Module 10: Solutions

Exercise 1: Mining Disasters

This examples is concerned with mining disasters in British mines from 1851 and 26,500 days forward, where the time for 109 accidents is measured in days since 1851. Read in the data set `mining.dat` in R:

```r
data_url <- "https://asta.math.aau.dk/course/bayes/2020/?file=mining.dat"
mining <- read.table(data_url)
mining <- mining[,1]
```

We model the data as a Poisson process with intensity function $\alpha(t)$, $t \in [0, 26500]$. This means that the number of disasters in an interval $[t_1, t_2]$ is Poisson distributed with parameter $\int_{t_1}^{t_2} \alpha(t)dt$. If t_i denotes the time of the ith disaster, then the log likelihood is given by

$$\sum_{i=1}^{109} \log(\alpha(t_i)) - \int_0^{26500} \alpha(t)dt.$$

During the period which the data covers, a law was introduced which greatly improved the safety in mines. This leads us to assume that α takes one value before this law was introduced and another after: Let j denote the time (measured in days since 1851) at which the change takes place – we call j for a **change point** – then we assume that $\alpha(t) = a$ if $t < j$ and $\alpha(t) = a + d$ if $t \geq j$, where $0 < a + d \leq a$. If the times of the disasters are in a vector `mining`, you can define a log likelihood function in R as

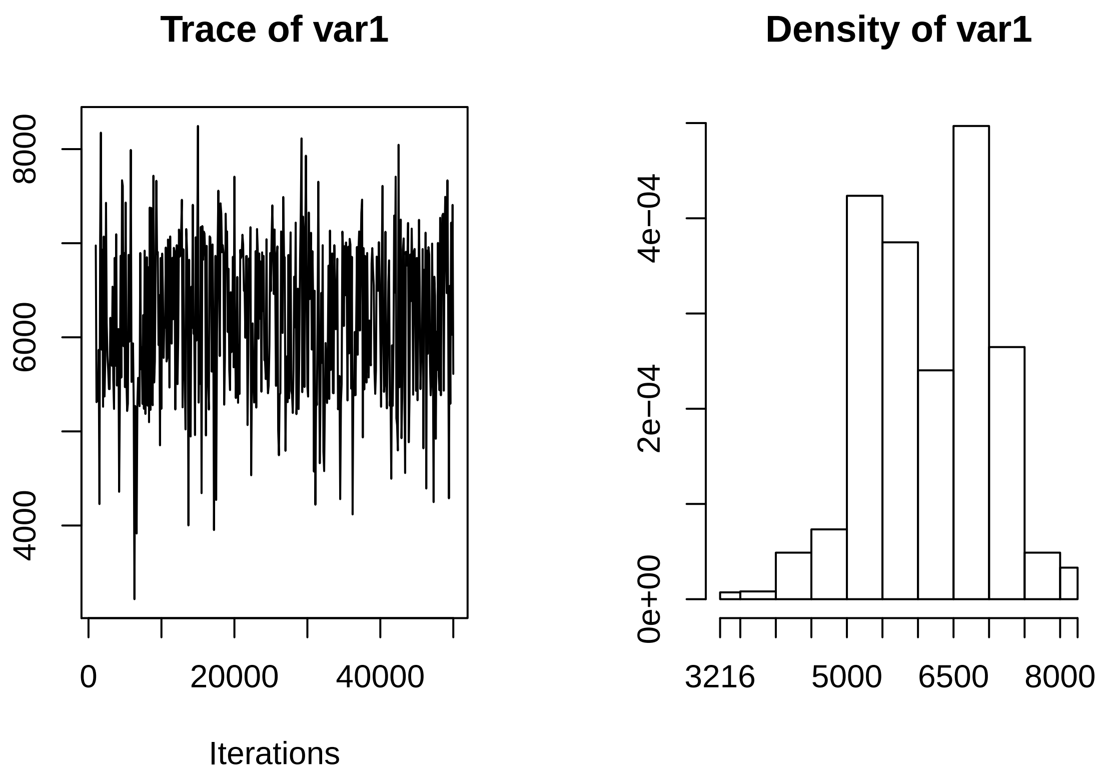
```r
llik <- function(j, a, d){
  ## BEWARE: We refer to the global variable mining here!
  sum(log(a+(mining>j)*d))- (j*a + (26500-j)*(a+d))
}
```

Assume for j a uniform prior on the interval $[0, 26500]$, whilst that $a = 0.007$ and $d = -0.004$ are known. Then construct a Metropolis-Hastings algorithm which generates a sample from the posterior distribution of j.

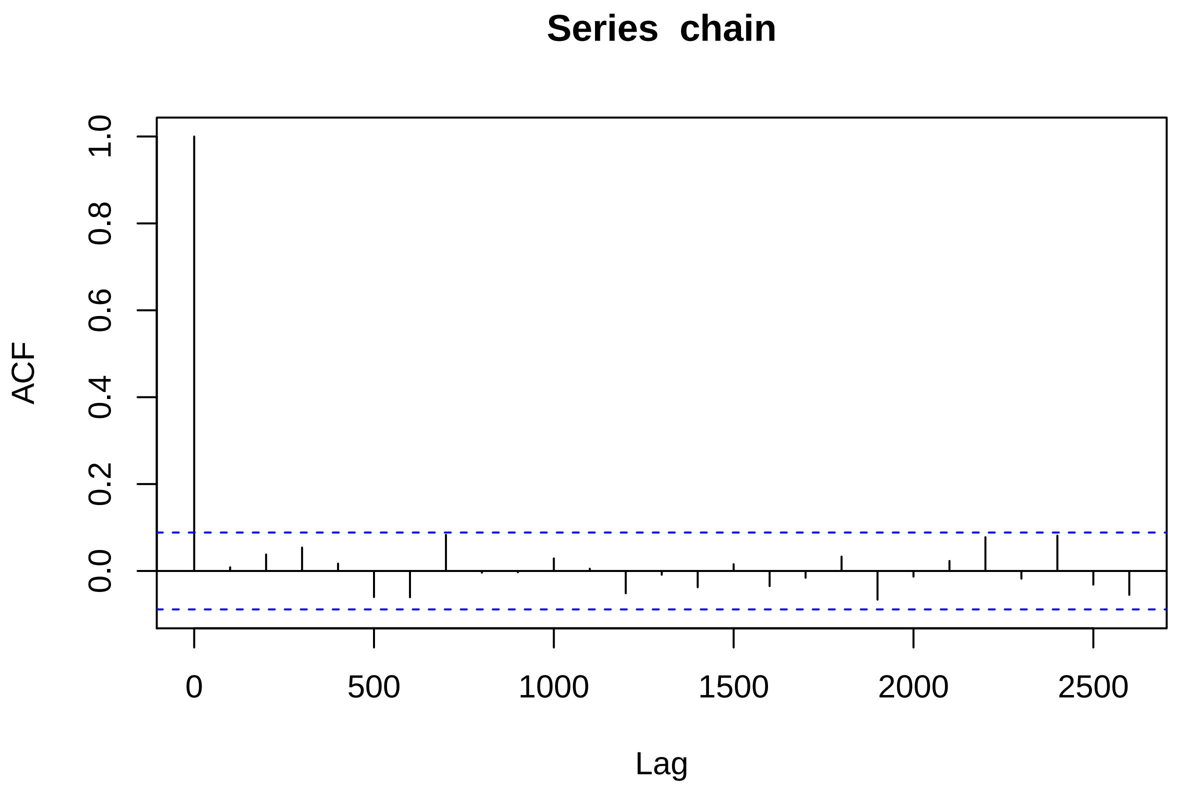
```r
library(coda)
lposterior <- function(j, a = 0.007, d = -0.004){
  if(j<0 | j>26500){
    return(-Inf)
  }
  llik(j, a, d)
}
myMH2 <- function(N, sigma = 1, a=0.007, d=-0.004, j0 = 5000){
  j <- rep(0, N) # Empty Markov chain
  j[1] <- j0 ## initial value
  for(i in 2:N){
    j_new <- sample(seq(j[i-1]-sigma, j[i-1]+sigma)[-(sigma+1)], size = 1)
    logH <- lposterior(j_new, a, d) - lposterior(j[i-1], a, d)
    if(log(runif(1))<logH){
      j[i] <- j_new
    }else{
      j[i] <- j[i-1]
    }
  }
  # Finally return the results
  return(mcmc(j))
}
# Try it out:
chain <- myMH2(50000, sigma = 4000)
chain <- window(chain, start = 1000, thin = 100)
summary(chain)
```

```
##
## Iterations = 1000:50000
## Thinning interval = 100
## Number of chains = 1
## Sample size per chain = 491
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##           Mean             SD       Naive SE Time-series SE
##        6166.96         859.38          38.78          38.78
##
## 2. Quantiles for each variable:
##
##  2.5%   25%   50%   75% 97.5%
##  4367  5452  6100  6905  7642
```

```
plot(chain)
```



```
acf(chain)
```



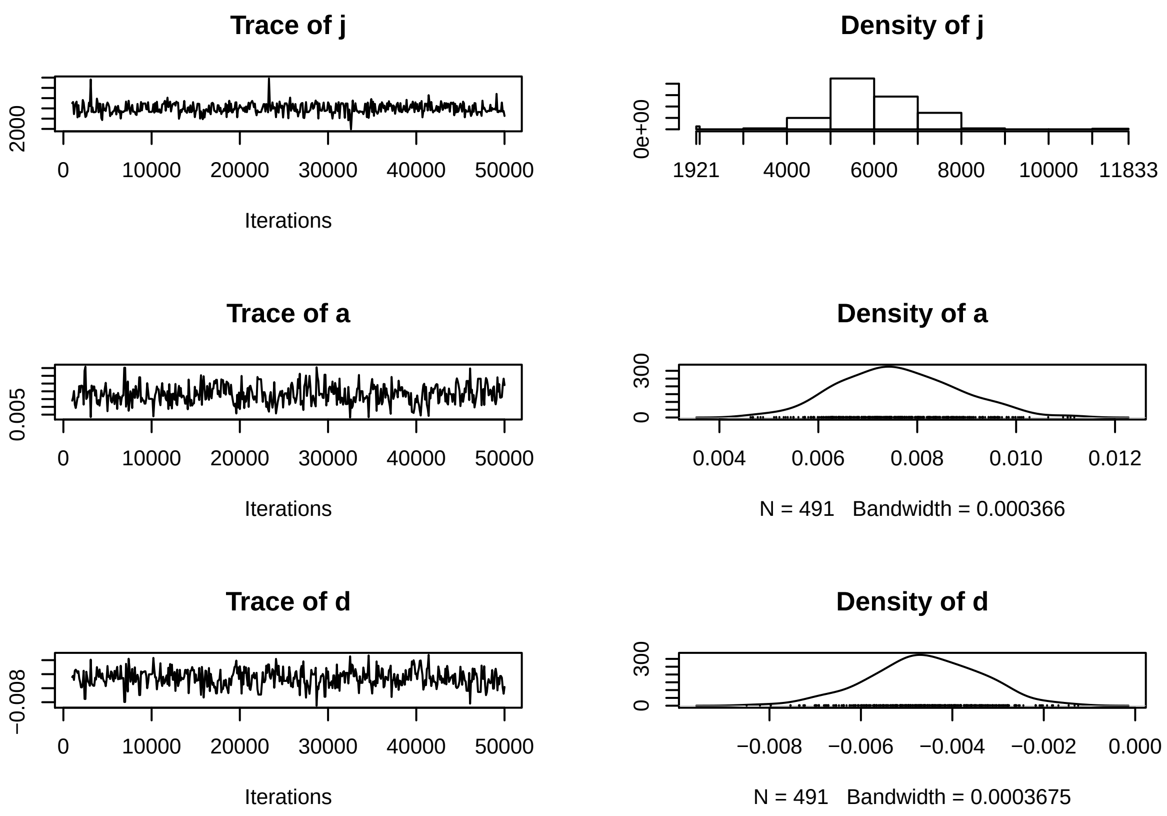
Discuss how to impose a prior on a and d and extend your code to this case of doing posterior simulations.

Below is a quick sketch of what could be done. It is not tested in detail.

```
lprior <- function(a,d){
  if(d>0 | d < -a){
    return(-Inf)
  } else{
    return(dgamma(a, shape = 1, scale = 109/26500, log = TRUE) + log(1/a))
  }
}
lposterior <- function(j, a, d){
  if(j<0 | j>26500){
    return(-Inf)
  }
  llik(j, a, d) + lprior(a,d)
}
myMH3 <- function(N, sigma_j = 1, sigma_a = 0.001, a0=0.007, d0=-0.004, j0 = 5000){
  j <- a <- d <- rep(0, N) # Empty Markov chains
  j[1] <- j0 ## initial value
  a[1] <- a0
  d[1] <- d0
  for(i in 2:N){
    j_new <- sample(seq(j[i-1]-sigma_j, j[i-1]+sigma_j)[-(sigma_j+1)], size = 1)
    a_new <- rnorm(1, a[i-1], sd = sigma_a)
    if(a_new>0){
      d_new <- runif(1, -a_new, 0)
      logH <- lposterior(j_new, a_new, d_new) - lposterior(j[i-1], a[i-1], d[i-1])
    }
    if(a_new>0 && log(runif(1))<logH){
        j[i] <- j_new
        a[i] <- a_new
        d[i] <- d_new
    } else{
      j[i] <- j[i-1]
      a[i] <- a[i-1]
      d[i] <- d[i-1]
    }
  }
  # Finally return the results
  return(mcmc(cbind(j=j, a=a, d=d)))
}
# Try it out:
chain <- myMH3(50000, sigma_j = 4000)
chain <- window(chain, start = 1000, thin = 100)
summary(chain)
```

```
##
## Iterations = 1000:50000
## Thinning interval = 100
## Number of chains = 1
## Sample size per chain = 491
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##         Mean        SD  Naive SE Time-series SE
## j  5.988e+03 1.002e+03 4.520e+01      4.864e+01
## a  7.618e-03 1.199e-03 5.411e-05      7.217e-05
## d -4.591e-03 1.224e-03 5.525e-05      7.035e-05
##
## 2. Quantiles for each variable:
##
##         2.5%        25%        50%        75%      97.5%
## j  4.133e+03  5.332e+03  5.804e+03  6.856e+03  7.504e+03
## a  5.396e-03  6.784e-03  7.545e-03  8.382e-03  9.987e-03
## d -6.992e-03 -5.342e-03 -4.609e-03 -3.738e-03 -2.179e-03
```

```
plot(chain)
```



Exercise 2: Speed of light

In this exercise you are asked to reproduce the model checking results for the speed of light example in the lecture. The 66 measurements of time can be found in the data file `newcomb.csv`:

```r
data_url <- "https://asta.math.aau.dk/course/bayes/2020/?file=newcomb.csv"
newcomb <- read.csv(data_url)
```

That is you have to:

1. Implement a Gibbs sampler to sample from the posterior distribution (**Hint:** You have implemented a similar sampler in a previous exercise.)
2. Run the sampler for a sufficient amount of time and show histograms of the parameters.
3. Generate samples from the predictive distribution, calculate the minimum for each sample and show a histogram of the minimums compared to the data minimum.

Prior parameters

```r
mu0 <- 0
tau0 <- 0.001
alpha <- 0.001
beta <- 1000
```

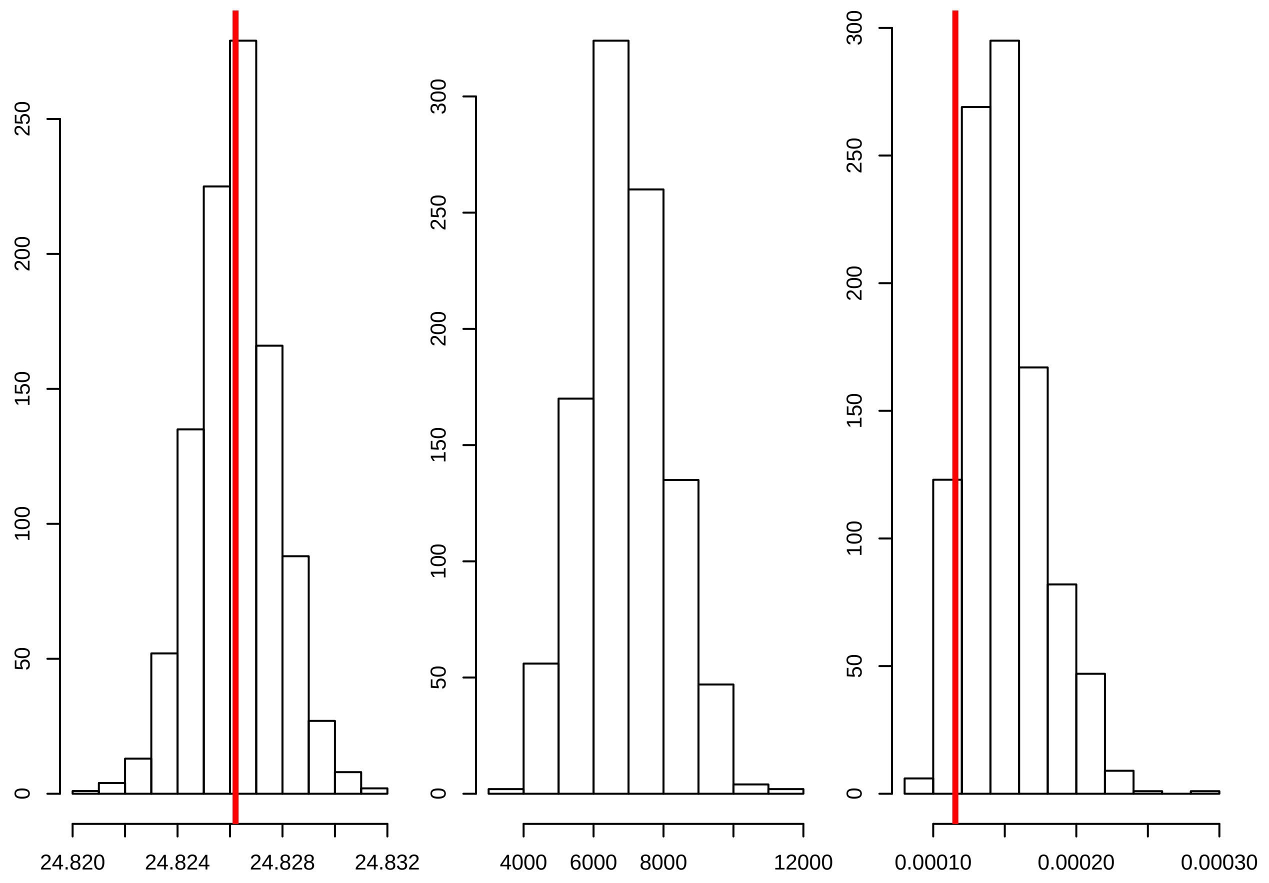
Gibbs sampler

```r
set.seed(1234)
n <- nrow(newcomb)
```

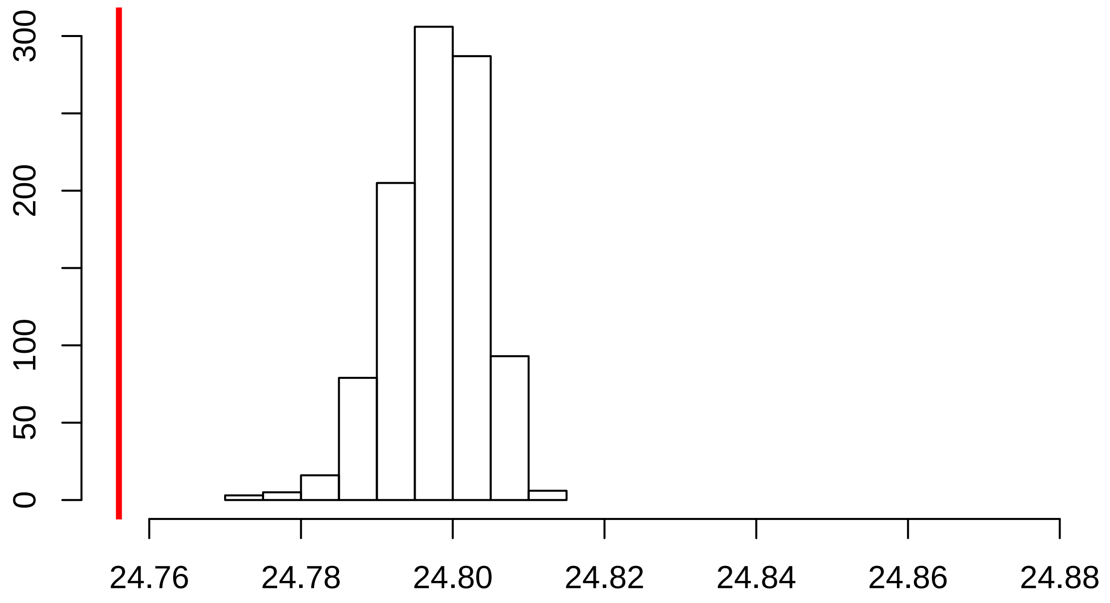
```r
times <- 1000
mu <- tau <- rep(0, times)
mu[1] <- mean(newcomb$time)
tau[1] <- 1/var(newcomb$time)
for(i in 2:times){
  cond_mean <- (n*tau[i-1]*mean(newcomb$time)+tau0*mu0)/(n*tau[i-1]+tau0)
  cond_tau <- n*tau[i-1]+tau0
  mu[i] <- rnorm(1, cond_mean, 1/sqrt(cond_tau))
  cond_shape <- n/2+alpha
  cond_scale <- 1/(0.5*sum((newcomb$time-mu[i])^2)+1/beta)
  tau[i] <- rgamma(1, shape = cond_shape, scale = cond_scale)
}
```

Posterior distribution

```r
par(mfrow=c(1,3), mar=c(2,2,0,1))
hist(mu, main="")
abline(v=mean(newcomb$time), lwd=3, col="red")
hist(tau,main="")
hist(1/tau,main="")
abline(v=var(newcomb$time), lwd=3, col="red")
```



```r
x <- matrix(0, n, times)
for(i in 1:times){
  x[,i] <- rnorm(n, mu[i], 1/sqrt(tau[i]))
}
minimums <- apply(x, 2, min)
hist(minimums, xlim=range(newcomb$time,x), xlab="", ylab="", main="")
abline(v=min(newcomb$time), lwd=3, col="red")
```


0
50
100
200
300
24.76
24.78
24.80
24.82
24.84
24.86
24.88