&= \theta^{(T-2)/2}|\mathbf{Q}+\mathbf{I}|^{-1/2}\exp\left(-\theta - \frac{1}{2}\mathbf{y}^T(\mathbf{I}-(\mathbf{Q}+\mathbf{I})^{-1})\mathbf{y}\right)
\end{aligned},$$

where $|\cdot|$ denotes the determinant and we have used that $|\mathbf{A}^{-1}| = \frac{1}{|A|}$. We use a grid θ_{grid} of values for θ and calculate the posterior marginal. The plot below shows the result, and it seems to be in concordance with the MCMC estimate displayed by the histogram.

```
# Function to calculate pi(theta|y) for each theta in the grid
pi_theta_y = function(theta.grid, y) {
  pi = rep(0,length(theta.grid))
  T = length(y)
```

```
  for(i in 1:length(pi)){
    theta = theta.grid[i]
    Q = make.Q(T, theta)
    deter = det(solve(Q+diag(T)))
    pi[i] = theta^(T/2-1) * exp(-theta) * deter^(0.5) * exp(-0.5 * t(y) %*% (diag(T)-solve(Q+diag(T))) %
  }
  return(pi)
}


thetas = seq(from = 0, to = 9, by = 0.1)     # Theta grid
pi = pi_theta_y(thetas, y)                   # Corresponding values for pi(theta|y)
plot(thetas, pi, xlab = bquote(theta), ylab = bquote(pi))   # Plotting
```



4.

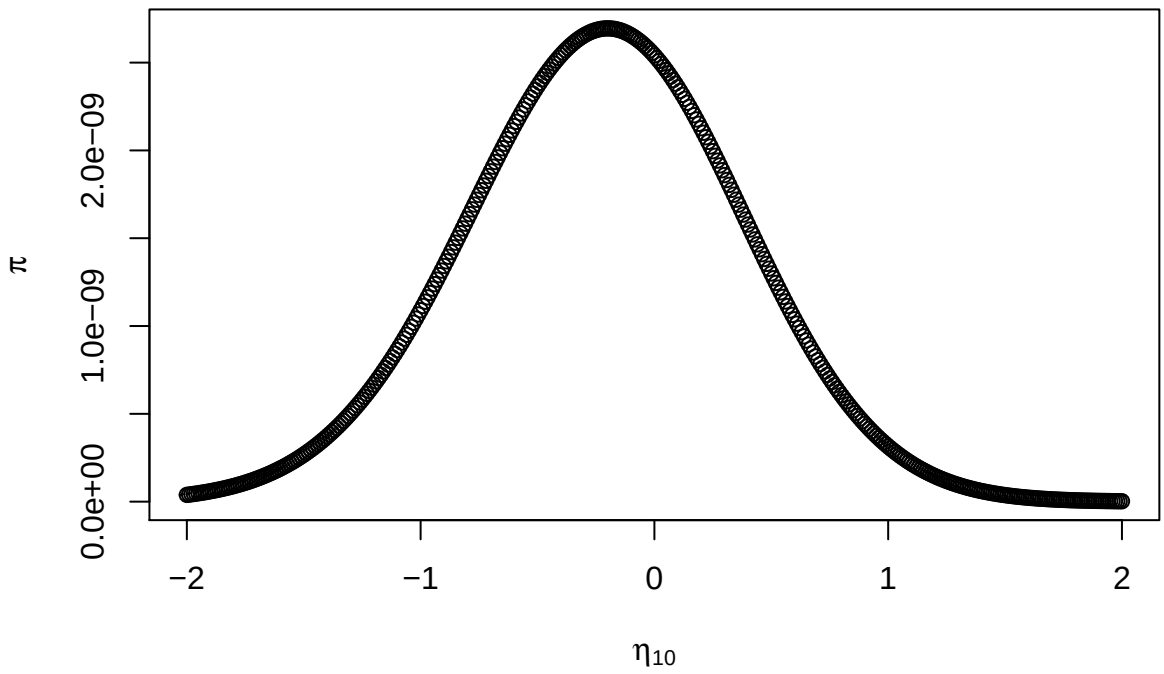
We also want to implement the INLA scheme for the approximation of $\pi(\eta_i \mid \mathbf{y})$. We have

$$\begin{aligned}\pi(\eta_i \mid \mathbf{y}) &= \int \pi(\eta_i \mid \mathbf{y}, \theta)\pi(\theta \mid \mathbf{y})d\theta \\ &\approx \sum_{\theta_k \in \theta_{\text{grid}}} \pi(\eta_i \mid \mathbf{y}, \theta_k)\pi(\theta_k \mid \mathbf{y})\Delta \quad ,\end{aligned}$$

where θ_{grid} is the grid of theta values from point 3, and Δ is the step size between the values in the grid. Since $\pi(\eta \mid \theta, \mathbf{y}) \propto \mathcal{N}((\mathbf{Q}+\mathbf{I})^{-1}\mathbf{y}, (\mathbf{Q}+\mathbf{I})^{-1})$, we assume that $\pi(\eta_i \mid \mathbf{y}, \theta) \sim \mathcal{N}([\mathbf{Ay}]_i, \mathbf{A}_{ii})$, where $\mathbf{A} = (\mathbf{Q}+\mathbf{I})^{-1}$.

We calculate $\pi(\eta_i \mid \mathbf{y})$ for $i = 10$ and values for $\eta_{10} \in [-2, 2]$. The plot below shows the result. The graph looks approximately normal with a small and negative mean, which also the estimation obtained using the block Gibbs sampling (displayed by the last histogram in point 2) does.

```
# Function for calculating pi(eta_10|y,theta_k) for each eta_10 in the grid
pi_etai_y_theta = function(etai.grid, theta, y) {
  i = 10
  T = length(y)
  Q = make.Q(T, theta)
  A = solve(Q + diag(T))
  mean = (A %*% y)[i]
  var = A[i,i]
  pi = dnorm(etai.grid, mean = mean, sd= sqrt(var))
  return(pi)    # Return the vector corresponding to each eta_10 in the grid
}

# Function for calculating pi(eta_10|y) for each eta_10 in the grid
pi_etai_y = function(y,theta.grid, eta.grid) {
  sums = rep(0, length(eta.grid))   # Vector for storing the approximations
  step = theta.grid[2]-theta.grid[1]  # Step size
  theta_y = pi_theta_y(theta.grid, y) # vector of pi(theta|y) for each theta in the grid
  for(k in (1:length(theta.grid))) {
    theta = theta.grid[k]             # theta_k
    sums = sums + pi_etai_y_theta(eta.grid, theta, y) * theta_y[k] * step # Adding the terms for theta_
  }
  return(sums)
}


thetas = seq(from = 0, to = 9, by = 0.1)    # Theta grid
eta.grid = seq(-2,2,0.01)                   # Eta grid
etai_y = pi_etai_y(y,thetas,eta.grid)       # Vector of pi(eta_10|y) for each eta_10 in the grid

plot(eta.grid,etai_y, ylab = bquote(pi), xlab = bquote(eta[10]))    # Plotting the result
```



5.

We now use built in inla function for the same estimates as above. In the first figure the estimated smooth effects using inla are plotted as a red line. The MCMC estimates are also plotted in the same figure as a black line. The estimates are very similar, so the lines are overlapping. The second figure shows the estimate of $\pi(\theta \mid \mathbf{y})$, which looks very similar to the ones from point 2 and 3. The last figure shows the estimate for $\pi(\eta_{10} \mid \mathbf{y})$, and it looks like the estimates from point 2 and 4.

```
library(INLA)
```

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

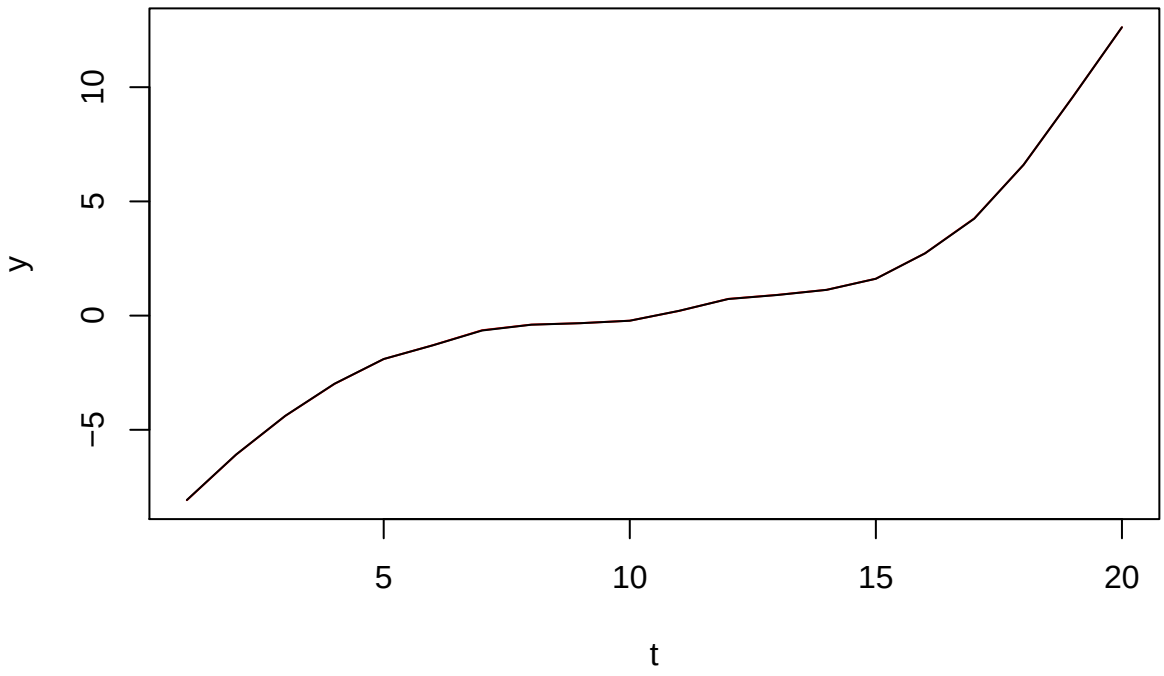
```
## Warning: package 'sp' was built under R version 3.6.3
```

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

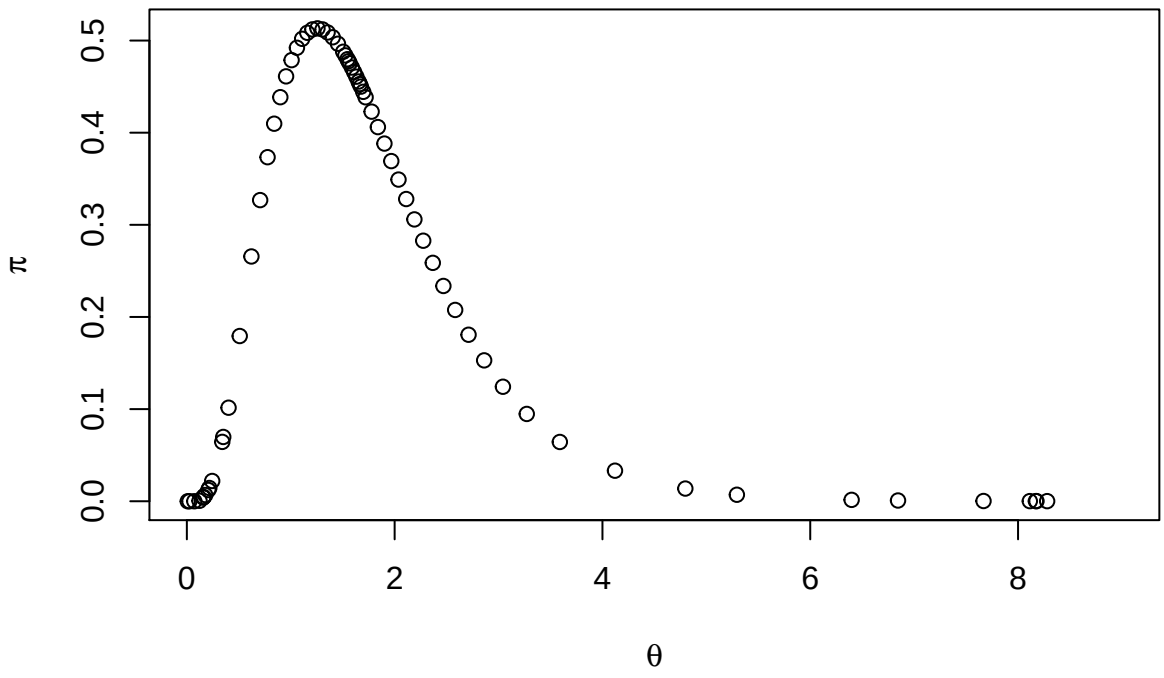
```
## This is INLA_19.09.03 built 2019-09-03 09:03:02 UTC.
## See www.r-inla.org/contact-us for how to get help.
```

```
T = 20
t = seq(from = 1, to = T, by = 1)
data = data.frame(y = y, t = t)
```

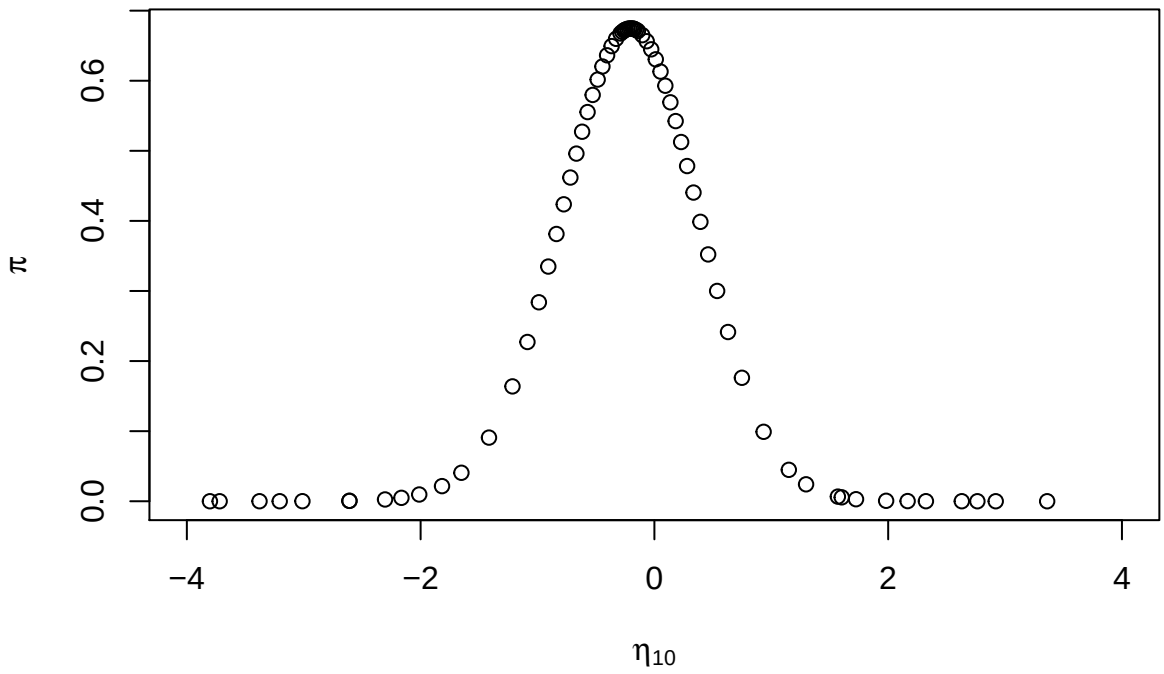
```
thetahyper = list(theta = list(prior = "log.gamma", param = c(1, 1)))
formula = y   f(t, model = "rw2", hyper = thetahyper, constr = FALSE) - 1
result1 = INLA::inla(formula = formula, family = "gaussian", data = data, control.family = list(hyper=li

plot(result1$summary.random$t$mean, xlab="t", ylab="y", type="l", col="red")
lines(t, f.mean)
```




```
plot(result1$marginals.hyperpar$`Precision for t`, xlim =c(0,9), xlab=bquote(theta),ylab=bquote(pi))
```




```
plot(result1$marginals.random$t$index.10,xlim=c(-4,4),xlab=bquote(eta[10]),ylab=bquote(pi))
```


π
0.0
0.2
0.4
0.6
−4
−2
0
2
4
η10