Statistical Methods for Data Science II

Final Project Report - JULY 2019

Bayesian Analysis using JAGS and MCMC on the classic example of Rats

Venkata Naga Sai Krishna Abhinay Pochiraju
Matricola: 1819771

Table of Contents

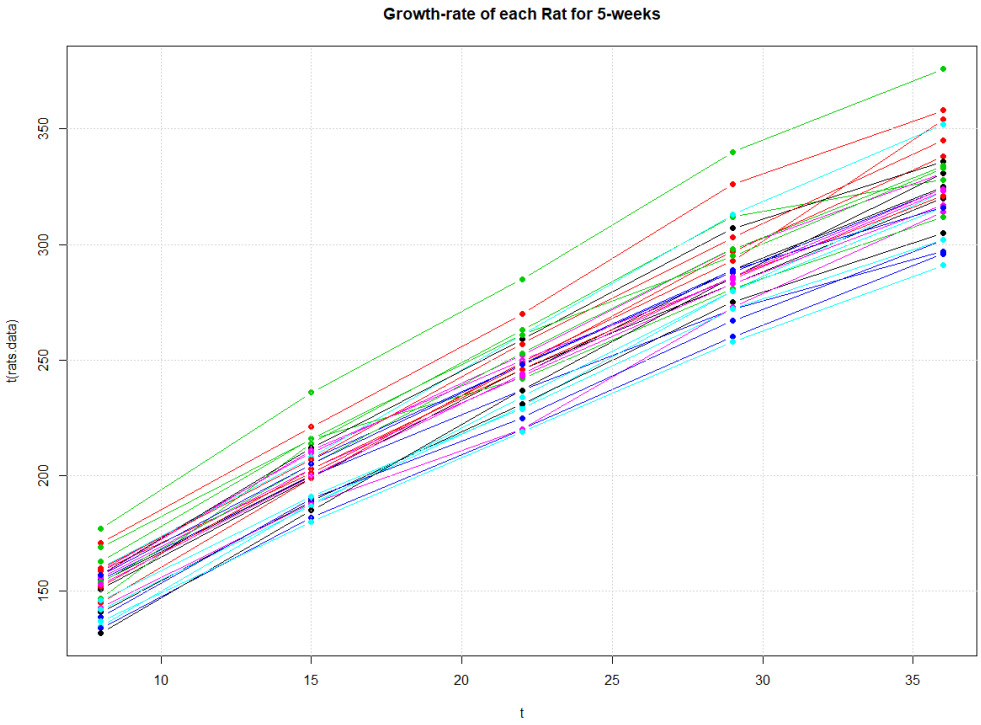
1. Data Description:

A classic example of rats given by Gelfand et al (1990) is taken in order to explain Bayesian inference using MCMC simulations. The Rats data is downloaded from https://people.maths.bris.ac.uk/~mazjcr/BMB/2016/home.html. This data is about 30 young rats whose weights are measured for 5 weeks. Our main goal is to find the initial weight of rats (at birth time) using Bayesian Analysis. First I download the csv file (rats.csv) from the source, read the csv into rats.data and plot the data.

```
# I download the data from
https://people.maths.bris.ac.uk/~mazjcr/BMB/2016/home.html
rats.data <- read.csv("D:\\Chrome Downloads\\SDS II\\Pochiraju\\Rats.csv",
sep = ",",encoding = "utf-8")
head(rats.data)
```

```
> head(rats.data)
   t1  t2  t3  t4  t5
1 151 199 246 283 320
2 145 199 249 293 354
3 147 214 263 312 328
4 155 200 237 272 297
5 135 188 230 280 323
6 159 210 252 298 331
```

```
matplot(t, t(rats.data), type = "b", pch = 16, lty = 1)
title(main = "Growth-rate of each Rat for 5-weeks")
grid()
```



Now, I initialized a list with this data, along with the age of the rat as x, data as Y, number of t's as T, N as 30 (the number of rats), and the average of x as xbar.

```
rats.data.raw <- list(x = c(8.0, 15.0, 22.0, 29.0, 36.0),
                      N = 30,
                      T = 5,
                      xbar=22,
                      Y = matrix(c(151, 199, 246, 283, 320,
                                   145, 199, 249, 293, 354,
                                   147, 214, 263, 312, 328,
                                   155, 200, 237, 272, 297,
                                   135, 188, 230, 280, 323,
                                   159, 210, 252, 298, 331,
                                   141, 189, 231, 275, 305,
                                   159, 201, 248, 297, 338,
                                   177, 236, 285, 350, 376,
                                   134, 182, 220, 260, 296,
                                   160, 208, 261, 313, 352,
                                   143, 188, 220, 273, 314,
                                   154, 200, 244, 289, 325,
                                   171, 221, 270, 326, 358,
                                   163, 216, 242, 281, 312,
                                   160, 207, 248, 288, 324,
                                   142, 187, 234, 280, 316,
                                   156, 203, 243, 283, 317,
                                   157, 212, 259, 307, 336,
                                   152, 203, 246, 286, 321,
                                   154, 205, 253, 298, 334,
                                   139, 190, 225, 267, 302,
                                   146, 191, 229, 272, 302,
                                   157, 211, 250, 285, 323,
                                   132, 185, 237, 286, 331,
                                   160, 207, 257, 303, 345,
                                   169, 216, 261, 295, 333,
                                   157, 205, 248, 289, 316,
                                   137, 180, 219, 258, 291,
                                   153, 200, 244, 286, 324),
                                 nrow=30, ncol=5, byrow=T))
```

```
Y <- rats.data.raw$Y
```

```
Y
```

```
> Y
      [,1] [,2] [,3] [,4] [,5]
 [1,]  151  199  246  283  320
 [2,]  145  199  249  293  354
 [3,]  147  214  263  312  328
 [4,]  155  200  237  272  297
 [5,]  135  188  230  280  323
 [6,]  159  210  252  298  331
 [7,]  141  189  231  275  305
 [8,]  159  201  248  297  338
 [9,]  177  236  285  350  376
[10,]  134  182  220  260  296
```

```
x <-rats.data.raw$x
x
```

```
 > x
[1]  8 15 22 29 36
```

```
xbar <- rats.data.raw$xbar
xbar
```

```
 > xbar
[1] 22
```

```
N <-rats.data.raw$N
N
```

```
 > N
[1] 30
```

```
T <- rats.data.raw$T
T
```

```
 > T
[1] 5
```

2. Defining the Model:

From the above graph we can say that each rat has its own separate line (with intercept and slope), but every line is following a common assumed distribution and it follows the following equation:

The model is essentially a random effects linear growth curve represented as:

$$Y_{ij} \sim N(\alpha_i + \beta_j x_j, \tau_c)$$

$$\alpha_i \sim N(\alpha_c, \tau_\alpha), \quad \beta_i \sim N(\beta_c, \tau_\beta)$$

Where τ = precision (inverse variance) of a Normal distribution. Our interest mainly focuses on finding the intercept at birth (zero-time) denoted by α_0

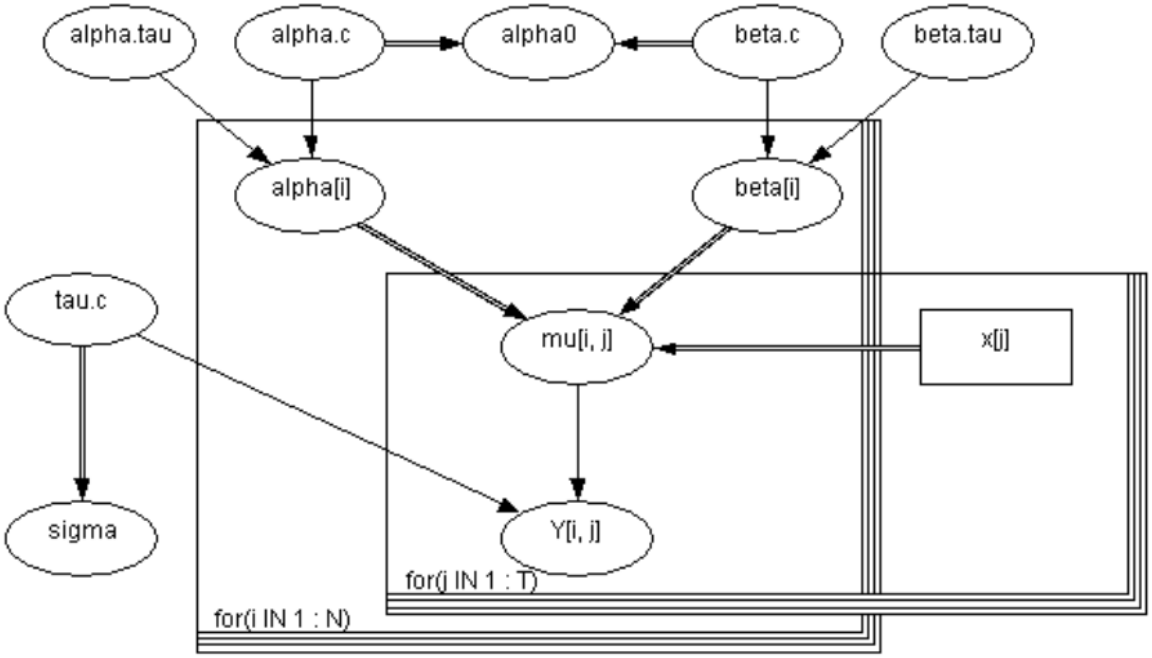
$$\alpha_0 = \alpha_c - \beta_c * \bar{x}$$

where $\alpha_c, \beta_c, \tau_\alpha, \tau_\beta, \tau_c$ are the given independent **non-informative priors** and $\bar{x} = xbar$ = average of x.

We must estimate the intercept and slope for each Rat then the initial weights of the rats are calculated using the above line.

The model for this dataset and the parameters we considered can be represented graphically as below [1]:

Graphical model for rats example:



BUGS *language for rats example:*

2.1 Frequentist Approach: The frequentist view is that the data is repeatable random sample or a random variable with a specific probability. Here, the underlying parameters and probabilities remain constant during this repeatable process.

```
# 2.1 Frequentist Approach

c <- rep(NA,N) #intercept
m <- rep(NA,N) #slope of the model
for(i in 1:N)
{
  lm.Frequentist <- lm(rats.data.raw$Y[i,] ~ rats.data.raw$x)
  c[i] <- lm.Frequentist$coefficients[[1]]
  m[i] <- lm.Frequentist$coefficients[[2]]
}
c
m
```

```
> c
 [1] 107.17143  87.08571 108.22857 120.31429  84.11429 114.22857  98.08571 10
5.91429 123.88571  92.05714 105.11429  93.40000
[13] 106.94286 118.65714 128.71429 116.85714  93.20000 114.05714 111.82857 10
9.28571 106.42857  97.94286 104.48571 117.60000
[25]  77.37143 107.94286 126.88571 116.65714  95.68571 106.88571
```

```
> m
 [1] 6.028571 7.314286 6.571429 5.085714 6.685714 6.171429 5.914286 6.485714
7.314286 5.742857 6.985714 6.100000 6.157143 6.842857
[15] 5.185714 5.842857 6.300000 5.742857 6.471429 6.014286 6.471429 5.757143
5.614286 5.800000 7.128571 6.657143 5.814286 5.742857
[29] 5.514286 6.114286
```

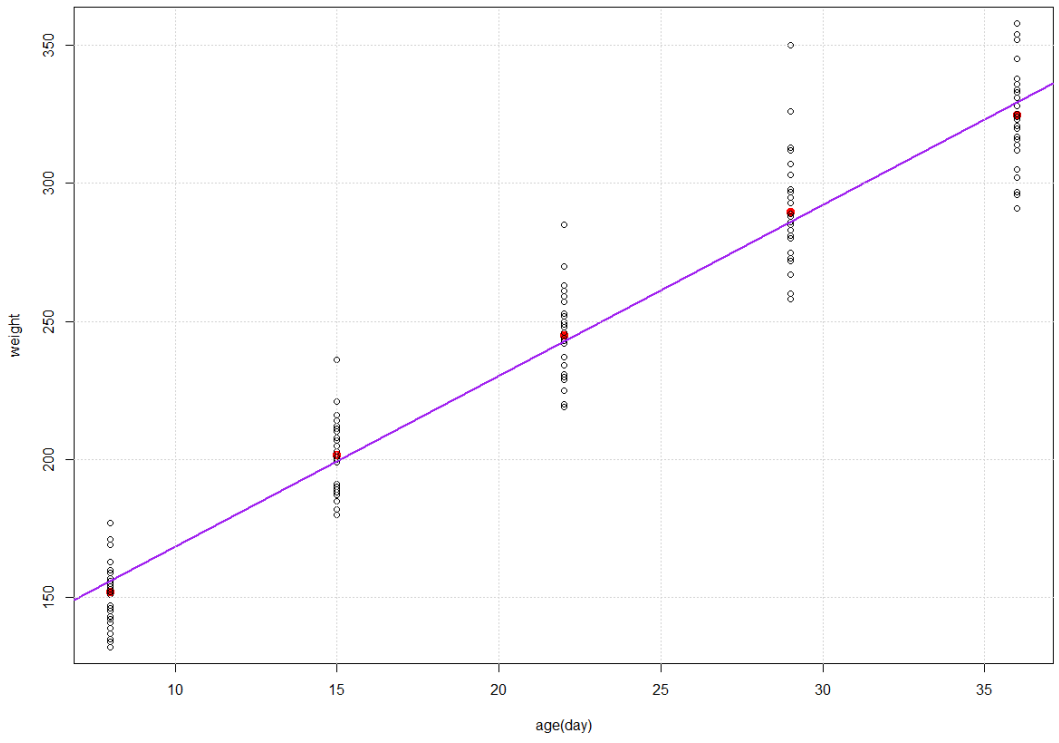
```
#mean of the intercept
mean.alpha <- mean(c)
mean.alpha
#106.5676
```

```
 > mean.alpha
[1] 106.5676
```

```
#mean of the slope
mean.beta <- mean(m)
mean.beta
#6.185714
```

```
 > mean.beta
[1] 6.185714
```

```
plot(rats.data.raw$x,colMeans(rats.data.raw$Y), lwd=4, xlab = "age(day)",
ylab = "weight",
     col="red", ylim=c(135,355))
points(rep(rats.data.raw$x[1],N), rats.data.raw$Y[,1])
points(rep(rats.data.raw$x[2],N), rats.data.raw$Y[,2])
points(rep(rats.data.raw$x[3],N), rats.data.raw$Y[,3])
points(rep(rats.data.raw$x[4],N), rats.data.raw$Y[,4])
points(rep(rats.data.raw$x[5],N), rats.data.raw$Y[,5])
abline(mean.alpha, mean.beta, col="purple", lwd=2)
grid()
```



2.2 Normal Hierarchical model with linear expected Value:

Step 1: Define the model and the priors

We know that prior independence doesn't force the posterior distributions to be independent. Also, as stated above we have taken the following non-informative independent priors for this model. They are as follows:

$$Y_{ij} \sim N(\alpha_i + \beta_j x_j, \tau_c)$$

$$\alpha_i \sim N(\alpha_c, \tau_\alpha), \quad \beta_i \sim N(\beta_c, \tau_\beta), \alpha_c \sim N(0,1.0E-6), \beta_c \sim N(0, 1.0E-6)$$

$$\tau_\alpha \sim Gamma(1.0E-3, 1.0E-3), \tau_\beta \sim Gamma(1.0E-3, 1.0E-3), \tau_c \sim Gamma(1.0E-3, 1.0E-3)$$

Here $\tau_\alpha, \tau_\beta, \tau_c$ are distributed as inverse Gamma.

The model is as follows:

```
# 2.2 Normal Hierarchial model with linear expected value

model1<- function()
{
  for (i in 1:N)
  {
    for (j in 1:T)
    {
      Y[i,j] ~ dnorm(mu[i,j], tau.c)
      mu[i, j] <- alpha[i] + beta[i] * (x[j])
    }
    alpha[i] ~ dnorm(alpha.c, tau.alpha)
    beta[i] ~ dnorm(beta.c, tau.beta)
  }
  #Priors
  alpha.c ~ dnorm(0, 1.0E-6)
  beta.c ~ dnorm(0, 1.0E-6)
  tau.c ~ dgamma(1.0E-3, 1.0E-3)
  tau.alpha ~ dgamma(1.0E-3, 1.0E-3)
  tau.beta ~ dgamma(1.0E-3, 1.0E-3)
  #Transformation
  sigma.c <- 1.0/sqrt(tau.c)
  xbar <- mean(x[])
  alpha0 <- alpha.c - beta.c*xbar
}
```

Step 2: As we are using JAGS (Just another Gibb's Sampling), first load the library

```
library(rjags)
```

Step 3: Read the Jags Parameters

```
# JAGS parameters
rats.params <- c("tau.c", "alpha.c", "beta.c", "tau.alpha", "tau.beta")
```

Step 4: Read the rats data for Jags

```
#read rats data for Jags
rats.data.jags.list <- list("Y", "x", "T", "N")
```

Step 5: Define Starting values for Jags

```
## Define the starting values for JAGS
rats.inits.1 <- function(){
  list("alpha"=c(250, 250, 250, 250, 250, 250, 250, 250, 250, 250, 250, 250,
250, 250, 250, 250, 250, 250, 250, 250, 250, 250, 250, 250, 250, 250,
250, 250, 250),
       "beta"=c(6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6,
6, 6, 6, 6, 6, 6, 6, 6, 6, 6),
       "alpha.c"= 150, "beta.c"= 10, "tau.c"= 1, "tau.alpha"= 1, "tau.beta"=
1)}
```

Step 6: Fitting the model created

```
## Fit the model in JAGS, having previously copied the BUGS model in my
working directory as "rats.model.jags"
ratsfit.model1 <- jags(data=rats.data.jags.list, inits=rats.inits.1,
rats.params, n.chains=2, n.iter=10000,
                       n.burnin=1000, n.thin = 1, model.file=model1,
DIC=TRUE)

ratsfit.model1
```

```
> ratsfit.model1
Inference for Bugs model at "C:/Users/rachu/AppData/Local/Temp/RtmpcRAx9A/model2ac61b17de3.txt", fit using jags,
 2 chains, each with 10000 iterations (first 1000 discarded)
 n.sims = 18000 iterations saved
          mu.vect sd.vect    2.5%     25%     50%     75%    97.5%  Rhat n.eff
alpha.c   106.554   2.314 102.024 105.040 106.552 108.055  111.063 1.001  7000
beta.c      6.186   0.106   5.979   6.116   6.186   6.256    6.397 1.001  4700
tau.alpha   0.010   0.004   0.005   0.007   0.010   0.012    0.020 1.001  4600
tau.beta    4.310   1.522   2.118   3.271   4.104   5.092    7.644 1.002  1000
tau.c       0.027   0.004   0.019   0.024   0.027   0.029    0.035 1.001 18000
deviance  969.974  14.690 943.663 959.596 969.212 979.223 1000.755 1.001 18000

For each parameter, n.eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor (at convergence, Rhat=1).

DIC info (using the rule, pD = var(deviance)/2)
pD = 107.9 and DIC = 1077.9
DIC is an estimate of expected predictive error (lower deviance is better).
```

```
summary(ratsfit.model1)
```

```
> summary(ratsfit.model1)
                  Length Class  Mode
model              8     jags   list
BUGSoutput        24     bugs   list
parameters.to.save  6     -none- character
```

```
model.file          1     -none- character
n.iter              1     -none- numeric
DIC                 1     -none- logical
```

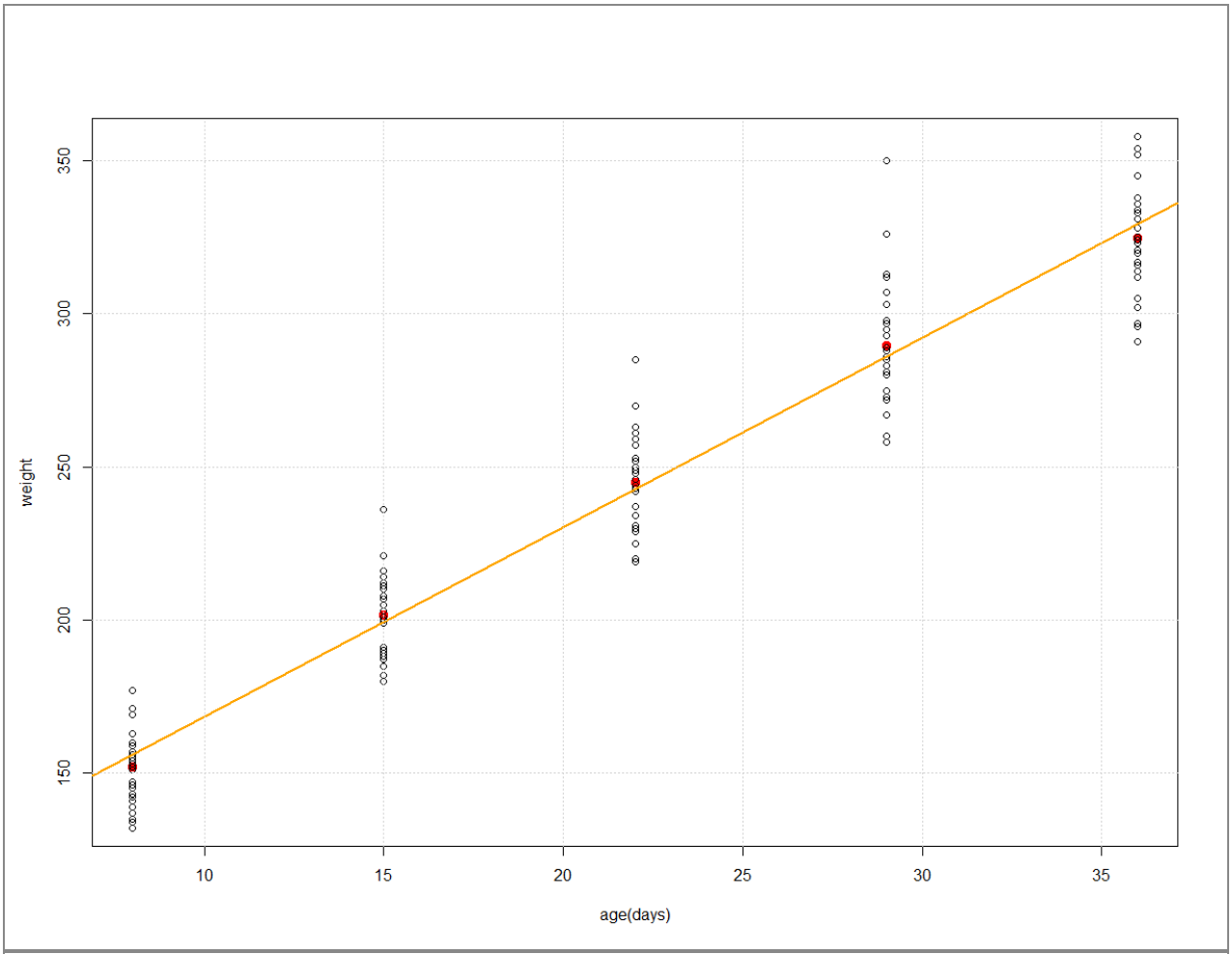
```
ratsfit.model1$BUGSoutput$DIC #lower is good
```

```
> ratsfit.model1$BUGSoutput$DIC #lower is good
[1] 1077.875
```

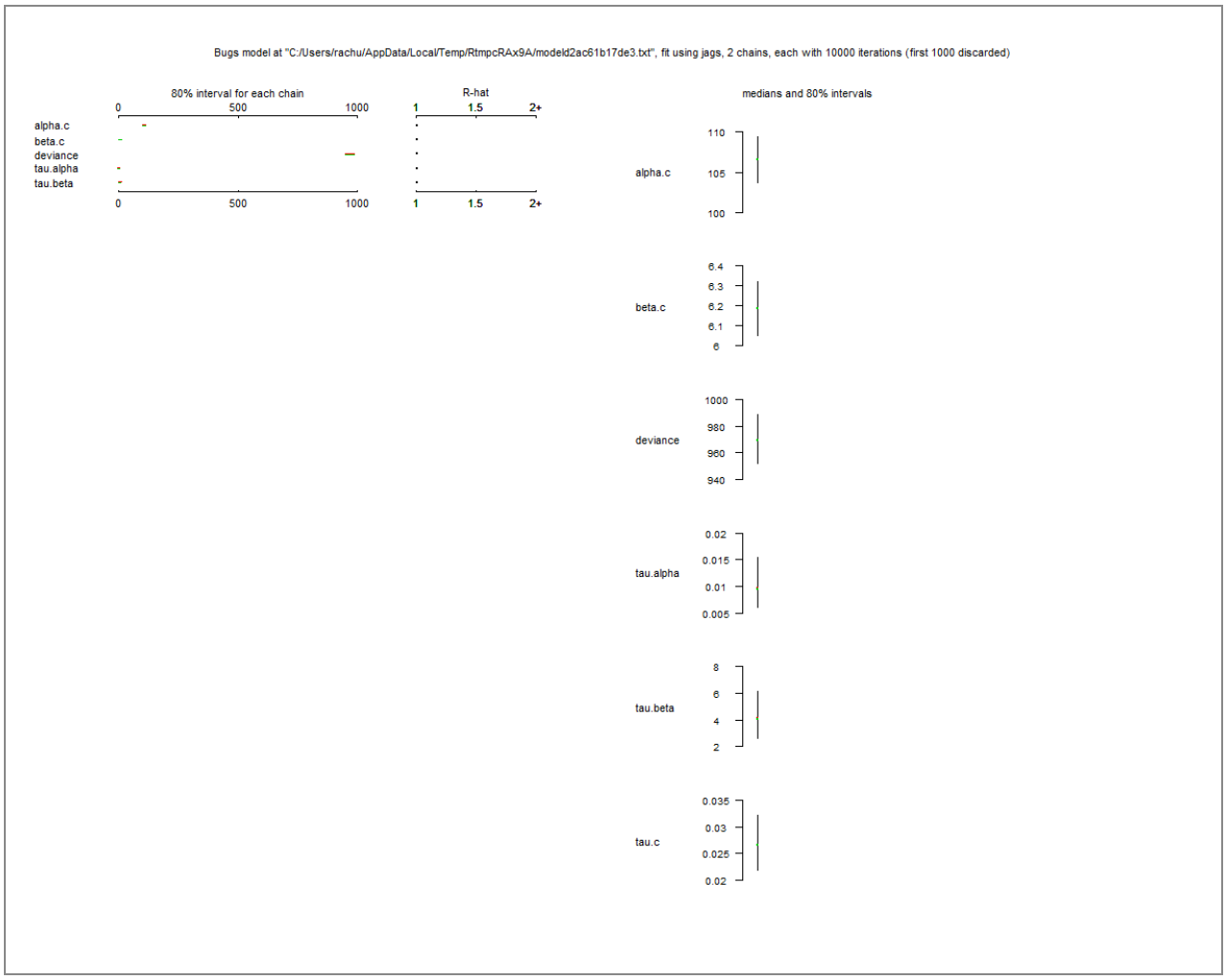
```
#now write the mean of theintercept for this model
ratsfit.model1$BUGSoutput$summary[,"mean"]["alpha.c"]

#intercept
ratsfit.model1$BUGSoutput$summary[,"mean"]["beta.c"]
```

```
 > ratsfit.model1$BUGSoutput$summary[,"mean"]["alpha.c"]
 alpha.c
106.5542
> ratsfit.model1$BUGSoutput$summary[,"mean"]["beta.c"]
  beta.c
6.186017
```

```
#plot the model1 just like frequentist model
plot(rats.data.raw$x,colMeans(rats.data.raw$Y), lwd=4, xlab = "age(days)",
ylab = "weight",
     col="red", ylim=c(135,355))
points(rep(rats.data.raw$x[1],N), rats.data.raw$Y[,1])
points(rep(rats.data.raw$x[2],N), rats.data.raw$Y[,2])
points(rep(rats.data.raw$x[3],N), rats.data.raw$Y[,3])
points(rep(rats.data.raw$x[4],N), rats.data.raw$Y[,4])
points(rep(rats.data.raw$x[5],N), rats.data.raw$Y[,5])
abline(ratsfit.model1$BUGSoutput$summary[,"mean"]["alpha.c"],
       ratsfit.model1$BUGSoutput$summary[,"mean"]["beta.c"], col="orange",
lwd=2)
grid()
```




```
#plot the model to see all the values
plot(ratsfit.model1)
```



2.3 Normal Hierarchical model with different priors tau.alpha and tau.beta

$$Y_{ij} \sim N(\alpha_i + \beta_j x_j, \tau_c)$$

$$\alpha_i \sim N(\alpha_c, \tau_\alpha), \quad \beta_i \sim N(\beta_c, \tau_\beta), \alpha_c \sim N(0,1.0E-6), \beta_c \sim N(0, 1.0E-6)$$

$$\tau_\alpha \sim Unif(0,100), \tau_\beta \sim Unif(0,100), \tau_c \sim Gamma(1.0E-3, 1.0E-3)$$

The model and the analysis as for the previous model, is as follows:

```
# 3. A normal Hierarchial model with different priors tau.alpha and tau.beta
model2 <- function()
{
  for (i in 1:N)
  {
    for (j in 1:T)
    {
      Y[i,j] ~ dnorm(mu[i,j], tau.c)
      mu[i, j] <- alpha[i] + beta[i] * (x[j])
    }
    alpha[i] ~ dnorm(alpha.c, tau.alpha)
```

```
    beta[i] ~ dnorm(beta.c, tau.beta)
  }
  alpha.c ~ dnorm(0, 1.0E-6)
  beta.c ~ dnorm(0, 1.0E-6)
  tau.c ~ dgamma(1.0E-3, 1.0E-3)
  sigma.alpha ~ dunif(0,100)
  sigma.beta ~ dunif(0,100)
  tau.alpha <- 1/(sigma.alpha*sigma.alpha)
  tau.beta <- 1/(sigma.beta*sigma.beta)
  sigma.c <- 1.0/sqrt(tau.c)
  xbar <- mean(x[])
  alpha0 <- alpha.c - beta.c*xbar
}
## Read in the rats data for JAGS
rats.data.list <- list("Y", "x", "T", "N")
## Name the JAGS parameters
rats.params <- c("tau.c", "alpha.c", "beta.c", "tau.alpha", "tau.beta")
#### Define the starting values for JAGS
rats.inits.2 <- function(){
  list(alpha = c(250, 250, 250, 250, 250, 250, 250, 250, 250, 250, 250, 250,
250, 250, 250,
                 250, 250, 250, 250, 250, 250, 250, 250, 250, 250, 250, 250,
250, 250, 250),
       beta = c(6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6,
                6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6),
       alpha.c = 150, beta.c = 10,
       tau.c = 1, sigma.alpha = 1, sigma.beta = 1)
}
ratsfit.model2 <- jags(data=rats.data.list, inits=rats.inits.2, rats.params,
n.chains=2, n.iter=10000,
             n.burnin=1000, n.thin = 1, model.file=model2, DIC=TRUE)
```

```
> ratsfit.model2 <- jags(data=rats.data.list, inits=rats.inits.2, rats.params
, n.chains=2, n.iter=10000,
+              n.burnin=1000, n.thin = 1, model.file=model2, DIC=TRUE)
Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 150
   Unobserved stochastic nodes: 65
   Total graph size: 537

Initializing model

  |++++++++++++++++++++++++++++++++++++++++++++++++++| 100%
  |**************************************************| 100%
```

```
ratsfit.model2

> ratsfit.model2

Inference for Bugs model at
"C:/Users/rachu/AppData/Local/Temp/RtmpcRAx9A/modeld2ac484372bd.txt", fit
using jags,

 2 chains, each with 10000 iterations (first 1000 discarded)
```

```
 n.sims = 18000 iterations saved

          mu.vect sd.vect    2.5%     25%     50%     75%   97.5%  Rhat n.eff

alpha.c   106.576   2.367 101.899 105.003 106.570 108.146 111.174 1.001 18000

beta.c      6.186   0.107   5.976   6.116   6.186   6.258   6.397 1.001  5200

tau.alpha   0.009   0.004   0.004   0.007   0.009   0.011   0.018 1.001  3200

tau.beta    4.132   1.431   2.001   3.131   3.923   4.877   7.527 1.002  1300

tau.c       0.027   0.004   0.020   0.024   0.027   0.030   0.036 1.001  7900

deviance  968.434  14.376 942.247 958.361 967.678 977.564 998.837 1.001  3900



For each parameter, n.eff is a crude measure of effective sample size,

and Rhat is the potential scale reduction factor (at convergence, Rhat=1).



DIC info (using the rule, pD = var(deviance)/2)

pD = 103.3 and DIC = 1071.7

DIC is an estimate of expected predictive error (lower deviance is better).
```

```
ratsfit.model2$BUGSoutput$DIC #lower is good
```

```
> ratsfit.model2$BUGSoutput$DIC #lower is good
[1] 1071.746
```

```
#now write the mean of theintercept for this model
ratsfit.model2$BUGSoutput$summary[,"mean"]["alpha.c"]
```

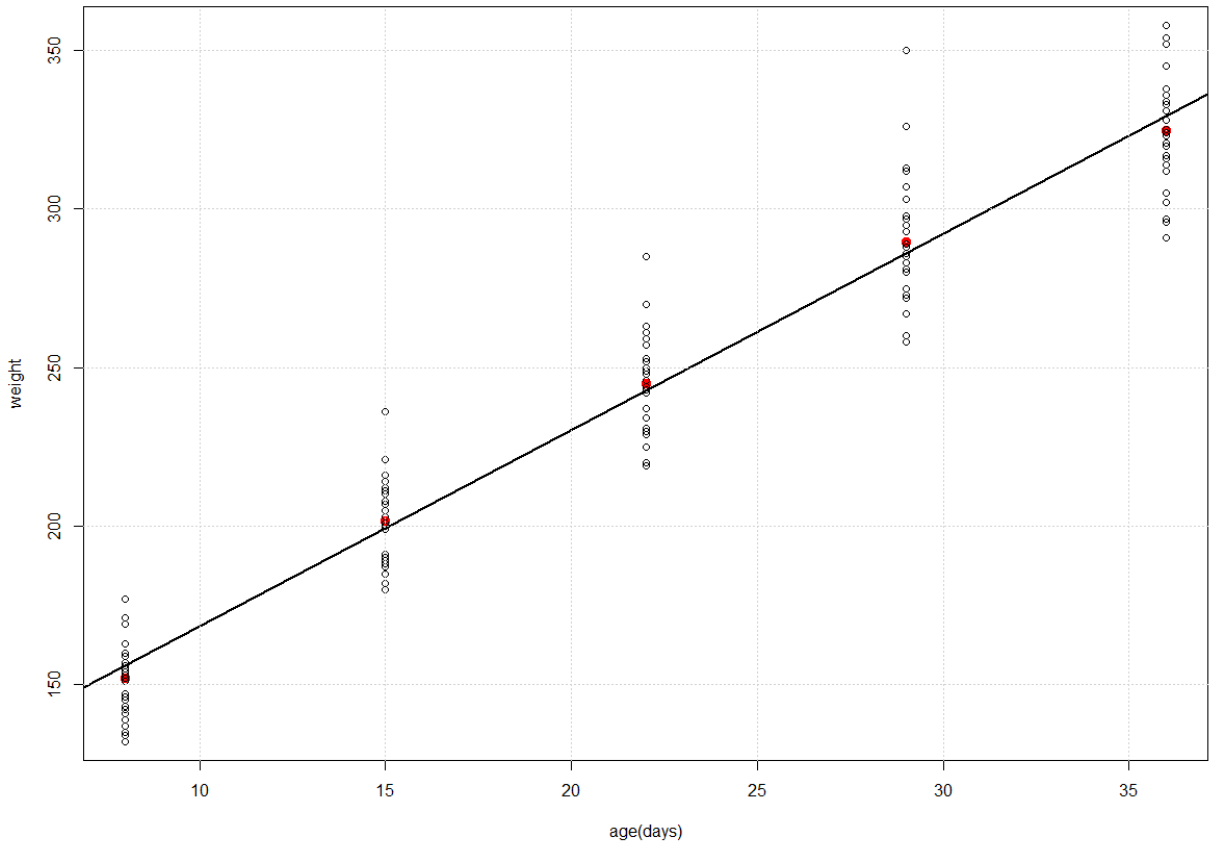
```
> ratsfit.model2$BUGSoutput$summary[,"mean"]["alpha.c"]
 alpha.c
106.5762
```

```
#intercept
ratsfit.model2$BUGSoutput$summary[,"mean"]["beta.c"]
```

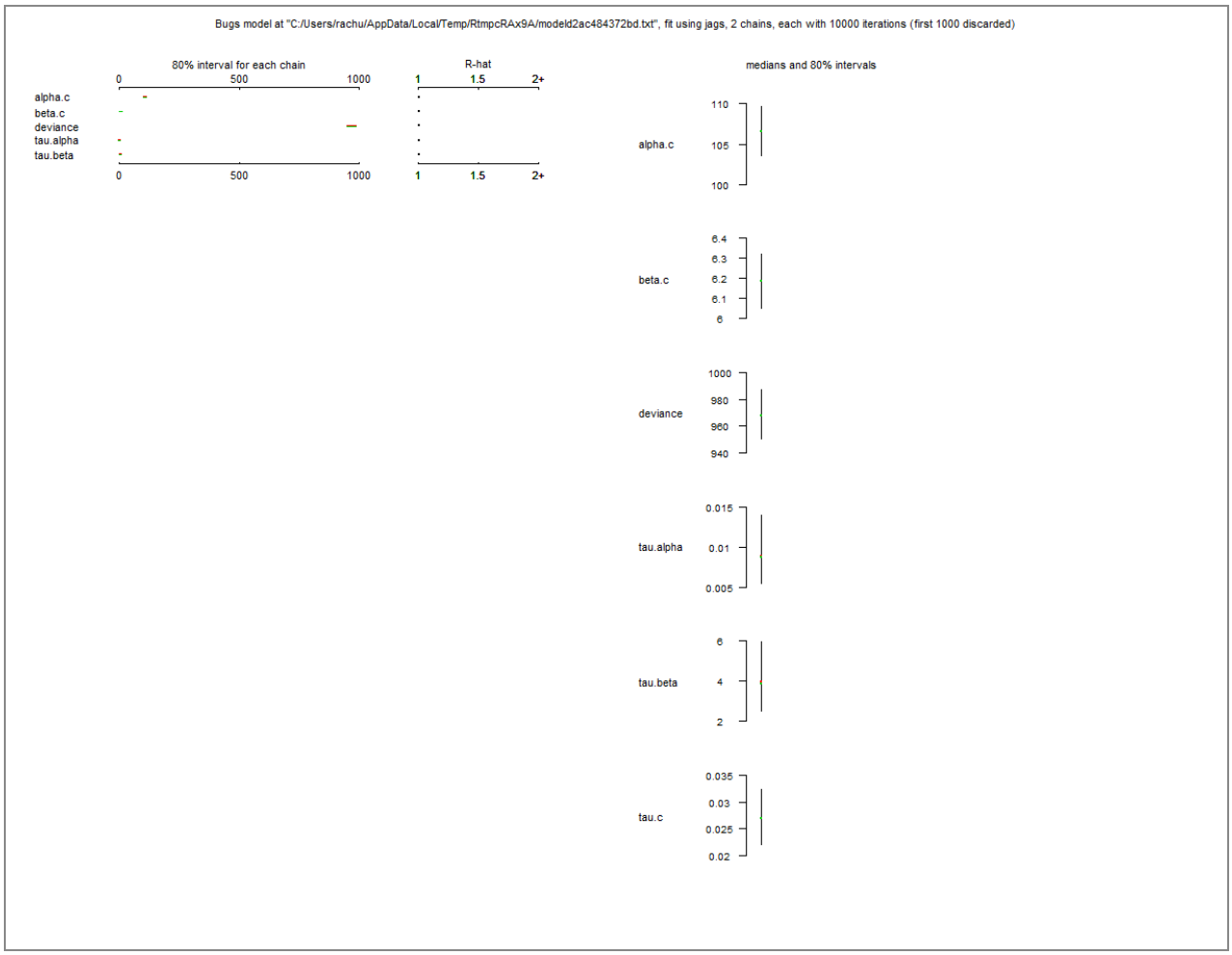
```
> ratsfit.model2$BUGSoutput$summary[,"mean"]["beta.c"]
  beta.c
6.186231
```

```
#plot the model1 just like frequentist model

plot(rats.data.raw$x,colMeans(rats.data.raw$Y), lwd=4, xlab = "age(days)",
ylab = "weight",
     col="red", ylim=c(135,355))
points(rep(rats.data.raw$x[1],N), rats.data.raw$Y[,1])
points(rep(rats.data.raw$x[2],N), rats.data.raw$Y[,2])
points(rep(rats.data.raw$x[3],N), rats.data.raw$Y[,3])
points(rep(rats.data.raw$x[4],N), rats.data.raw$Y[,4])
```

```
points(rep(rats.data.raw$x[5],N), rats.data.raw$Y[,5])
abline(ratsfit.model2$BUGSoutput$summary[,"mean"]["alpha.c"],
       ratsfit.model2$BUGSoutput$summary[,"mean"]["beta.c"], col="black",
lwd=2)
grid()
```




```
#plot the model to see all the values
plot(ratsfit.model2)
```



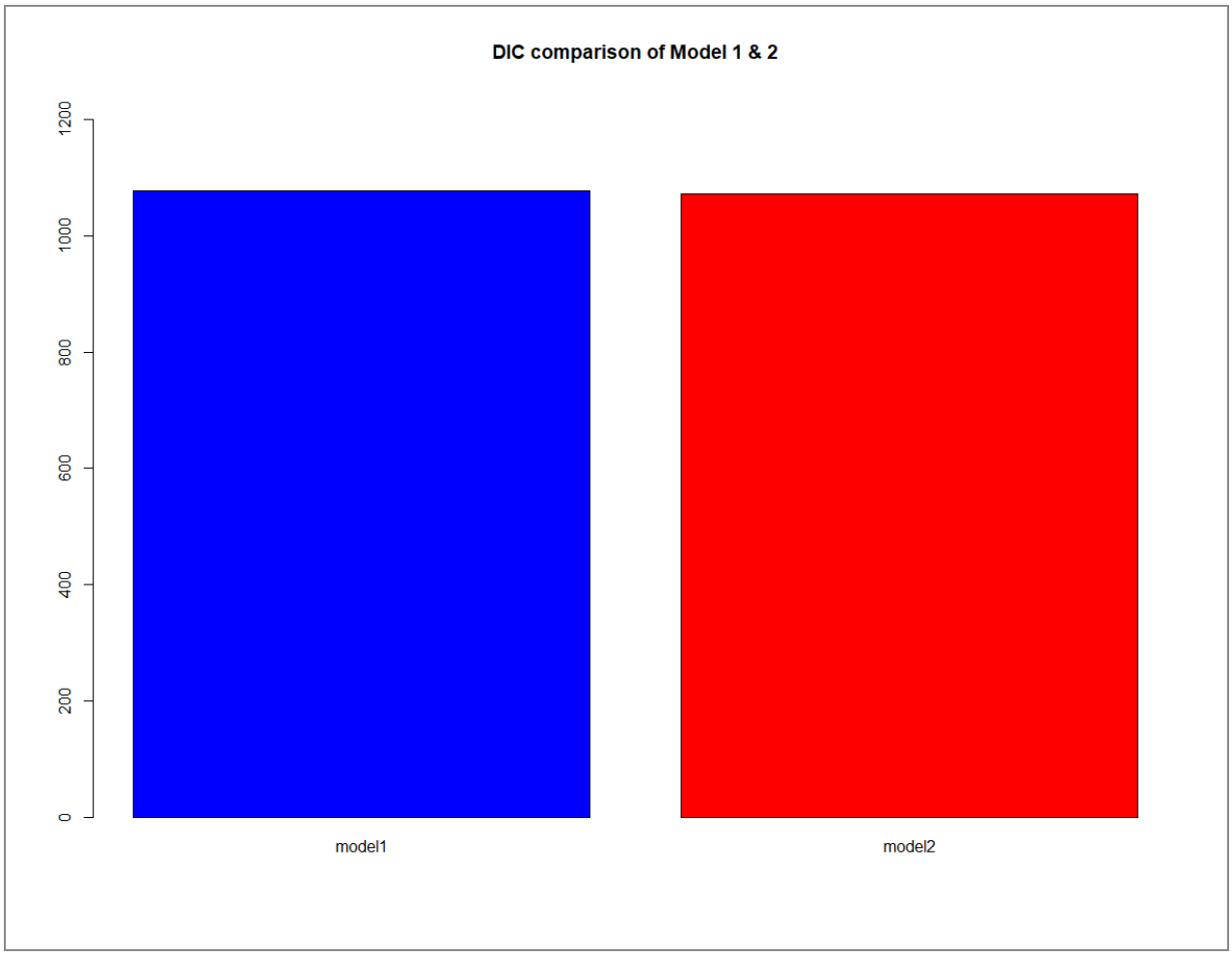
3. Model Comparison with DIC (Deviance Information Criterion)

Now I compared the DIC values for both the models to evaluate the models to find the best model.

```
# Compare the 2 models using DIC
DIC.Models <- cbind(model1.DIC=ratsfit.model1$BUGSoutput$DIC,
                    model2.DIC=ratsfit.model2$BUGSoutput$DIC)

DIC.Models
```

```
> DIC.Models
     model1.DIC model2.DIC
[1,]   1077.875   1071.746
```

```
barplot(c(ratsfit.model1$BUGSoutput$DIC, ratsfit.model2$BUGSoutput$DIC),
col=c("blue", "red"), main="DIC comparison of Model 1 & 2",
        names.arg = c("model1", "model2"), ylim=c(0,1250))
```

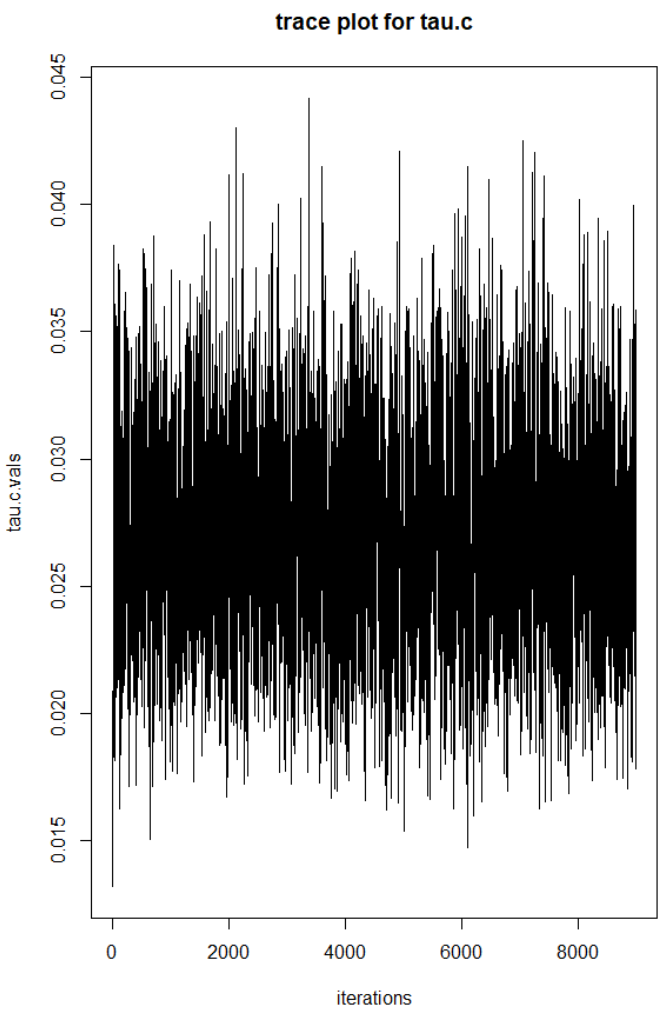


We can see that the DIC values for both the models are almost similar.

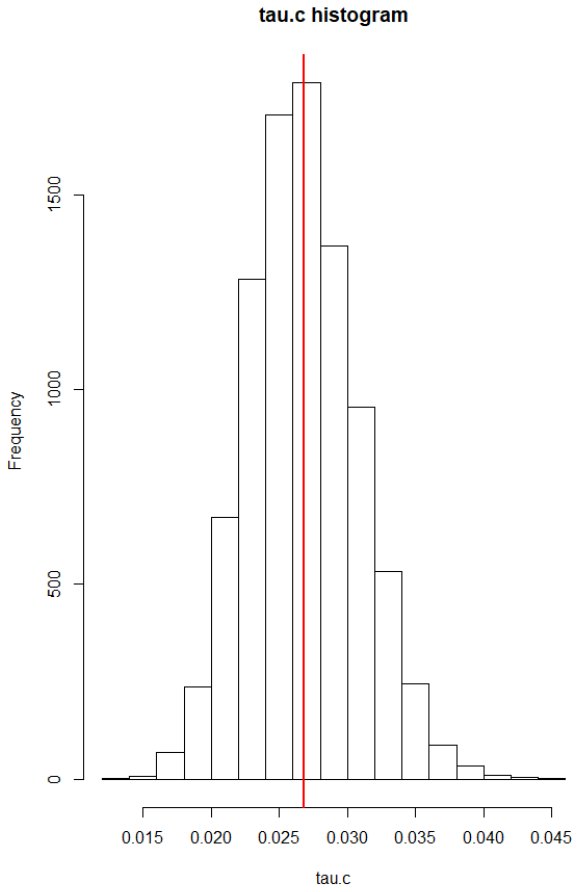
4. MCMC Analysis

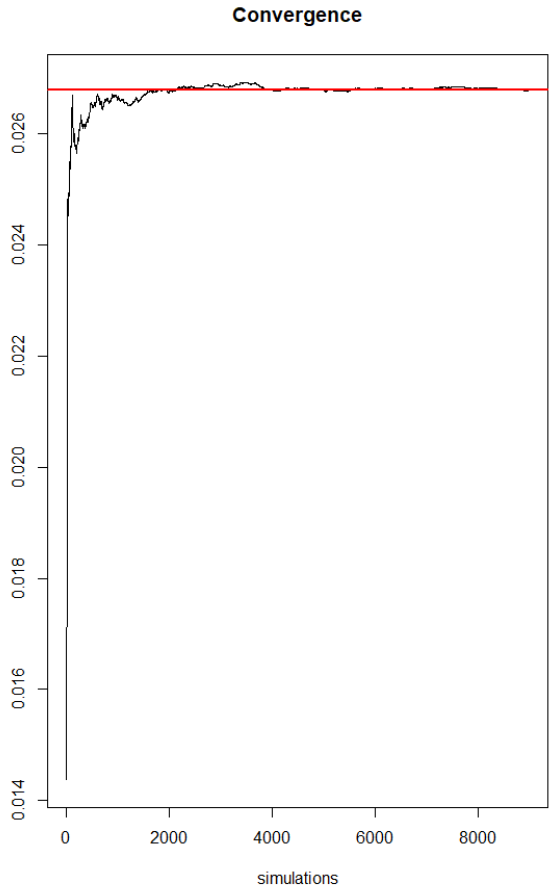
An essential step in MCMC is verifying the convergence. Bayesian Inference based on MCMC sampling is valid if and only if the Markov chain has convergence. We have to verify convergence for all the parameters, we can do this by plotting trace plots, histograms and cumsum plots for the parameters.

```
# Convergence tests
#tau.c
tau.c.vals <- ratsfit.model1$BUGSoutput$sims.array[,1,"tau.c"]
plot(tau.c.vals, xlab = "iterations", main="trace plot for tau.c",type="l")
```

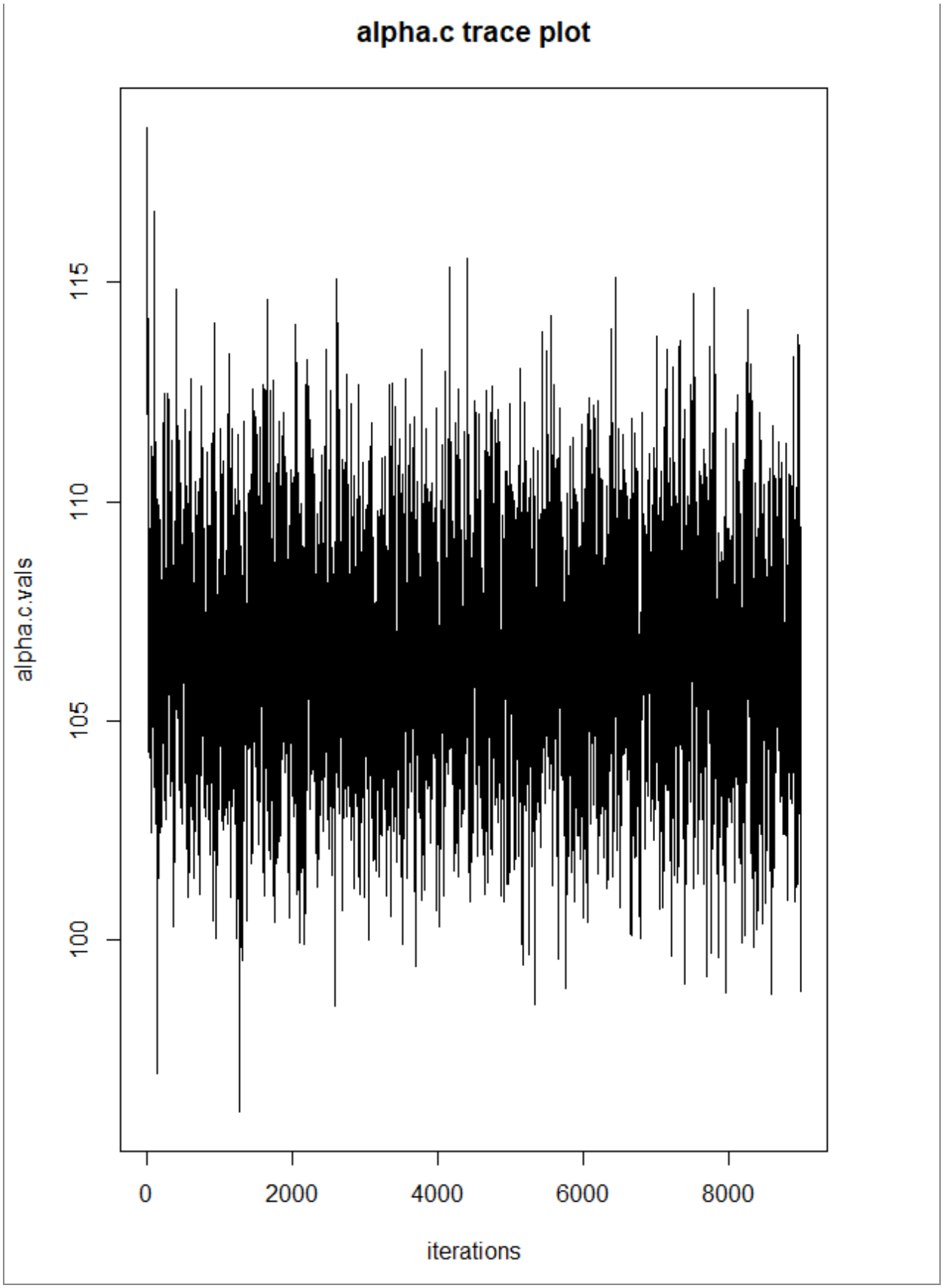



```
par(mfrow=c(1,2))
hist(tau.c.vals, main= "tau.c histogram", xlab = "tau.c")
abline(v=mean(tau.c.vals), col="red", lwd=2)
plot(cumsum(tau.c.vals)/(1:length(tau.c.vals)), type="l", ylab="",
     main="behaviour empirical average", xlab="simulations")
abline(h=mean(tau.c.vals), col="red", lwd=2)
```

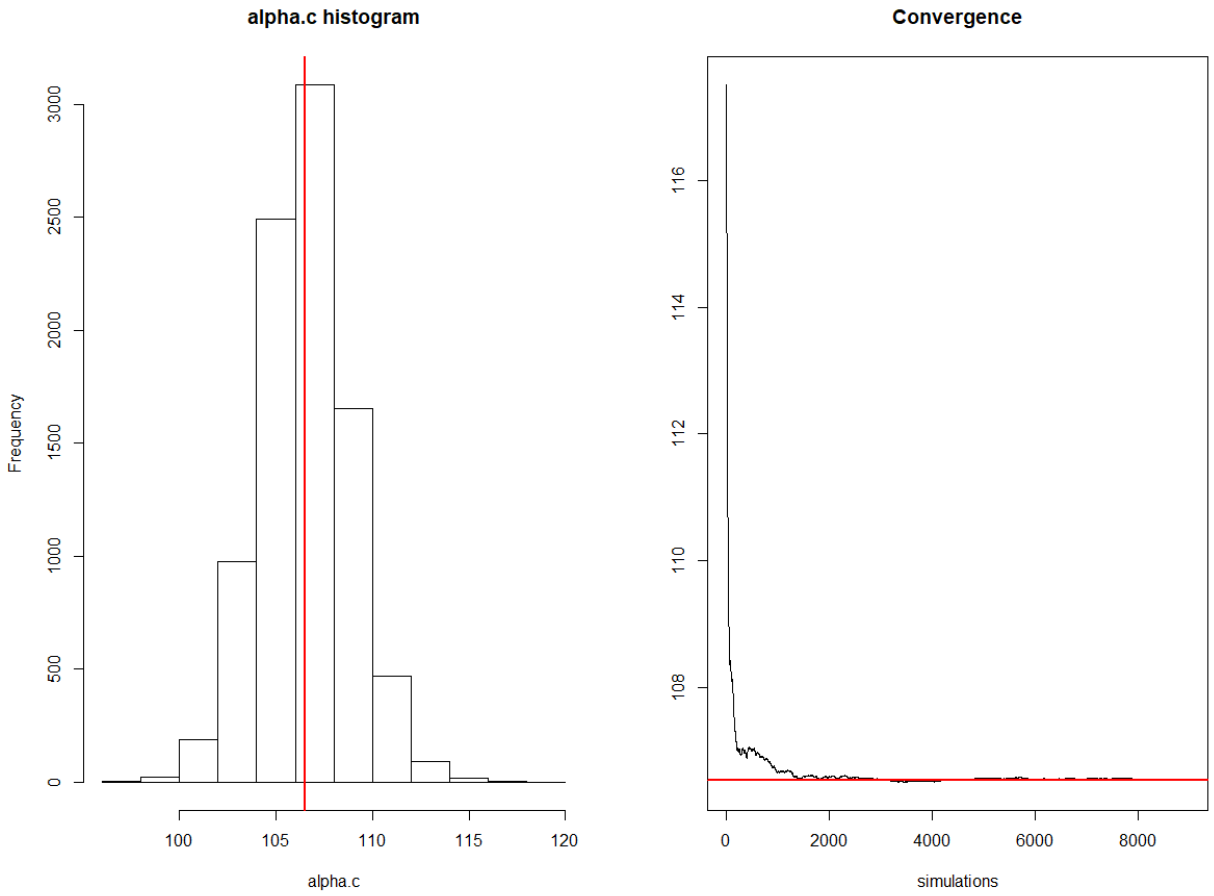




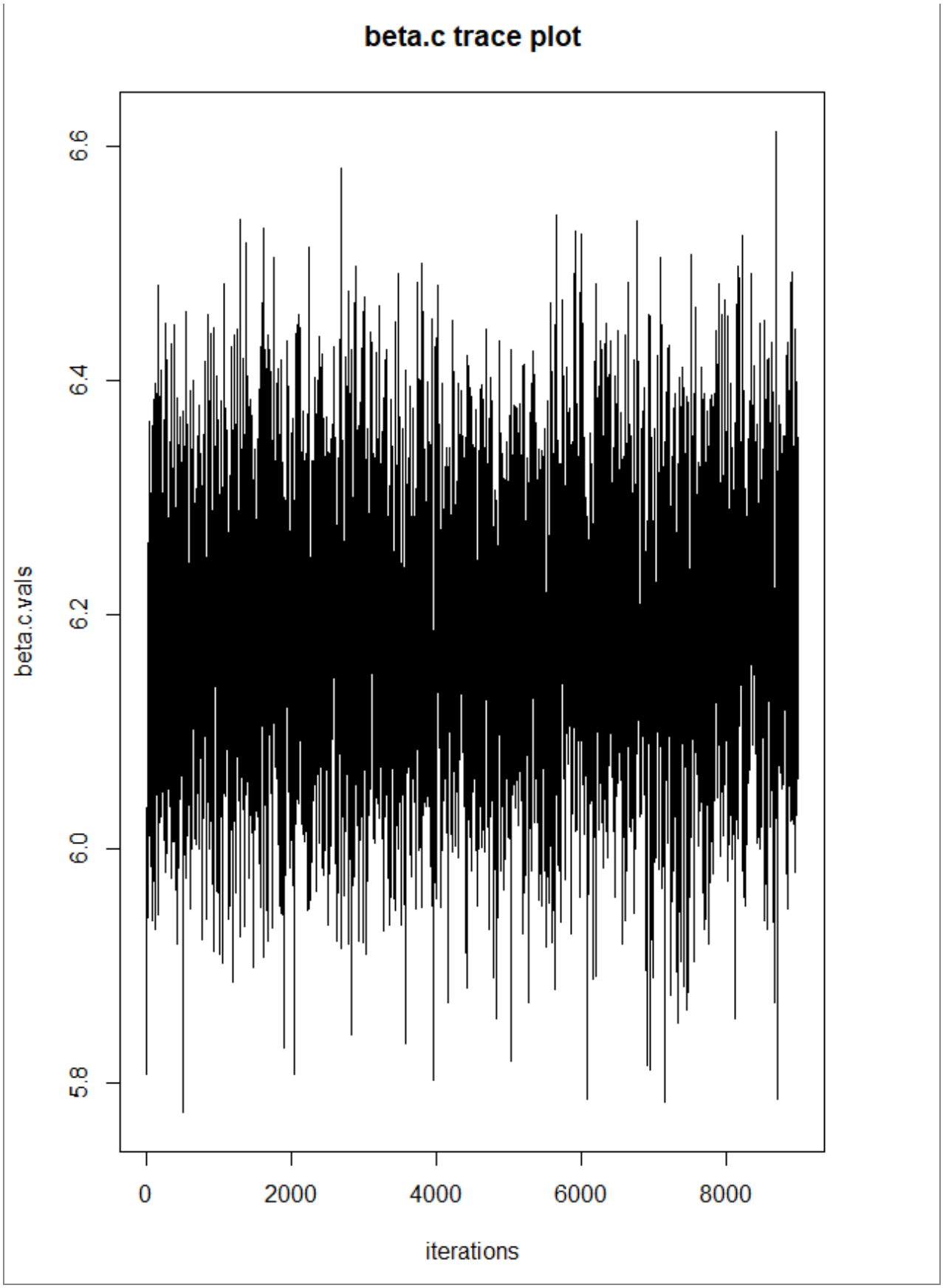

```
#alpha.c

alpha.c.vals <- ratsfit.model1$BUGSoutput$sims.array[,1,"alpha.c"]
plot(alpha.c.vals, xlab = "iterations", main="alpha.c trace plot",type="l")
```


alpha.c trace plot
alpha.c.vals
100
105
110
115
0
2000
4000
6000
8000
iterations

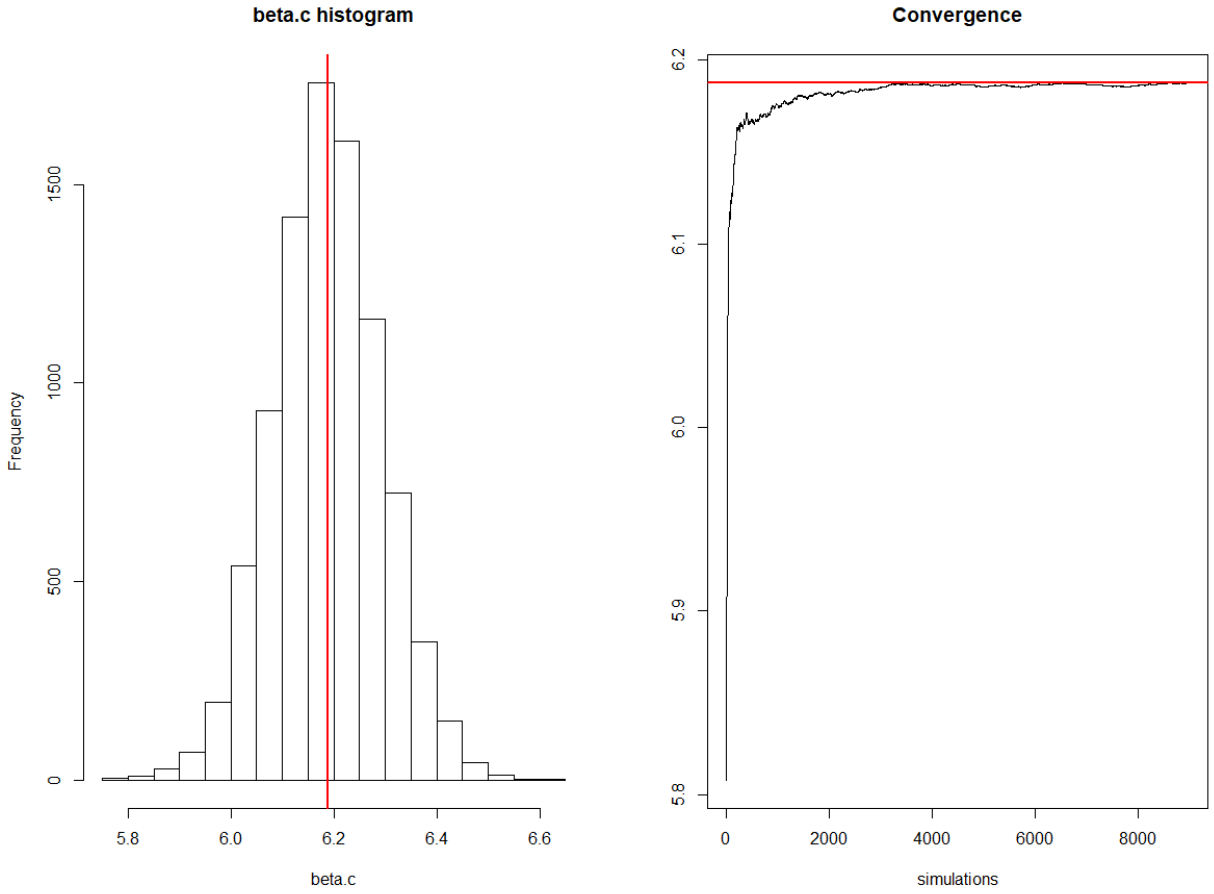
```
par(mfrow=c(1,2))
hist(alpha.c.vals, main= "alpha.c histogram", xlab = "alpha.c")
abline(v=mean(alpha.c.vals), col="red", lwd=2)
plot(cumsum(alpha.c.vals)/(1:length(alpha.c.vals)), type="l", ylab="",
     main="Convergence", xlab="simulations")
abline(h=mean(alpha.c.vals), col="red", lwd=2)
```



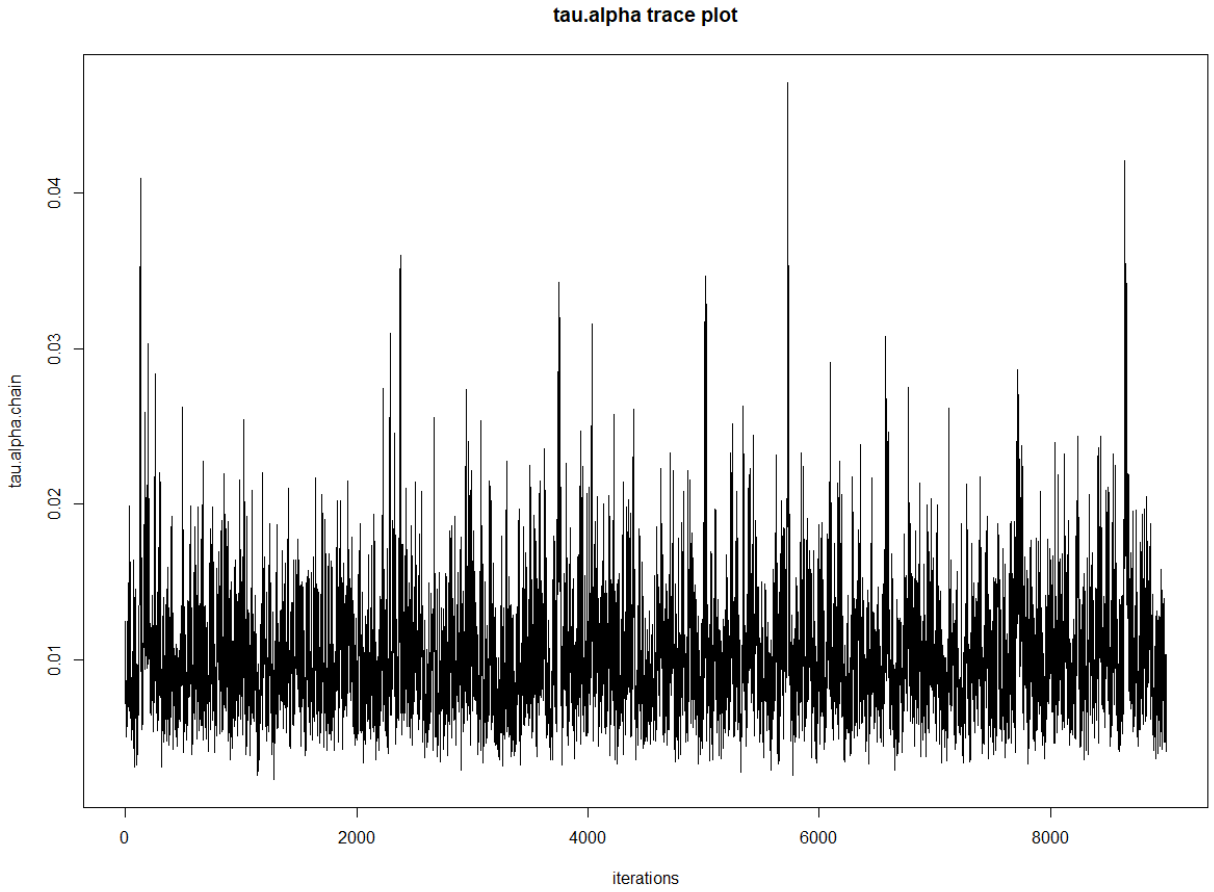

```
 #beta.c
beta.c.vals <- ratsfit.model1$BUGSoutput$sims.array[,1,"beta.c"]

plot(beta.c.vals, xlab = "iterations", main="beta.c trace plot",type="l")
```


beta.c trace plot
beta.c.vals
6.6
6.4
6.2
6.0
5.8
0
2000
4000
6000
8000
iterations

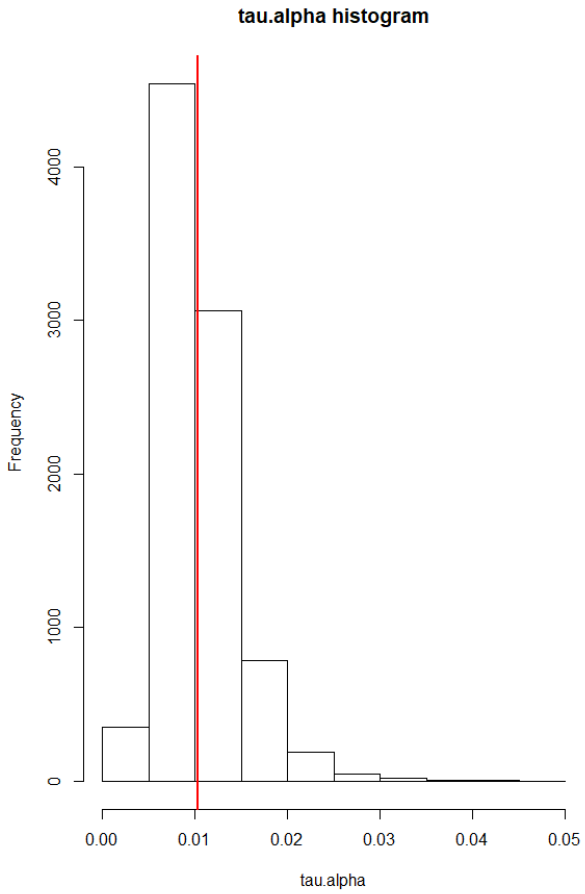
```
 par(mfrow=c(1,2))

hist(beta.c.vals, main= "beta.c histogram", xlab = "beta.c")
abline(v=mean(beta.c.vals), col="red", lwd=2)

plot(cumsum(beta.c.vals)/(1:length(beta.c.vals)), type="l", ylab="",
     main="Convergence", xlab="simulations")
abline(h=mean(beta.c.vals), col="red", lwd=2)
```

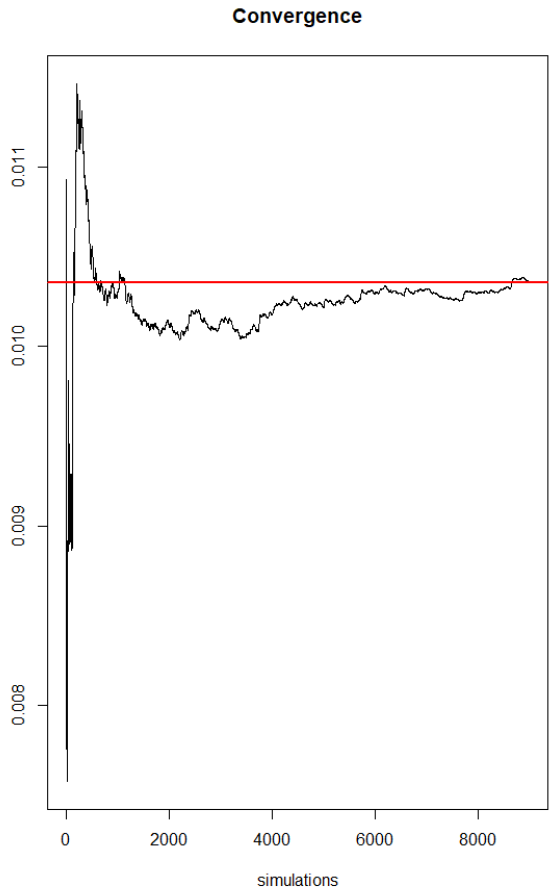



```
 #tau.alpha
tau.alpha.chain <- ratsfit.model1$BUGSoutput$sims.array[,1,"tau.alpha"]
plot(tau.alpha.chain, xlab = "iterations", main="tau.alpha trace
plot",type="l")
```

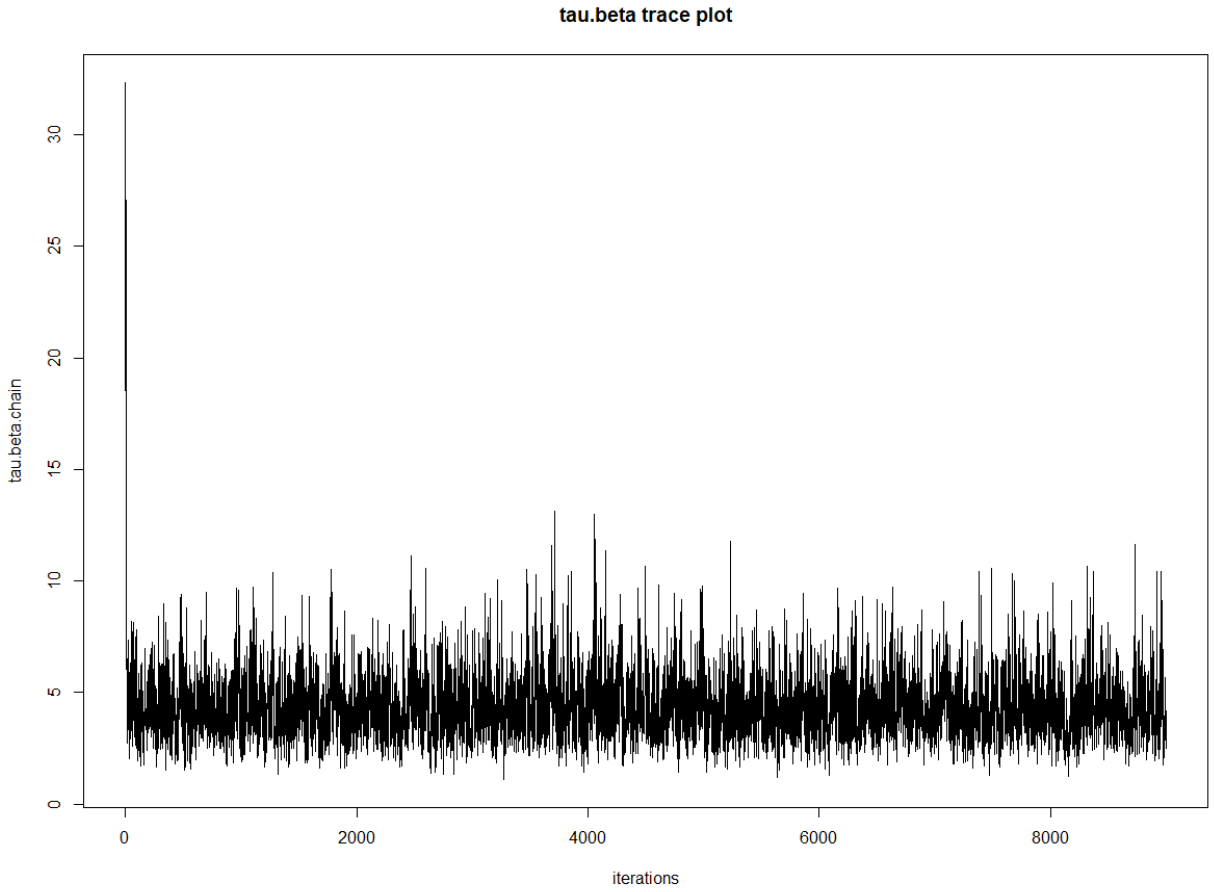



```
 par(mfrow=c(1,2))
hist(tau.alpha.vals, main= "tau.alpha histogram", xlab = "tau.alpha")
abline(v=mean(tau.alpha.vals), col="red", lwd=2)

plot(cumsum(tau.alpha.vals)/(1:length(tau.alpha.vals)), type="l", ylab="",
     main="Convergence", xlab="simulations")
abline(h=mean(tau.alpha.vals), col="red", lwd=2)
```

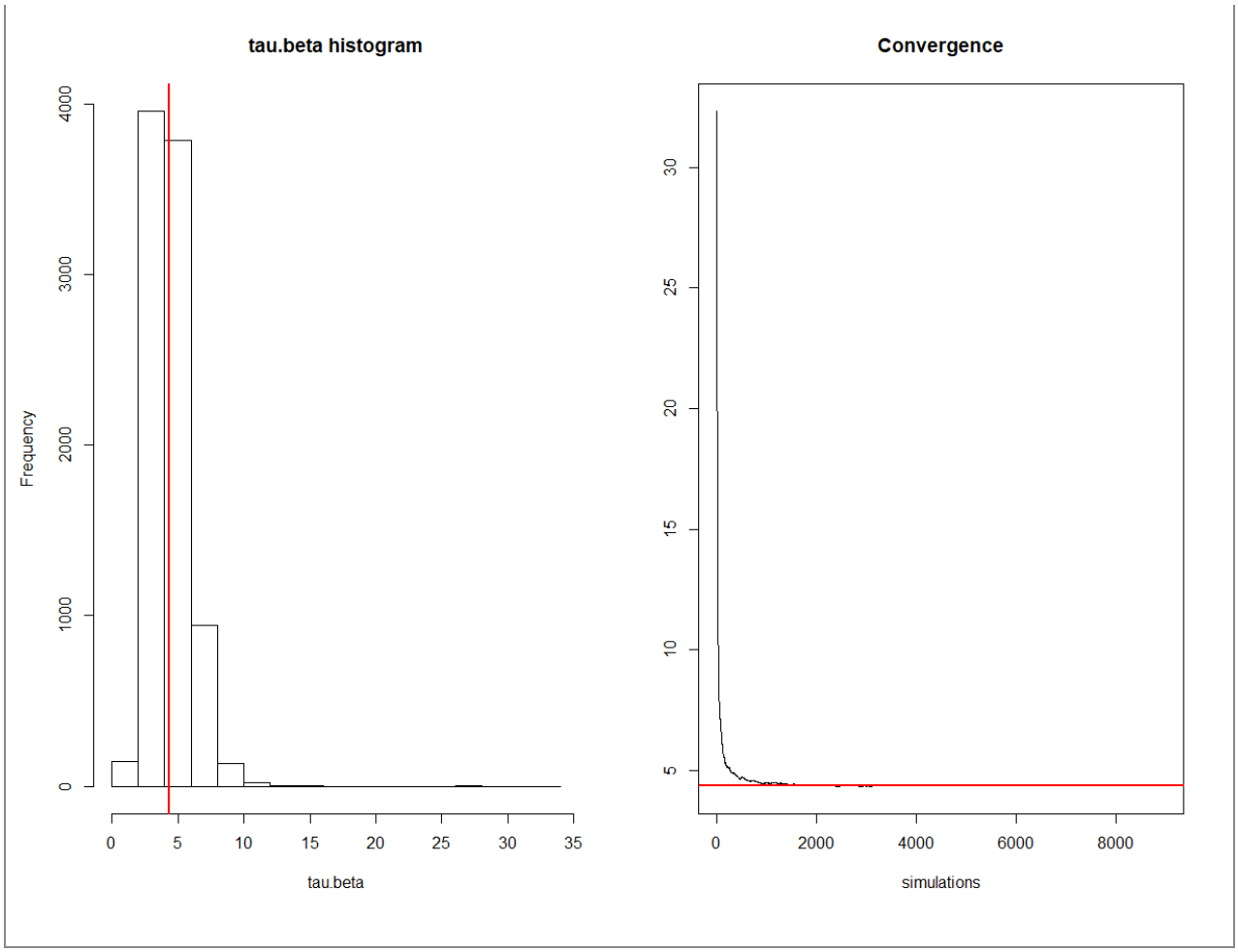





```
 #tau.beta
par(mfrow=c(1,1))
# tau.beta
tau.beta.vals <- ratsfit.model1$BUGSoutput$sims.array[,1,"tau.beta"]
plot(tau.beta.vals, xlab = "iterations", main="tau.beta trace plot",type="l")
```




```
par(mfrow=c(1,2))
hist(tau.beta.vals, main= "tau.beta histogram", xlab = "tau.beta")
abline(v=mean(tau.beta.vals), col="red", lwd=2)

plot(cumsum(tau.beta.vals)/(1:length(tau.beta.vals)), type="l", ylab="",
     main="Convergence", xlab="simulations")
abline(h=mean(tau.beta.vals), col="red", lwd=2)
```



5. Predictions

So, now I will try to predict the values. The length of the values of the parameters (tau.alpha, tau.beta and tau.c) is 9000. So we do the prediction as follows:

```
# Prediction
prediction <- function(x){
  alpha <- rep(NA, 9000)
  beta <- rep(NA, 9000)
  y.pred <- rep(NA, 9000)
  for(i in 1:9000){
    alpha[i] = rnorm(1,alpha.c.vals[i], tau.alpha.vals[i])
    beta[i] = rnorm(1,beta.c.vals[i], tau.beta.vals[i])
    y.pred[i] = alpha[i]+beta[i]*x
  }
  return(y.pred)
}
```

```
mean(prediction(0))
```

```
> mean(prediction(0))
```

```
[1] 106.5263
```

```
mean(prediction(8))
```

```
> mean(prediction(8))
[1] 156.1393
```

```
mean(prediction(15))
```

```
> mean(prediction(15))
[1] 199.3002
```

```
mean(prediction(22))
```

```
> mean(prediction(22))
[1] 242.8912
```

```
mean(prediction(29))
```

```
> mean(prediction(29))
[1] 283.9683
```

```
mean(prediction(36))
```

```
> mean(prediction(36))
[1] 329.4324
```

References:

1. http://www.openbugs.net/Examples/Rats.html
2. Practical Data Analysis with JAGS using R (http://bendixcarstensen.com/Bayes/Cph-2012/pracs.pdf)
3. http://www.stats.ox.ac.uk/~steffen/teaching/gm11/bayes.pdf
4. https://nature.berkeley.edu/~pdevalpine/MCMC_comparisons/some_BUGS_comparisons/rats/nimble_rats_comparisons.html