R Lab. - Exercise 8

Michele Guadagnini - Mt. 1230663

June 10, 2020

Exercise 1 - Zeeman effect experiment

Students from the Bachelor degree in Physics performed an experiment to study the Zeeman effect. The apparatus contains a Ne source lamp whose position can be changed. During the setting up of the apparatus, the source position has to be adjusted in order to maximize the intensity of the detected light signal.

The following table gives the position of the source (in mm) and the corresponding height of the peak (arbitrary units) for the wavelength under study:

x_i	2.44	3.49	3.78	3.31	3.18	3.15	3.1	3	3.6	3.4
y_i	129	464	189	562	589	598	606	562	360	494

Assume a quadratic dependence of the peak height, y_i, as a function of the source position x_i,

$$f(x) = c_0 + c_1 x + c_2 x^2 \tag{1}$$

All the measured values are affected by a Gaussian noise with zero mean, such that

$$y_i = f(x_i) + \epsilon \tag{2}$$

where ϵ follows a normal distribution with mean $\mu = 0$ and unknown standard deviation, σ.

A) Build a Markov Chain Monte Carlo to estimate the best parameters of the quadratic dependence of the data and the noise that affects the measured data.

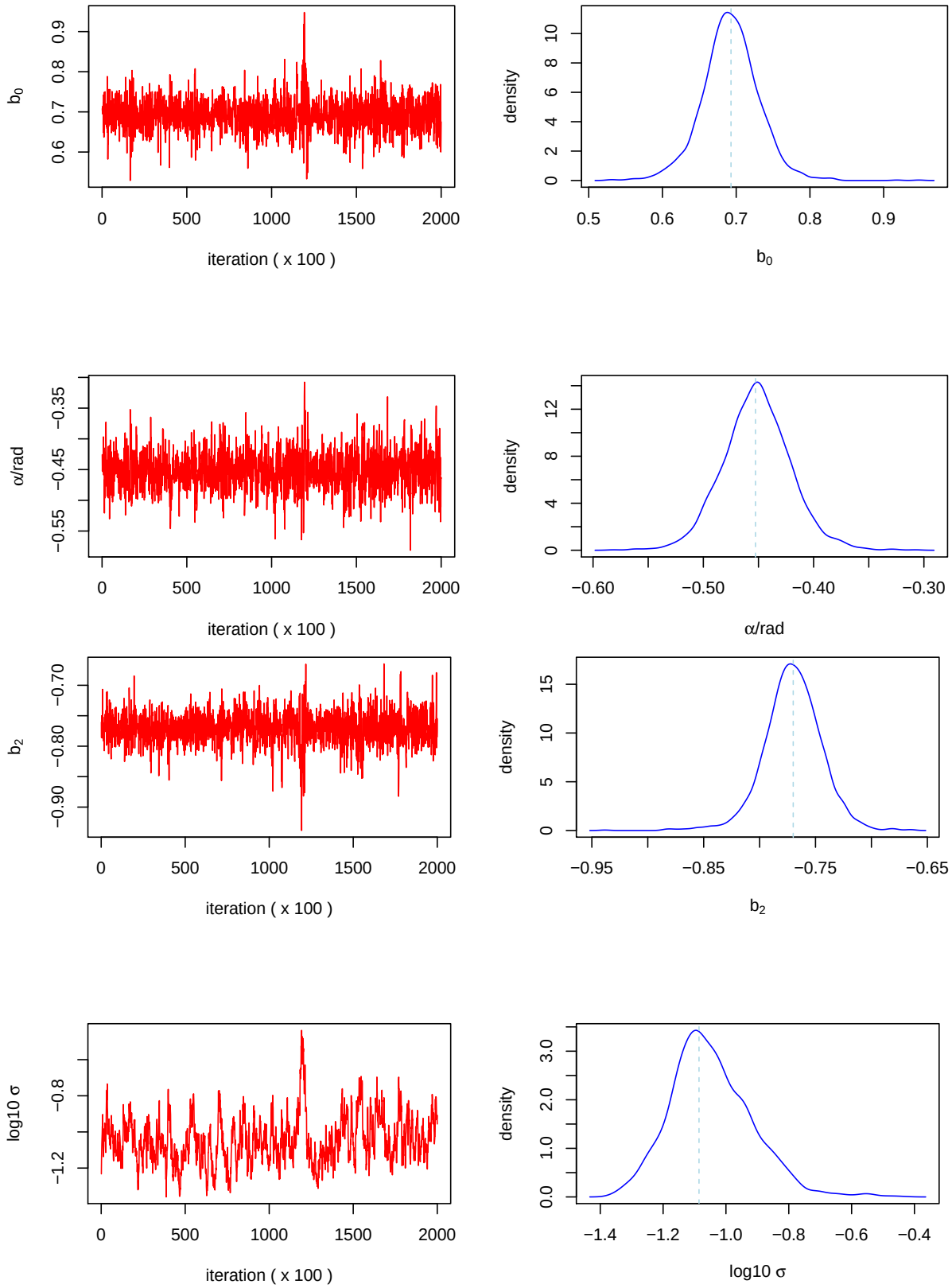
```
# define metropolis algorithm
metrop <- function(func, theta.init, nBurnIn, nsamples, smCov, verbose, data) {

  theta.cur <- theta.init
  func.cur <- func(theta.cur, data)
  func.samp <- matrix(data=NA , nrow=nsamples, ncol=2+length(theta.init))
  n.accept <- 0
  rate.accept <- 0.0

  for(n in 1:(nBurnIn+nsamples)) {

    theta.prop <- rmvnorm(n=1, mean=theta.cur, sigma=smCov, method="eigen")
    func.prop  <- func(theta.prop, data)
    logMR <- sum(func.prop) - sum(func.cur)     # Log10 of the Metropolis ratio

    if ( logMR>=0 || logMR>log10(runif(1)) ) {
      theta.cur <- theta.prop
      func.cur  <- func.prop
      n.accept  <- n.accept + 1
      rate.accept <- n.accept/n
    }

    if (n > nBurnIn) {
      func.samp[n-nBurnIn, 1:2] <- func.cur
```

```
      func.samp[n-nBurnIn, 3:(2+length(theta.cur))] <- theta.cur
    }

    if (n %% verbose == 0) {
      cat(paste("Iteration:",n,"  accepted:",n.accept,
                "  accept.rate:",rate.accept,"\n"))
    }
  }
  return (func.samp)
}

# define the log prior for the parameters
logprior.quadraticmodel <- function(theta) {
  b0Prior <- dnorm( theta[1], mean=0, sd=10)
  aPrior <- 1
  b2Prior <- dnorm( theta[3], mean=0, sd=5)
  logysigPrior <- 1
  logPrior <- sum( log10(b0Prior), log10(aPrior),
                   log10(b2Prior), log10(logysigPrior) )
  return (logPrior)
}

# define the log likelihood
loglike.quadraticmodel <- function(theta, data) {

  theta[2] <- tan(theta[2])
  theta[4] <- 10^theta[4]
  modPred <- drop( theta[1:3] %*% t(cbind(1, data$x, data$x^2)) )

  logLike <- (1/log(10))*sum( dnorm(modPred - data$y, mean=0, sd=theta[4], log=TRUE) )

  return(logLike)
}

# define the log posterior
logpost.quadraticmodel <- function(theta, data) {

  logprior <- logprior.quadraticmodel(theta)

  if(is.finite(logprior)) {
    return( c(logprior, loglike.quadraticmodel(theta, data)) )
  } else {
    return( c(-Inf, -Inf) )
  }
}

# import data
xs <- c(2.44, 3.49, 3.78, 3.31, 3.18, 3.15, 3.1, 3, 3.6, 3.4)
ys <- c(129, 464, 189, 562, 589, 598, 606, 562, 360, 494)

# data standardization
xs.norm <- (xs - mean(xs))/sd(xs)
ys.norm <- (ys - mean(ys))/sd(ys)
```

```r
data <- data.frame(cbind(xs.norm, ys.norm))

# run the MCMC
sampleCov <- diag(c(0.1, 0.01, 0.01, 0.01)^2)
thetaInit <- c(27.4, atan(-11.7), 1.18, log10(2.4))
set.seed(250)
allSamp <- metrop(func = logpost.quadraticmodel, theta.init = thetaInit,
                  nBurnIn =2e4, nsamples =2e5, smCov = sampleCov,
                  verbose =2.2e5, data = data)
```

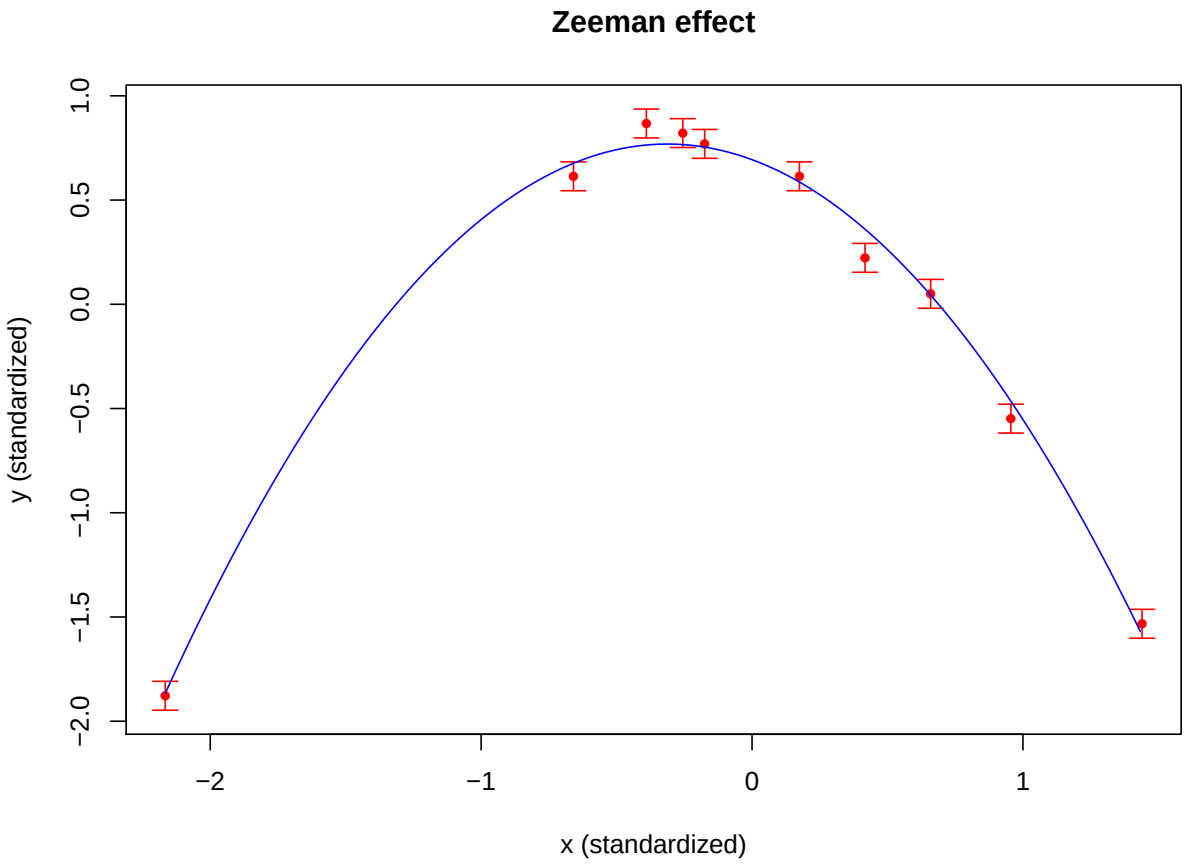
```
## Iteration: 220000   accepted: 73970   accept.rate: 0.336234914429873
```

```r
# estimation of true values
xs.norm2 <- I(xs.norm^2)
quadratic.fit <- lm(ys.norm ~ xs.norm + xs.norm2)

#summary(quadratic.fit)
fit.res <- c(0.69295, -0.48641, -0.76994) # fit parameters
sigNoise <- 0.08209 # residuals standard error
theta.true <- c(fit.res[1], atan(fit.res[2]), fit.res[3], log10(sigNoise))

# thinning
thin.by <- 100
thinIds <- seq(1, nrow(allSamp), by=thin.by)
thinSamp <- allSamp[thinIds,]

# plot the results (marginal posteriors)
par(mfrow=c(2,2))
parnames <- c( expression(b[0], paste(alpha,"/rad"), b[2], paste("log10 ",sigma)) )
for(col in (1:length(thetaInit)+2)) {
    plot(1:nrow(thinSamp), thinSamp[,col], col="red", type="l",
         ylab=parnames[col-2], xlab=paste("iteration ( x",thin.by,")") )

    thinDen <- density(thinSamp[,col], n=2^10)
    plot(thinDen$x, thinDen$y, col="blue", type="l", ylab="density",
         xlab=parnames[col-2])

    abline(v=theta.true[col-2], lty=2, col="lightblue")
}
```


b0
0.6
0.7
0.8
0.9
0
500
1000
1500
2000
iteration (x 100)
density
0
2
4
6
8
10
0.5
0.6
0.7
0.8
0.9
b0
α/rad
−0.55
−0.45
−0.35
iteration (x 100)
density
0
4
8
12
−0.60
−0.50
−0.40
−0.30
α/rad
b2
−0.90
−0.80
−0.70
iteration (x 100)
density
0
5
10
15
−0.95
−0.85
−0.75
−0.65
b2
log10 σ
−1.2
−0.8
iteration (x 100)
density
0.0
1.0
2.0
3.0
−1.4
−1.2
−1.0
−0.8
−0.6
−0.4
log10 σ

```
#
#
#
```

```r
# estimation of the best parameters
posMAP    <- which.max(thinSamp[,1] + thinSamp[,2])
thetaMAP  <- thinSamp[posMAP, 3:6]
covMatrix  <- cov(thinSamp[, 3:6]) # covariance
corrMatrix <- cor(thinSamp[, 3:6]) # correlation

b0 <- thetaMAP[1]
b1 <- tan(thetaMAP[2])
b2 <- thetaMAP[3]
sig.norm <- 10^thetaMAP[4]

par(mfrow=c(1,1))
plotCI(xs.norm, ys.norm, col="red", pch=20, uiw=sig.norm, gap=0,
       xlab="x (standardized)", ylab="y (standardized)", main="Zeeman effect")
xx <- seq(min(xs.norm), max(xs.norm), by=0.01)
yy <- b0 + b1*xx + b2*xx*xx
lines(xx, yy, col="blue")
```



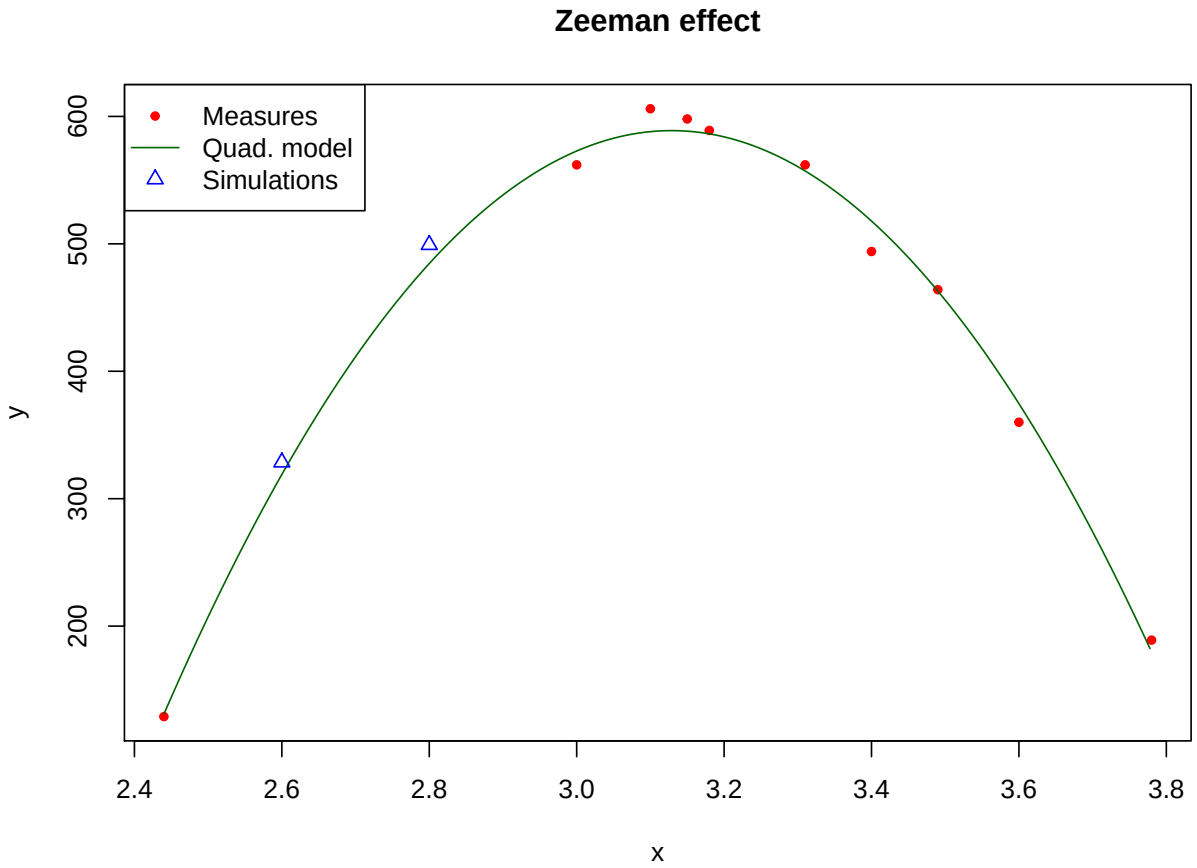
The resulting best estimates of the parameters are:

- c_0 : 0.6936 ± 0.0015
- c_1 : -0.48 ± 0.001
- c_2 : -0.767 ± 0.001
- σ : 0.069 ± 0.017

As can be seen from our data, the students forgot to take measurements in the region x $\in$ (2.44, 3.0).

B) Run a Markov Chain Monte Carlo to predict peak height measurements at $x_1 = 2.8$ mm and $x_2 = 2.6$ mm

```
# new measurements
x1 <- 2.8; x2 <- 2.6
# standardization of new measures
x1.norm <- (x1 - mean(xs))/sd(xs)
x2.norm <- (x2 - mean(xs))/sd(xs)
# compute simulated data
y1.norm <- b0 + x1.norm*tan(b1) + (x1.norm^2)*b2 + rnorm(1, mean=0, sd=sig.norm)
y1 <- y1.norm*sd(ys) + mean(ys)
y2.norm <- b0 + x2.norm*tan(b1) + (x2.norm^2)*b2 + rnorm(1, mean=0, sd=sig.norm)
y2 <- y2.norm*sd(ys) + mean(ys)

plot(xs, ys, col="red", pch=20, main="Zeeman effect", xlab="x", ylab="y")
xx.std <- seq(min(xs.norm), max(xs.norm), by=0.01)
yy.std <- b0 + b1*xx.std + b2*xx.std**2
xx <- xx.std*sd(xs) + mean(xs)
yy <- yy.std*sd(ys) + mean(ys)
lines(xx, yy, col="darkgreen")
points(c(x1, x2), c(y1, y2), pch=24, col="blue")
legend("topleft", c("Measures", "Quad. model", "Simulations"),
       col=c("red","darkgreen","blue"), lty=c(0,1,0), pch=c(20,NA,24))
```



Exercise 2 - British coal mine disasters

The number of British coal mine disasters has been recorded from 1851 to 1962. By looking at the data it seems that the number of incidents decreased towards the end of the sampling period. We model the data as follows: before some year, we call τ, the data follow a Poisson distribution, where the logarithm of the mean value, $\log(\mu_t) = b_0$, while for later years, we can model it as $\log(\mu_t) = b_0 + b_1$. The dependence can be modeled as follows: $y_t \sim \text{Pois}(\mu_t)$, where $\log(\mu_t) = b_0 + b_1$ Step(t - τ).

Implement the model in jags, trying to infer the parameters b_0, b_1 and τ. Assign a uniform prior to b_0, b_1 and a uniform prior in the interval (1, N), where N = 112 is the number of years our data span on.

Before running jags, assign an initial value to the parameters as follows: $b_0 = 0$, $b_1 = 0$ and $\tau = 50$.

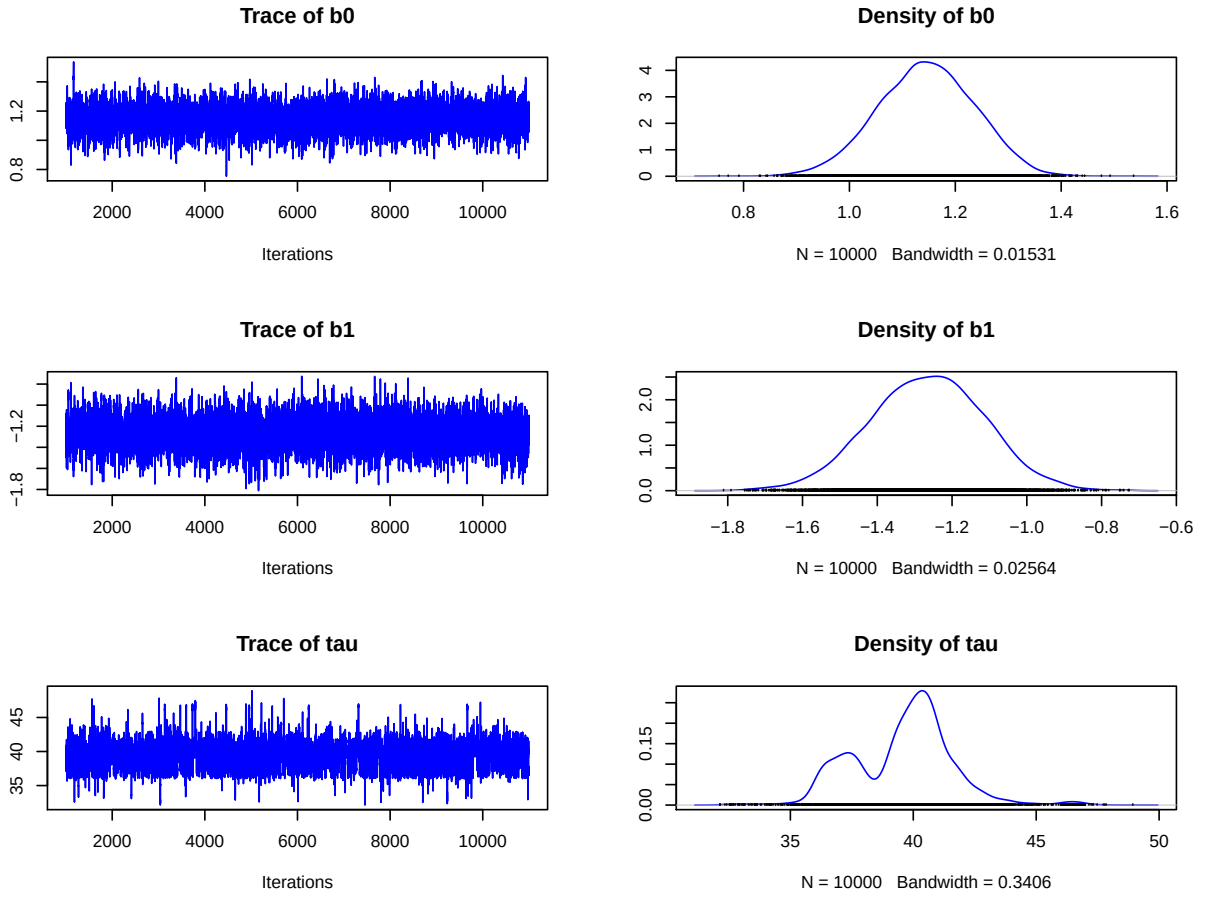
A) Plot the posterior distributions of the parameters and extract their mean values, and 95% credible intervals.

```
# Import data
data <- NULL
data$D <- c( 4 , 5 , 4 , 1 , 0 , 4 , 3 , 4 , 0 , 6 , 3 , 3 , 4 , 0 , 2 ,
             6 , 3 , 3 , 5 , 4 , 5 , 3 , 1 , 4 , 4 , 1 , 5 , 5 , 3 , 4 ,
             2 , 5 , 2 , 2 , 3 , 4 , 2 , 1 , 3 , 2 , 1 , 1 , 1 , 1 , 1 ,
             3 , 0 , 0 , 1 , 0 , 1 , 1 , 0 , 0 , 3 , 1 , 0 , 3 , 2 , 2 ,
             0 , 1 , 1 , 1 , 0 , 1 , 0 , 1 , 0 , 0 , 0 , 2 , 1 , 0 , 0 ,
             0 , 1 , 1 , 0 , 2 , 2 , 3 , 1 , 1 , 2 , 1 , 1 , 1 , 1 , 2 ,
             4 , 2 , 0 , 0 , 0 , 1 , 4 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 ,
             0 , 1 , 0 , 0 , 1 , 0 , 0 )
data$N <- 112

# create the model in the bug file
cat("model {
      for (t in 1:N) {
         mu[t] <- exp(b0 + b1*step(t-tau))
         D[t] ~ dpois(mu[t])
      }

      # Prior
      b0  ~ dunif(0, 4)
      b1  ~ dunif(-4, 0)
      tau ~ dunif(1, N)
    }", file="model_coal_mine.bug")

# set initial values and load the model
init <- NULL
init$b0  <- 0
init$b1  <- 0
init$tau <- 50

model.bug <- "model_coal_mine.bug"
jag.mod <- jags.model(model.bug, data, init)
```

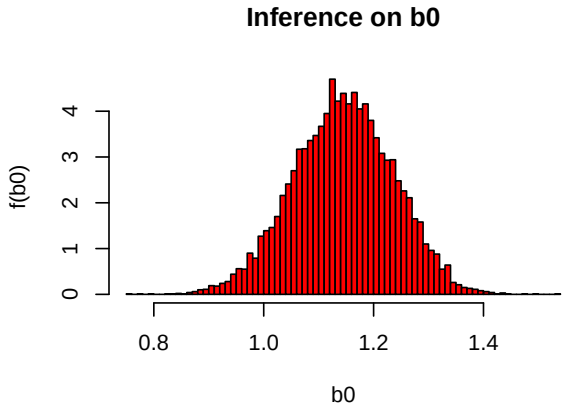
```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 112
##    Unobserved stochastic nodes: 3
```

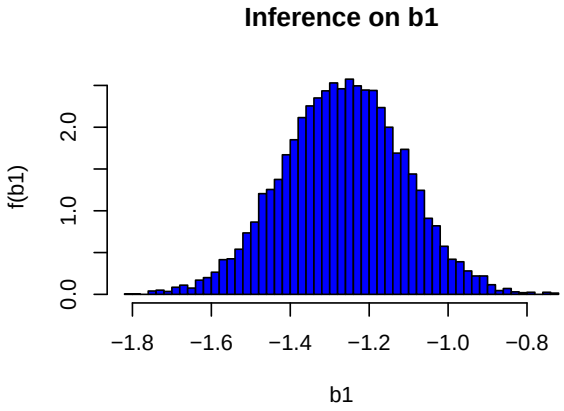
```
##    Total graph size: 790
##
## Initializing model
```

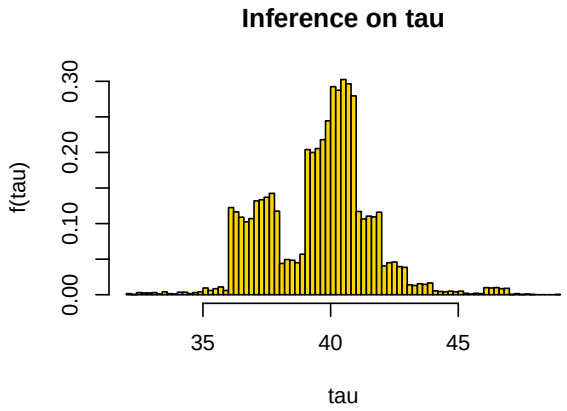
```r
# run the model
chain <- coda.samples(jag.mod, c("b0","b1","tau"), n.iter=10000)

# plot results
plot(chain, col="blue")
```




```r
chain.df <- as.data.frame(as.mcmc(chain))
par(mfrow=c(2,2))
hist(chain.df$b0 , nc=60, prob=TRUE, col="red", xlab="b0",
     ylab="f(b0)", main="Inference on b0")
hist(chain.df$b1 , nc=60, prob=TRUE, col="blue", xlab="b1",
     ylab="f(b1)", main="Inference on b1")
hist(chain.df$tau, nc=60, prob=TRUE, col="gold", xlab="tau",
     ylab="f(tau)", main="Inference on tau")

res <- summary(chain)
```








```
# parameters mean and standard deviation
res$statistics[,1:2]
```

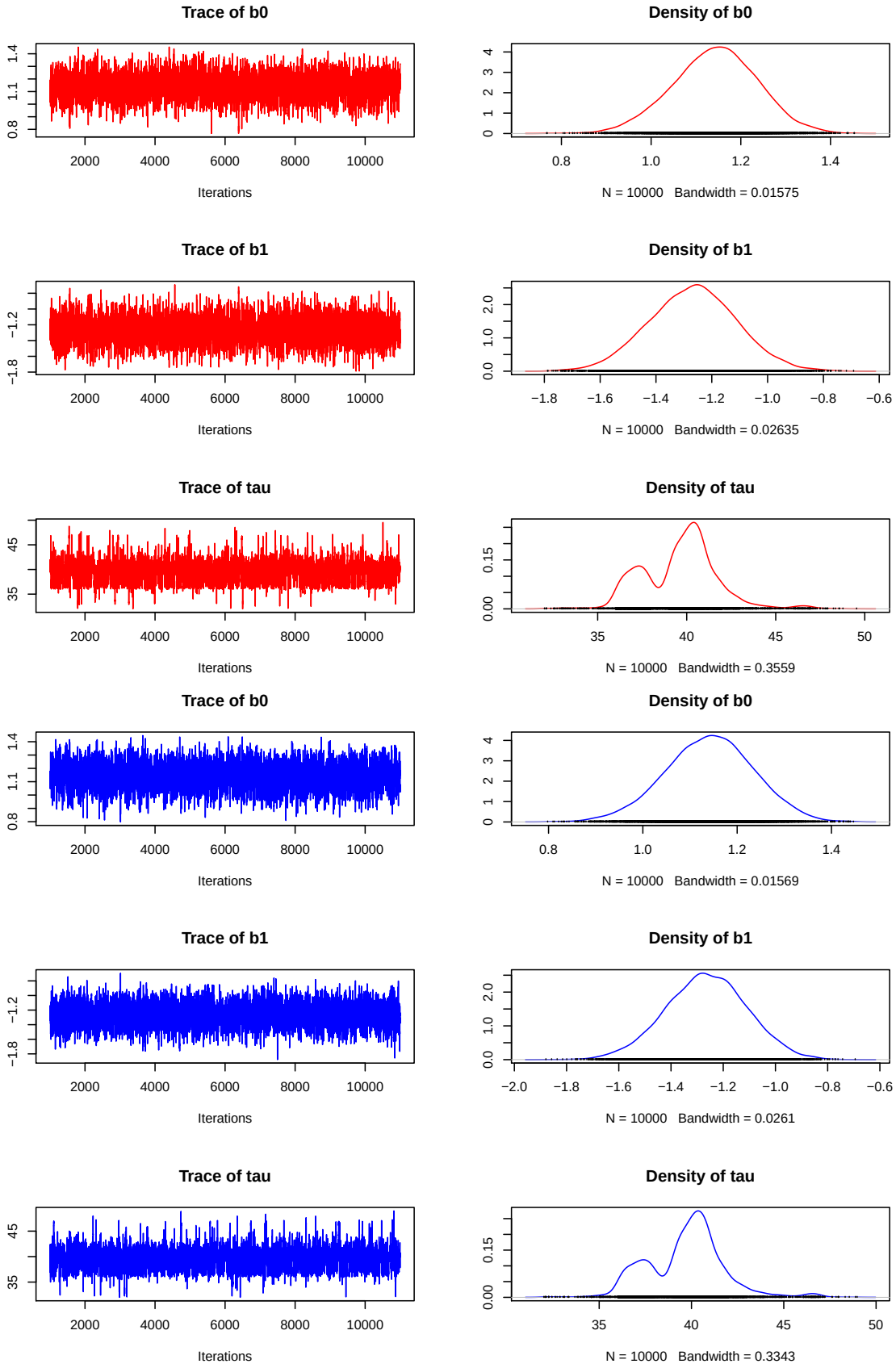
```
##          Mean         SD
## b0   1.145258 0.09112164
## b1  -1.266191 0.15264847
## tau 39.586806 2.02730908
```

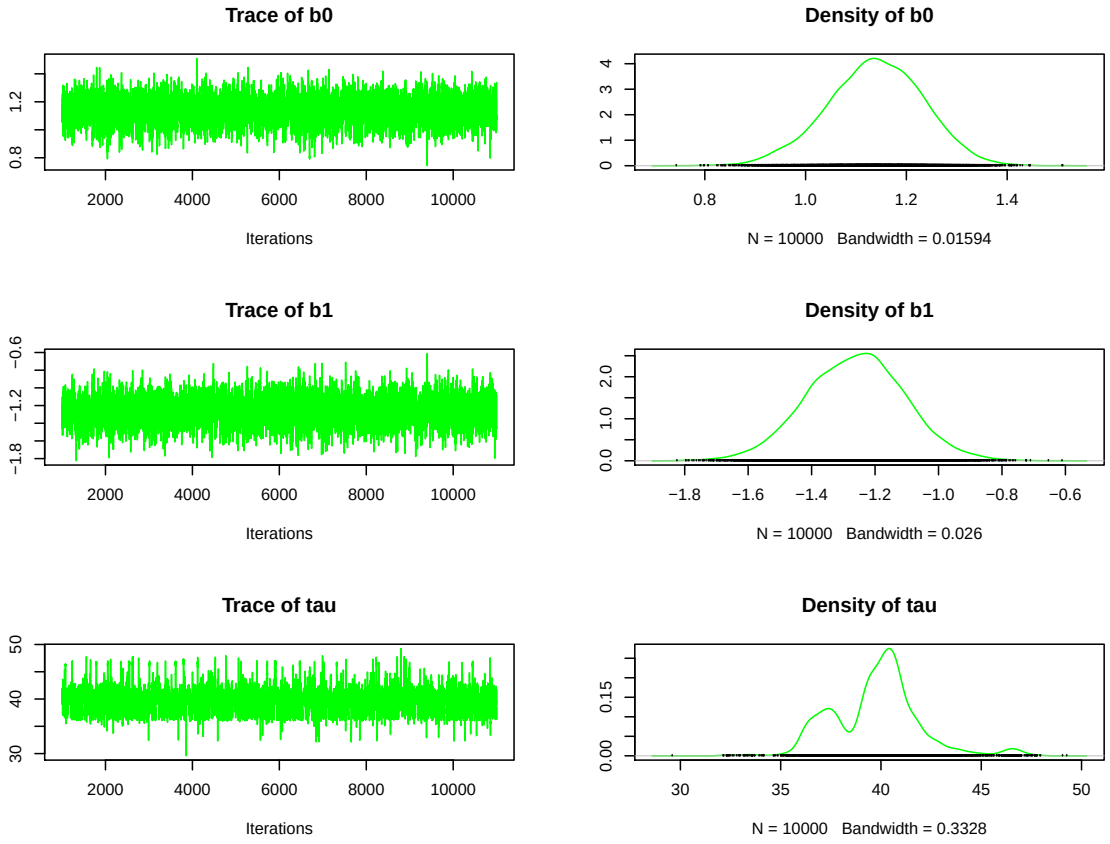
```
# 95% credible intervals of parameters
res$quantiles[,c(1,5)]
```

```
##           2.5%      97.5%
## b0   0.9647684  1.3175105
## b1  -1.5690670 -0.9713484
## tau 36.0801403 43.5016819
```

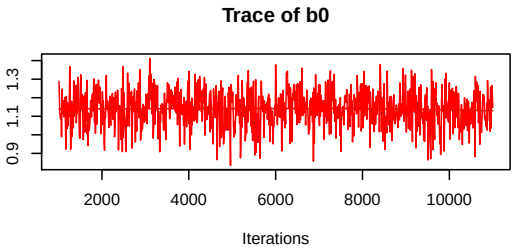
B) Explore the features of the chains and try to understand the effects of the burn-in and thinning.

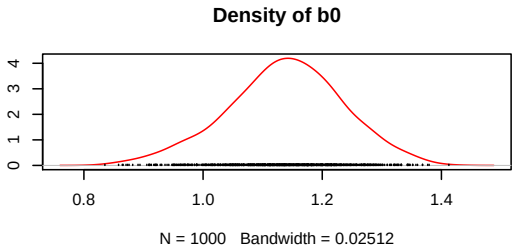
```
colors  <- c("red", "blue", "green")
burnins <- c(2000,3000,5000)
for(j in 1:3) {
    model.bug <- "model_coal_mine.bug"
    jag.mod <- jags.model(model.bug, data, init, quiet=TRUE)
    chain <- coda.samples(jag.mod, c("b0","b1","tau"), n.burnin=burnins[j], n.iter=10000)
    plot(chain, col=colors[j])
}
```

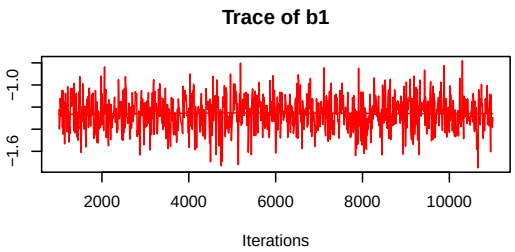

Trace of b0
Density of b0
Iterations
N = 10000 Bandwidth = 0.01575
Trace of b1
Density of b1
Iterations
N = 10000 Bandwidth = 0.02635
Trace of tau
Density of tau
Iterations
N = 10000 Bandwidth = 0.3559
Trace of b0
Density of b0
Iterations
N = 10000 Bandwidth = 0.01569
Trace of b1
Density of b1
Iterations
N = 10000 Bandwidth = 0.0261
Trace of tau
Density of tau
Iterations
N = 10000 Bandwidth = 0.3343

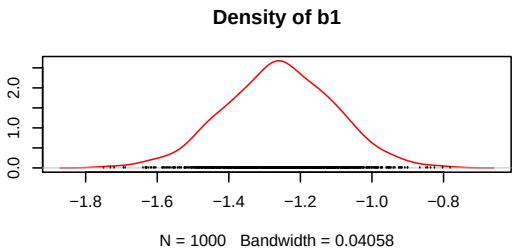


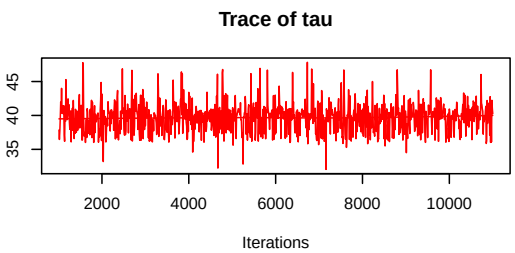

```
thins <- c(10,100,200)
for(j in 1:3) {
    model.bug <- "model_coal_mine.bug"
    jag.mod <- jags.model(model.bug, data, init, quiet=TRUE)
    chain <- coda.samples(jag.mod, c("b0","b1","tau"), n.iter=10000, thin=thins[j])
    plot(chain, col=colors[j])
}
```

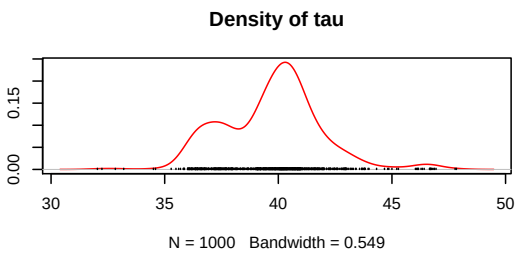

Trace of b0
0.9
1.1
1.3
2000
4000
6000
8000
10000
Iterations

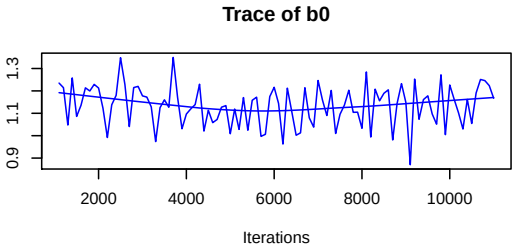

Density of b0
0
1
2
3
4
0.8
1.0
1.2
1.4
N = 1000 Bandwidth = 0.02512

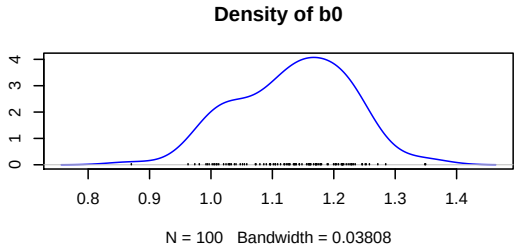

Trace of b1
−1.6
−1.0
2000
4000
6000
8000
10000
Iterations

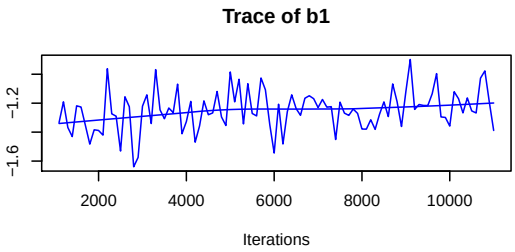

Density of b1
0.0
1.0
2.0
−1.8
−1.6
−1.4
−1.2
−1.0
−0.8
N = 1000 Bandwidth = 0.04058

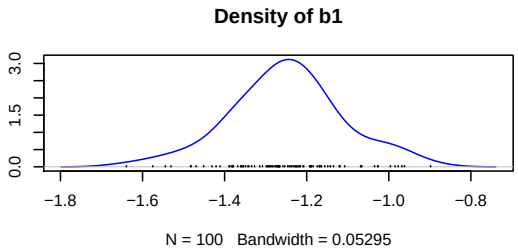

Trace of tau
35
40
45
2000
4000
6000
8000
10000
Iterations

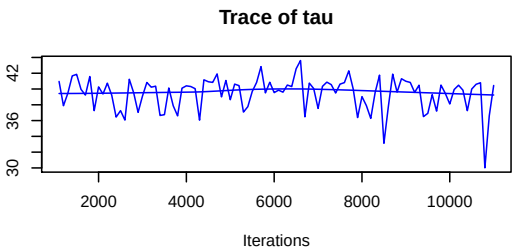

Density of tau
0.00
0.15
30
35
40
45
50
N = 1000 Bandwidth = 0.549

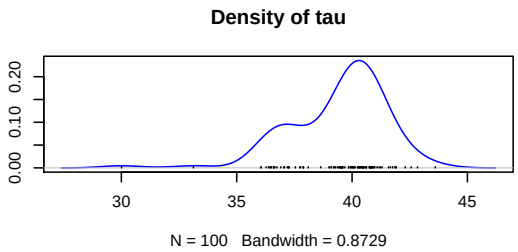

Trace of b0
0.9
1.1
1.3
2000
4000
6000
8000
10000
Iterations

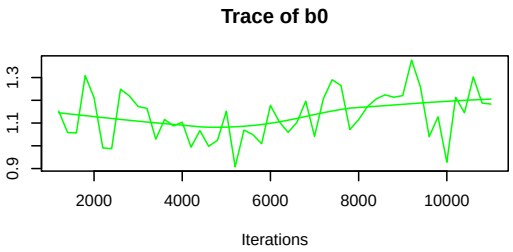

Density of b0
0
1
2
3
4
0.8
0.9
1.0
1.1
1.2
1.3
1.4
N = 100 Bandwidth = 0.03808

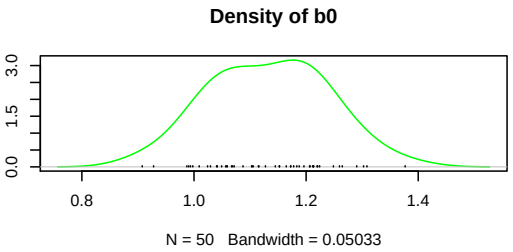

Trace of b1
−1.6
−1.2
2000
4000
6000
8000
10000
Iterations

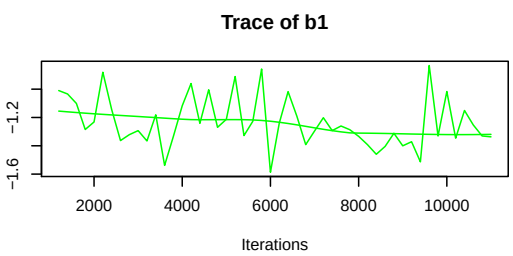

Density of b1
0.0
1.5
3.0
−1.8
−1.6
−1.4
−1.2
−1.0
−0.8
N = 100 Bandwidth = 0.05295

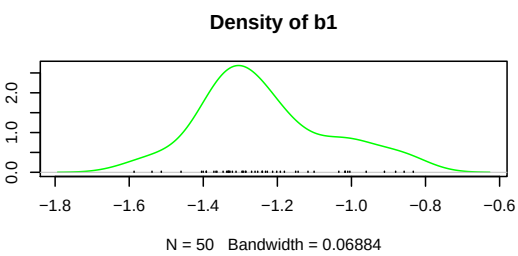

Trace of tau
30
36
42
2000
4000
6000
8000
10000
Iterations

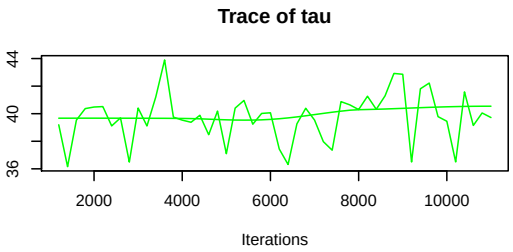

Density of tau
0.00
0.10
0.20
30
35
40
45
N = 100 Bandwidth = 0.8729

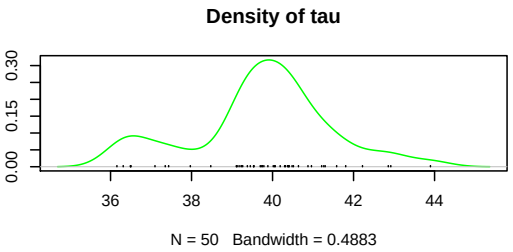

Trace of b0
0.9
1.1
1.3
2000
4000
6000
8000
10000
Iterations


Density of b0
0.0
1.5
3.0
0.8
1.0
1.2
1.4
N = 50 Bandwidth = 0.05033


Trace of b1
−1.6
−1.2
2000
4000
6000
8000
10000
Iterations


Density of b1
0.0
1.0
2.0
−1.8
−1.6
−1.4
−1.2
−1.0
−0.8
−0.6
N = 50 Bandwidth = 0.06884


Trace of tau
36
40
44
2000
4000
6000
8000
10000
Iterations


Density of tau
0.00
0.15
0.30
36
38
40
42
44
N = 50 Bandwidth = 0.4883