Statistical Methods in Data Science and Laboratory II: Final Project

Hepatitis: random effects model with measurement error

Alessandro Gallo

Contents

Introduction

Hepatits B (HB) is endemic in many parts of the world. In highly endemic areas such as West Africa, almost everyone is infected with the HB virus during childhood. About 20% of those infected don't completely clear the infected and risk chronic liver disease in adult life and death for liver cancer.

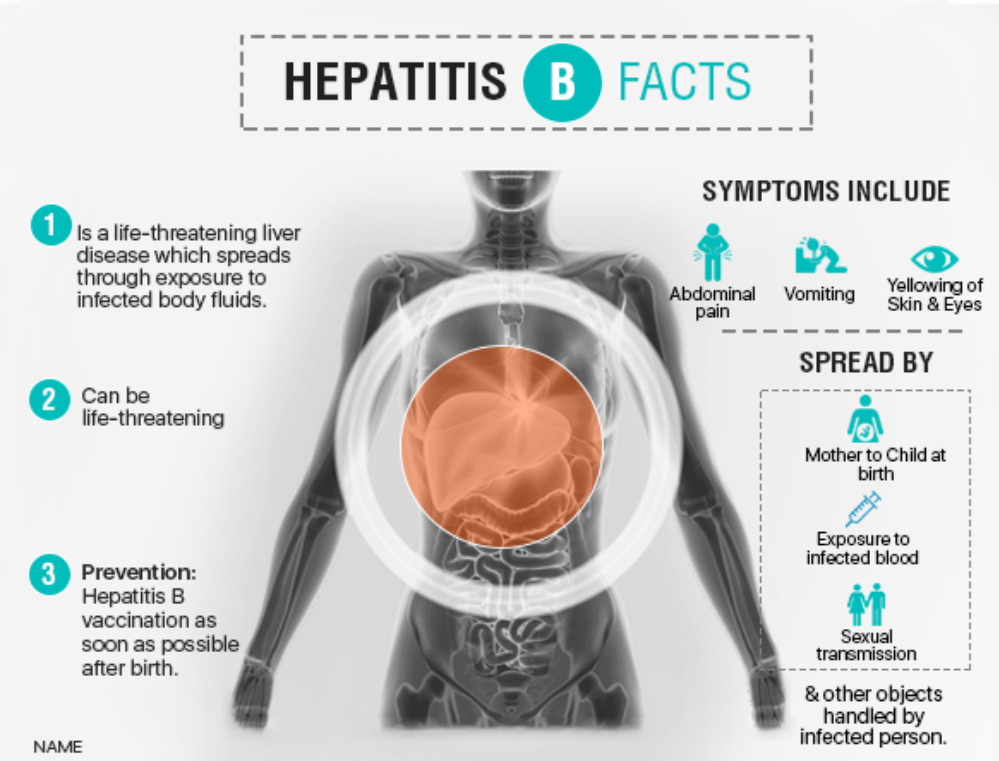

Figure 1: Hepatitis B - Summary

In this work blood samples were periodically collected from each infant, and the amount of surface-antibody; the measurement is called anti-HBs titre, measured in milli-International Units (mIU).

A similar study conclued that for all infants:

$$anti - HBs\ titre \propto \frac{1}{t} \tag{1}$$

where t denotes time since the infant's final vaccination, and the constant of proportionality may vary between infants. This is equivalent to a linear relationship between log titre and log time:

$$y = \alpha_i - 1 \times logt \tag{2}$$

where y denotes log anti-HBs titre and α_i is constant after the final dose of vaccine for each infant i.

The dataset

Let's consider the raw data for a subset of 106 infants from the dataset. These infants each had a baseline anti-HBs titre measurement taken at the time of the final vaccination, and at least two titre measurements taken subsequently.

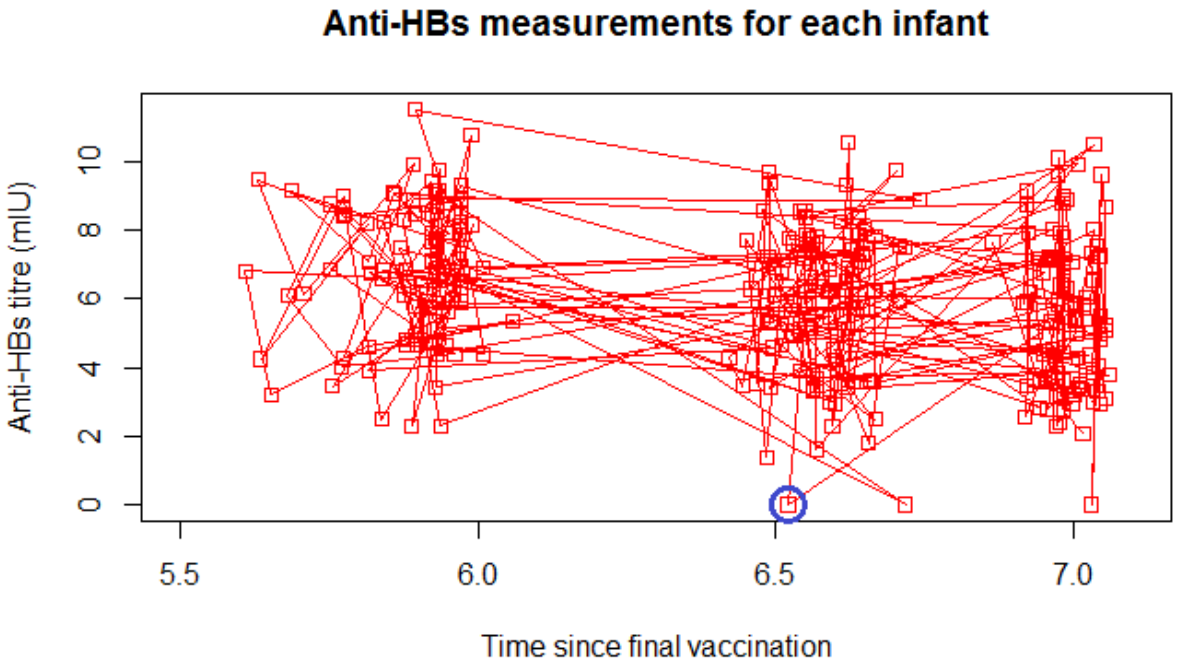

Figure 2: Anti-Hbs titre measurements for 106 infants (in log[t])

Of particular note is the infant labelled with blue circle whose titre apparently rose from 0 mIU to a much higher mIU in a new time . This somewhat atypical behaviour could be thought of as an outlier with respect to the change of titre over time.

Model 1

The first model was represented by Spiegelhalter (Markov Chain Montecarlo in Practice) by a random effects linear growth curve.

$$\mu_{ij} = \alpha_i + \beta_i(logt_{ij} - log730) + \gamma(y_{i0} - y_{.0}) \tag{3}$$

where:

- $logt_{ij}$ is standardized around $log730$ for numerical stability

- $y_{.0}$ is the mean of the observations y_{i0}; the covariate y_{i0} is "centred" by subtracting $y_{.0}$. This will help to reduce posterior correlations between γ and other parameters, and subsequently to improve mixing in the Gibbs sampler.

Model:

$$Y_{ij} \sim \mathcal{N}(\mu_{ij}, \tau) \tag{4}$$

$$\mu_{ij} = \alpha_i + \beta_i(t_{ij} - t_{bar}) + \gamma(y_{i0} - y_{.0}) \tag{5}$$

where τ represents the precision (1/variance) of a normal distribution:

$$\tau \sim \Gamma(0.001, 0.001) \tag{6}$$

and considering random effects on α_i and β_i:

$$\alpha_i \sim \mathcal{N}(\alpha_c, \tau_\alpha) \tag{7}$$

$$\beta_i \sim \mathcal{N}(\beta_0, \tau_\beta) \tag{8}$$

Priors:

$$\tau \sim \Gamma(0.001, 0.001) \tag{9}$$

$$\alpha_0 \sim \mathcal{N}(0.0, 1.0\text{E}{-6}) \tag{10}$$

$$\tau_\alpha \sim \Gamma(0.001, 0.001) \tag{11}$$

$$\beta_0 \sim \mathcal{N}(0.0, 1.0\text{E}{-6}) \tag{12}$$

$$\tau_\beta \sim \Gamma(0.001, 0.001) \tag{13}$$

$$\gamma \sim \mathcal{N}(0.0, 1.0\text{E}{-6}) \tag{14}$$

In hierarchical models such as ours, it is particularly important to avoid casual use of standard improper priors since these may result in improper posterior distributions (DuMouchel and Waternaux, 1992).

BUGS Directed Acyclic Graph (DAG)

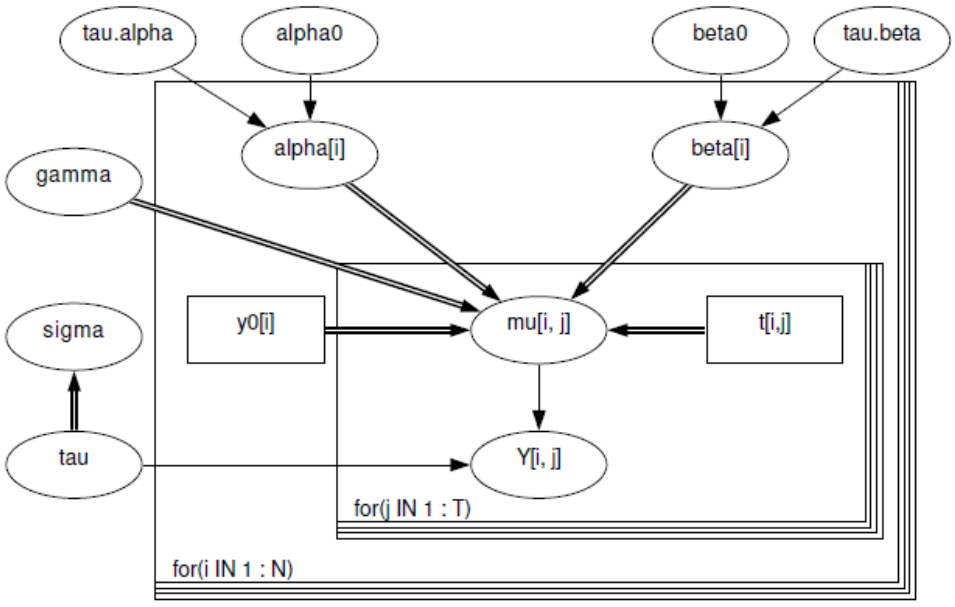

Figure 3: Hepatitis Model 1 DAG

BUGS Model code

```
model
{
   for( i in 1 : N ) {
      for( j in 1 : n[i] ) {
         Y[i , j] ~ dnorm(mu[i , j],tau)
         mu[i , j] <- alpha[i] + beta[i] * (t[i,j] - 6.5) +
                     gamma * (y0[i] - mean(y0[]))
      }
      alpha[i] ~ dnorm(alpha0,tau.alpha)
      beta[i] ~ dnorm(beta0,tau.beta)
   }
   tau ~ dgamma(0.001,0.001)
   sigma <- 1 / sqrt(tau)
   alpha0 ~ dnorm(0.0,1.0E-6)
   tau.alpha ~ dgamma(0.001,0.001)
   beta0 ~ dnorm(0.0,1.0E-6)
   tau.beta ~ dgamma(0.001,0.001)
   gamma ~ dnorm(0.0,1.0E-6)
}
```

Model Evaluation

```
library(ggplot2)
library(rjags)
library(ggmcmc)

set.seed(123)

source("dataset.R")

N=106
J=3
t=data$t
y0=data$y0

tau=sqrt(1/0.7)
alpha_c=6
tau_alpha=sqrt(1/0.7)
beta_c=-1
tau_beta=sqrt(1/0.7)
gamma = 1

alpha = rnorm(n=N,alpha_c,tau_alpha)
beta = rnorm(n=N,beta_c,tau_beta)
Y=data.frame(matrix(ncol = J, nrow = N))

for( i in 1 : N ) {
  for( j in 1 : J ) {
    mu <- alpha[i] + beta[i] * (t[i,j] - 6.5) + gamma * (y0[i] - mean(y0[]))
    Y[i , j] = rnorm(n=1, mean=mu, sd=tau)
  }
}


data1.eval = data
data1.eval$Y = Y
data1.eval$n=apply(!is.na(data1.eval$Y),1,sum)


hep1.eval.inits <- list(alpha0 = 4, beta0 = 0, gamma = 0, tau.alpha = 1,
tau.beta = 1, tau = 1 )
hep1.eval.sim = jags.model("modelHEP1.txt", data=data1.eval, inits=hep1.eval.inits,
n.adapt=5000, n.chains=3)
```

```
#update(sim.model, n.iter=5000)
hep1.eval.mod.samp = coda.samples(model=hep1.eval.sim,
                                  variable.names=c("alpha0", "beta0","gamma",
                                  "tau.alpha","tau.beta","tau"),
                                  n.iter=50000,
                                  n.thin=10,
                                  n.burnin=5000)
summary(hep1.eval.mod.samp)
```

Model evaluation Summary:

```
Iterations = 1:50000
Thinning interval = 1
Number of chains = 3
Sample size per chain = 50000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

             Mean      SD  Naive SE Time-series SE
alpha0     6.1232 0.11676 0.0003015      0.0005050
beta0     -1.3363 0.15493 0.0004000      0.0007571
gamma      0.9905 0.06548 0.0001691      0.0003445
tau        0.5992 0.08203 0.0002118      0.0006479
tau.alpha  1.3718 0.47897 0.0012367      0.0047451
tau.beta   0.9378 0.30117 0.0007776      0.0025930

2. Quantiles for each variable:

             2.5%     25%     50%     75%   97.5%
alpha0     5.8927  6.0452  6.1239  6.2011  6.3510
beta0     -1.6439 -1.4393 -1.3348 -1.2320 -1.0362
gamma      0.8620  0.9471  0.9905  1.0343  1.1182
tau        0.4525  0.5418  0.5940  0.6514  0.7736
tau.alpha  0.7763  1.0641  1.2771  1.5604  2.5263
tau.beta   0.4997  0.7226  0.8881  1.0967  1.6686
```

Simulation

```
# http://www.openbugs.net/Examples/Hepatitis.html

library(ggplot2)
```

```
library(rjags)
library(ggmcmc)
#library(runjags)

set.seed(123)

source("dataset.R")

plot(data$t,data$Y,type="l",col="red",xlim=c(5.5,7.1),main="Anti-HBs measurements
for each infant",
     xlab="Time since final vaccination", ylab="Anti-HBs titre (mIU)")
points(data$t,data$Y,col="red",xlim=c(5.5,7.1),pch=0)

hep1.inits <- list(alpha0 = 4, beta0 = 0, gamma = 0, tau.alpha = 1, tau.beta = 1,
tau = 1 )

hep1.sim = jags.model("modelHEP1.txt", data=data, inits=hep1.inits, n.adapt=5000,
n.chains=3)

#update(hep1.sim, n.iter=5000)

hep1.mod.samp = coda.samples(model=hep1.sim,
                             variable.names=c("alpha0", "beta0","gamma",
                             "tau.alpha","tau.beta","tau"),
                             n.iter=50000,
                             n.thin=10,
                             n.burnin=5000)
```

Simulation results

Simulation Summary:

```
Iterations = 1:50000
Thinning interval = 1
Number of chains = 3
Sample size per chain = 50000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

              Mean        SD  Naive SE Time-series SE
alpha0      6.1367   0.15058 0.0003888      4.537e-04
beta0      -1.0650   0.13515 0.0003490      4.212e-03
```

```
gamma       0.6738   0.08408 0.0002171      7.467e-04
tau         1.0034   0.10999 0.0002840      7.074e-04
tau.alpha   0.5090   0.08581 0.0002216      3.202e-04
tau.beta  223.9962 419.41491 1.0829246      1.257e+01

2. Quantiles for each variable:

             2.5%     25%     50%      75%     97.5%
alpha0     5.8417  6.0356  6.1364   6.2370    6.4353
beta0     -1.3334 -1.1562 -1.0610  -0.9702   -0.8111
gamma      0.5091  0.6176  0.6732   0.7299    0.8401
tau        0.8021  0.9272  0.9986   1.0737    1.2341
tau.alpha  0.3602  0.4485  0.5022   0.5619    0.6958
tau.beta   2.4950 13.9410 57.8122 240.0229 1409.5419
```

Simulation Density:

```
hep1_results = ggs(hep1.mod.samp)
ggs_density(hep1_results)
```

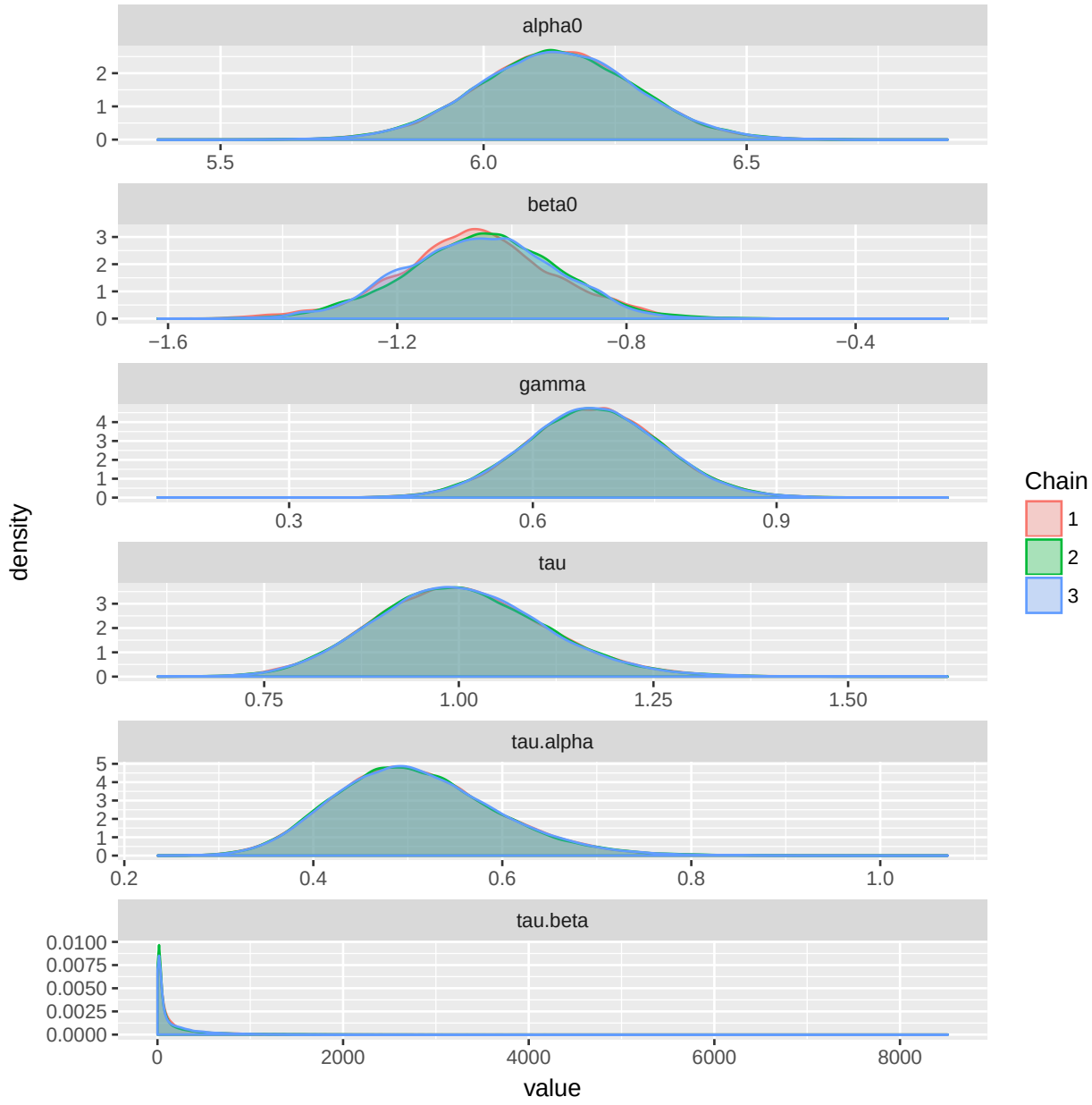

Figure 4: Simulation Density

Simulation Traceplot:

```
ggs_traceplot(hep1_results)
```

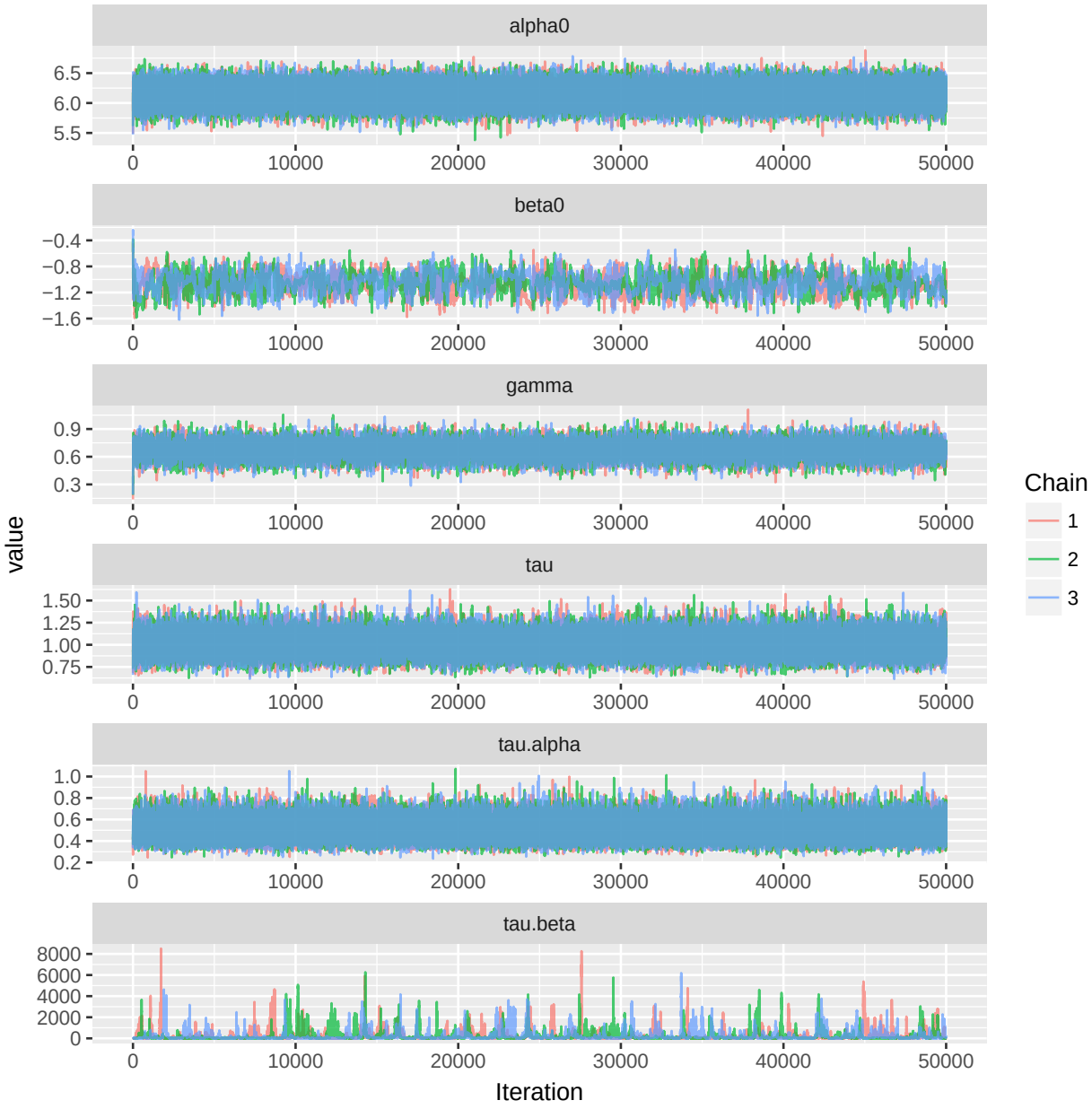

Figure 5: Simulation Traceplot

Simulation Running:

```
ggs_running(hep1_results)
```

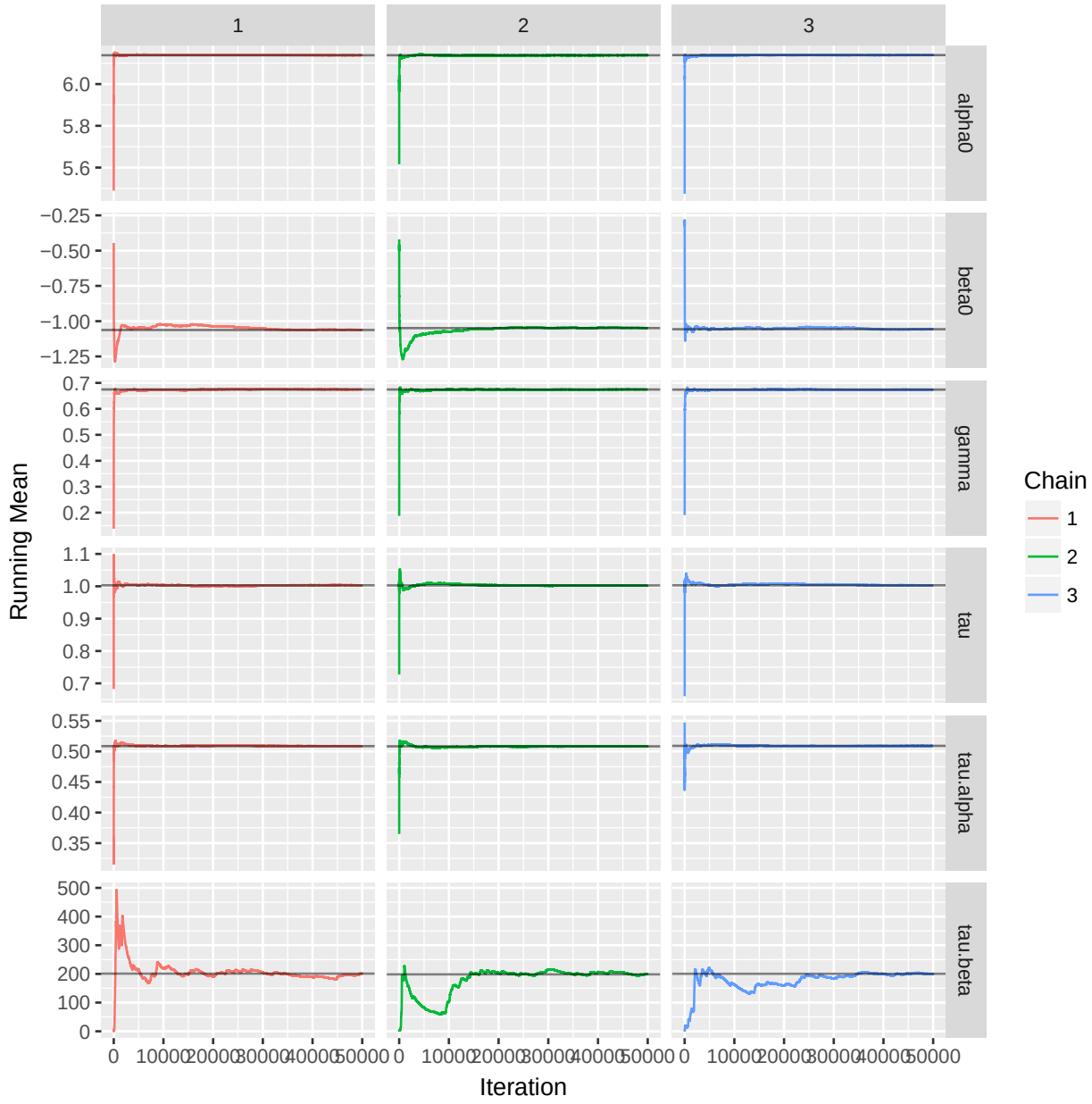

Figure 6: Simulation Runnings

Simulation Autocorrelation:

```
ggs_autocorrelation(hep1_results)
```

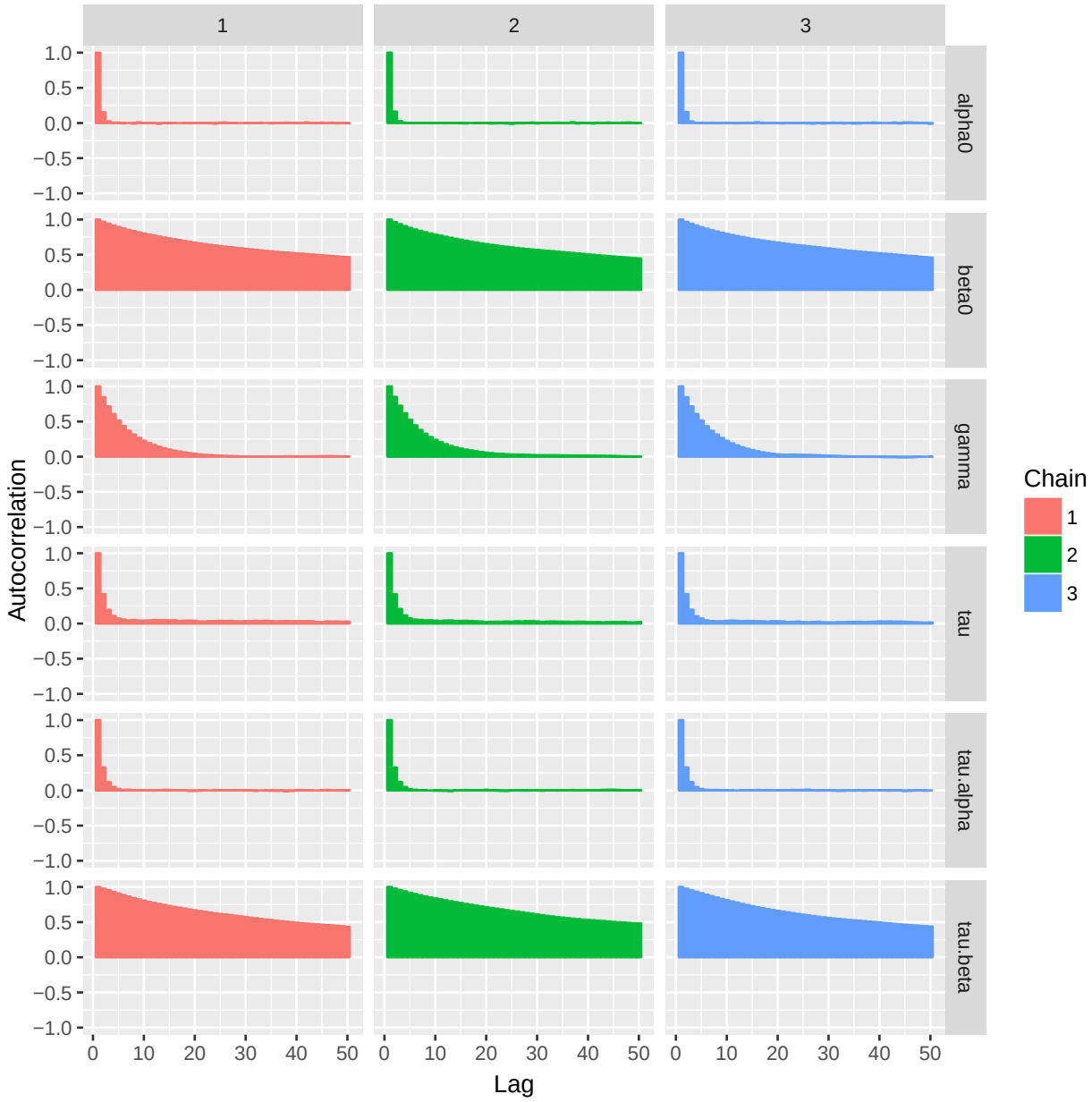

Figure 7: Simulation autocorrelations

Simulation Crosscorrelation

```
ggs_crosscorrelation(hep1_results)
```

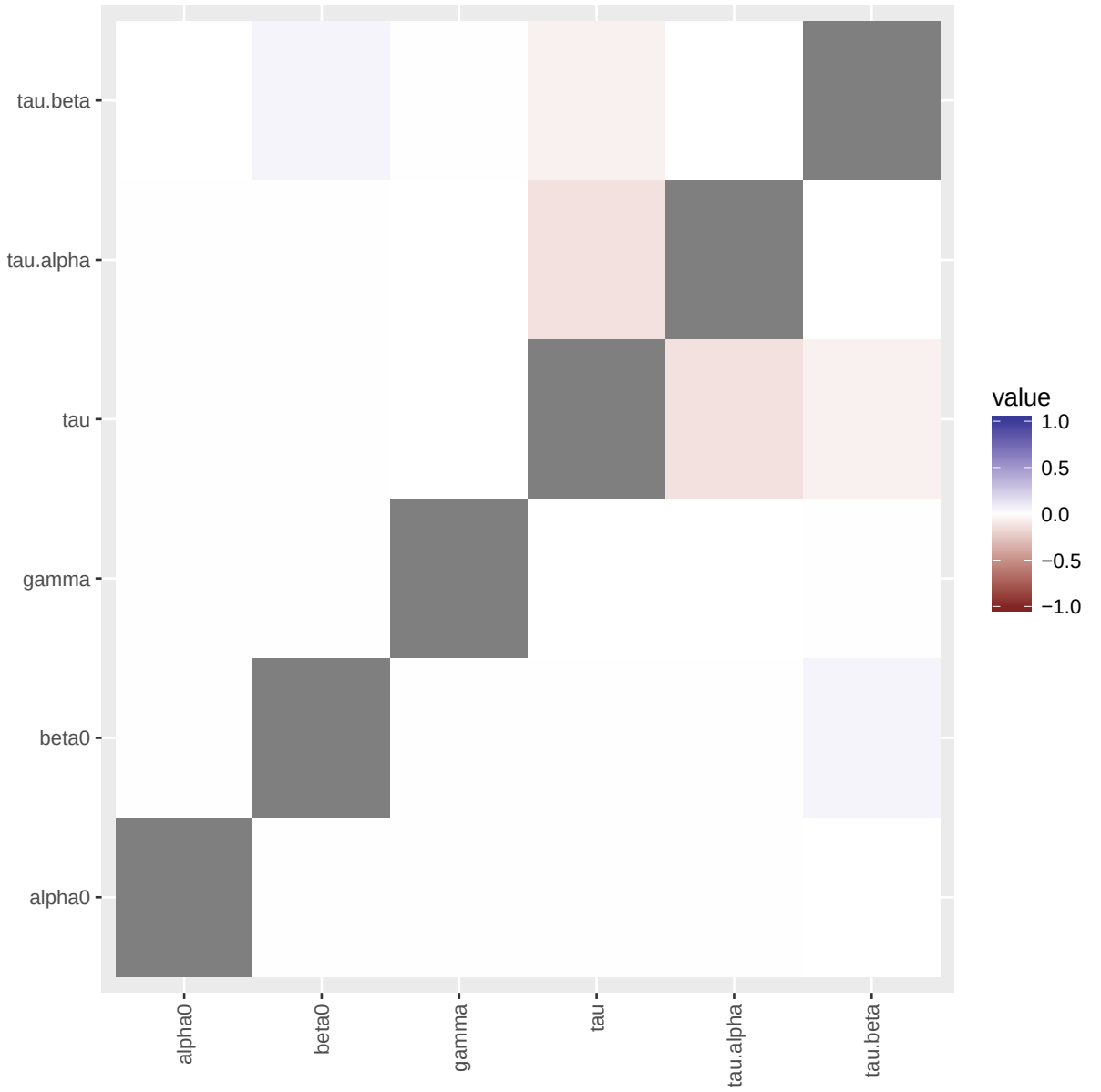

Figure 8: Simulation crosscorrelation

Giving a look to the summary we can ascertain that we found again the initial parameters, apart for β and τ_β where there has been some little anomaly in the density of the chains.

From the trace-plots infact we can see that the simulations of the other parameters, after the burn-in periods, have converged to their stationary distributions but the result of β is more uncertain (as well as τ_β).

And furthermore in the simulation runnings β and τ_β are slower to converge.

In the autocorrelations values tend to zero, showing how the chains are all Markovian (no long-memory processes), apart for β and τ_β where there have been some difficulties.

In the crosscorrelations only τ present just a slight negative correlation with τ_α and τ_β showing uncorrelations between our parameters.

Model 2

Let's make now a 2 model considering the measurement error:

$$Y_{ij} \sim \mathcal{N}(\mu_{ij}, \tau) \tag{15}$$

$$\mu_{ij} = \alpha_i + \beta_i(t_{ij} - t_{bar}) + \gamma(\mu_{0i} - y_{.0}) \tag{16}$$

where τ represents the precision (1/variance) of a normal distribution:

$$\tau \sim \Gamma(0.001, 0.001) \tag{17}$$

and considering random effects on α_i and β_i.

$$\alpha_i \sim \mathcal{N}(\alpha_c, \tau_\alpha) \tag{18}$$

$$\beta_i \sim \mathcal{N}(\beta_0, \tau_\beta) \tag{19}$$

Priors:

$$\tau \sim \Gamma(0.001, 0.001) \tag{20}$$

$$\alpha_0 \sim \mathcal{N}(0.0, 1.0\text{E}{-6}) \tag{21}$$

$$\tau_\alpha \sim \Gamma(0.001, 0.001) \tag{22}$$

$$\beta_0 \sim \mathcal{N}(0.0, 1.0\text{E}{-6}) \tag{23}$$

$$\tau_\beta \sim \Gamma(0.001, 0.001) \tag{24}$$

$$\gamma \sim \mathcal{N}(0.0, 1.0\text{E}{-6}) \tag{25}$$

and:

$$\theta \sim \mathcal{N}(0.0, 1.0\text{E}{-6}) \tag{26}$$

$$\psi \sim \Gamma(0.001, 0.001) \tag{27}$$

$$\mu_{0i} \sim \mathcal{N}(\theta, \psi) \tag{28}$$

$$y_0 \sim \mathcal{N}(\mu_{0i}, \tau) \tag{29}$$

BUGS Directed Acyclic Graph (DAG)

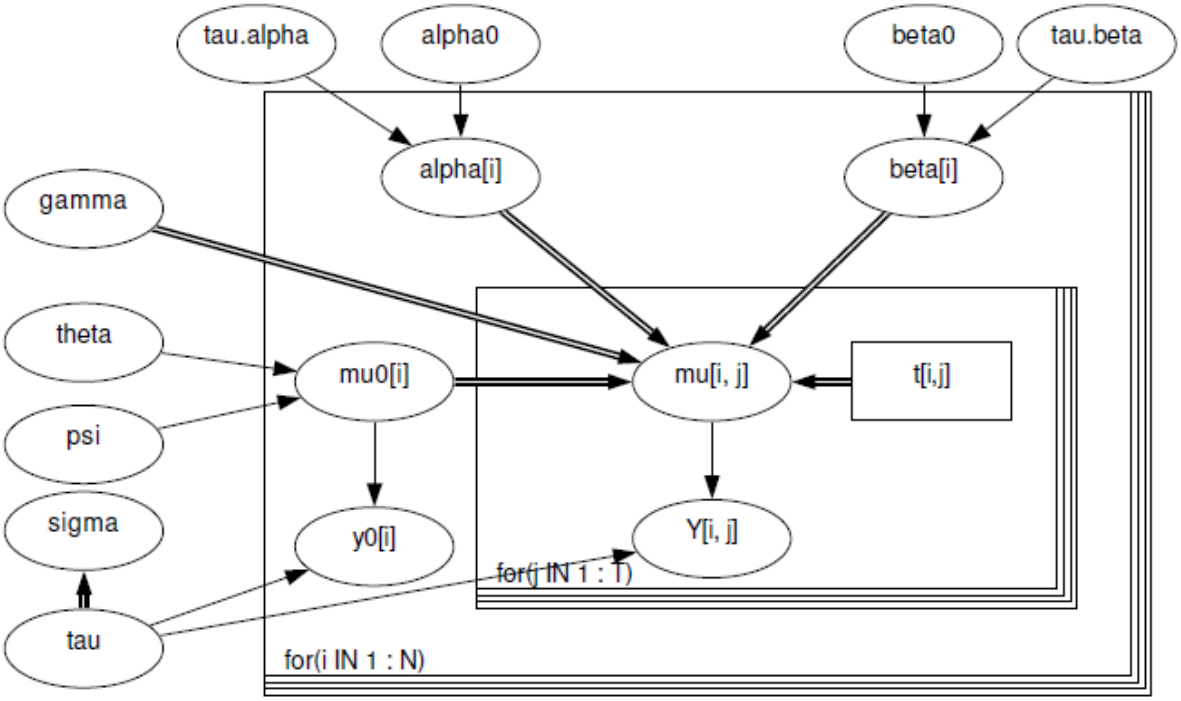

Figure 9: Hepatitis Model 2 DAG

BUGS Model code

```
model
{
  tau.alpha ~ dgamma(0.001,0.001)
  alpha0 ~ dnorm( 0.0,1.0E-6)
  beta0 ~ dnorm( 0.0,1.0E-6)
  tau.beta ~ dgamma(0.001,0.001)
  for( i in 1 : N ) {
     alpha[i] ~ dnorm(alpha0,tau.alpha)
     beta[i] ~ dnorm(beta0,tau.beta)
     y0[i] ~ dnorm(mu0[i],tau)
     mu0[i] ~ dnorm(theta,psi)
  }
  for( i in 1 : N ) {
     for( j in 1 : n[i] ) {
        Y[i , j] ~ dnorm(mu[i , j],tau)
        mu[i , j] <- alpha[i] + beta[i] * (t[i , j] - 6.5) +
           gamma * (mu0[i] - mean(y0[]))
```

```
    }
  }
  tau ~ dgamma(0.001,0.001)
  sigma <- 1 / sqrt(tau)
  gamma ~ dnorm( 0.0,1.0E-6)
  theta ~ dnorm( 0.0,1.0E-6)
  psi ~ dgamma(0.001,0.001)
}
```

Model Evaluation

```
library(ggplot2)
library(rjags)
library(ggmcmc)

source("dataset.R")

N=106
J=3
t=data$t



tau=sqrt(1/0.8)
alpha_c=6
tau_alpha=sqrt(1/0.8)
beta_c=-1
tau_beta=sqrt(1/0.8)
gamma = 10
theta=2
psi=sqrt(1/0.8)

alpha = rnorm(n=N,alpha_c,tau_alpha)
beta = rnorm(n=N,beta_c,tau_beta)
mu0=rep(NA,N)
y0=rep(NA,N)

Y=data.frame(matrix(ncol = J, nrow = N))

for( i in 1 : N ) {
  mu0[i] = rnorm(n=1,theta,psi)
  y0[i] = rnorm(n=1,mu0[i],tau)
}
```

```
for( i in 1 : N ) {
  for( j in 1 : J ) {
    mu <- alpha[i] + beta[i] * (t[i,j] - 6.5) + gamma * (mu0[i] - mean(y0[]))
    Y[i , j] = rnorm(n=1, mean=mu, sd=tau)
  }
}

data2.eval = data
data2.eval$Y = Y
data2.eval$n=apply(!is.na(data2.eval$Y),1,sum)


hep2.eval.inits <- list(alpha0 = 4, beta0 = 0, gamma = 0, tau.alpha = 1,
tau.beta = 1, tau = 1, theta = 6, psi = 1)
hep2.eval.mod = jags.model("modelHEP2.txt", data=data2.eval, inits=hep2.eval.inits,
n.adapt=5000, n.chains=3)
#update(sim.model, n.iter=5000)
hep2.eval.mod.samp = coda.samples(model=hep2.eval.mod,
                                  variable.names=c("alpha0", "beta0","gamma",
                                  "tau.alpha","tau.beta","tau","theta","psi"),
                                  n.iter=50000,
                                  n.thin=100,
                                  n.burnin=5000)

summary(hep2.eval.mod.samp)
```

Model evaluation Summary:

```
Iterations = 1:50000
Thinning interval = 1
Number of chains = 3
Sample size per chain = 50000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

              Mean       SD  Naive SE Time-series SE
alpha0      6.1374   0.1636 0.0004223       0.001617
beta0      -1.0611   0.1360 0.0003510       0.004552
gamma       1.0627   0.1978 0.0005107       0.005189
psi         0.4999   0.1252 0.0003234       0.002112
tau         0.9681   0.1177 0.0003038       0.002282
tau.alpha  13.2871 140.8727 0.3637317       6.238071
```

```
tau.beta  205.8340 386.2766 0.9973618      11.742527
theta       6.7576   0.1733 0.0004475       0.001092

2. Quantiles for each variable:

             2.5%     25%     50%      75%     97.5%
alpha0     5.8116  6.0294  6.1389   6.2469    6.4553
beta0     -1.3206 -1.1553 -1.0640  -0.9689   -0.7906
gamma      0.7435  0.9284  1.0367   1.1664    1.5365
psi        0.3163  0.4137  0.4800   0.5626    0.8031
tau        0.7472  0.8870  0.9649   1.0452    1.2086
tau.alpha  0.4749  0.6724  0.8424   1.1523   43.9016
tau.beta   2.5266 14.7492 56.9468 218.5541 1310.8710
theta      6.4173  6.6418  6.7567   6.8733    7.0996
```

Simulation

```
library(ggplot2)
library(rjags)
library(ggmcmc)

source("dataset.R")

hep2.inits <- list(alpha0 = 4, beta0 = 0, gamma = 0, tau.alpha = 1, tau.beta = 1,
tau = 1, theta = 6, psi = 1)

hep2.sim = jags.model("modelHEP2.txt", data=data, inits=hep2.inits, n.adapt=5000,
n.chains=3)

#update(sim.model, n.iter=5000)

hep2.mod.samp = coda.samples(model=hep2.sim,
                             variable.names=c("alpha0", "beta0","gamma",
                             "tau.alpha","tau.beta","tau","theta","psi"),
                             n.iter=50000,
                             n.thin=10,
                             n.burnin=5000)
```

Simulation results

Simulation Summary:

```
Iterations = 1:50000
Thinning interval = 1
Number of chains = 3
Sample size per chain = 50000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

              Mean       SD  Naive SE Time-series SE
alpha0      6.1374   0.1636 0.0004223       0.001617
beta0      -1.0611   0.1360 0.0003510       0.004552
gamma       1.0627   0.1978 0.0005107       0.005189
psi         0.4999   0.1252 0.0003234       0.002112
tau         0.9681   0.1177 0.0003038       0.002282
tau.alpha  13.2871 140.8727 0.3637317       6.238071
tau.beta  205.8340 386.2766 0.9973618      11.742527
theta       6.7576   0.1733 0.0004475       0.001092

2. Quantiles for each variable:

             2.5%     25%     50%      75%     97.5%
alpha0     5.8116  6.0294  6.1389   6.2469    6.4553
beta0     -1.3206 -1.1553 -1.0640  -0.9689   -0.7906
gamma      0.7435  0.9284  1.0367   1.1664    1.5365
psi        0.3163  0.4137  0.4800   0.5626    0.8031
tau        0.7472  0.8870  0.9649   1.0452    1.2086
tau.alpha  0.4749  0.6724  0.8424   1.1523   43.9016
tau.beta   2.5266 14.7492 56.9468 218.5541 1310.8710
theta      6.4173  6.6418  6.7567   6.8733    7.0996
```

Simulation Density:

```
ggs_density(hep2.results)
```

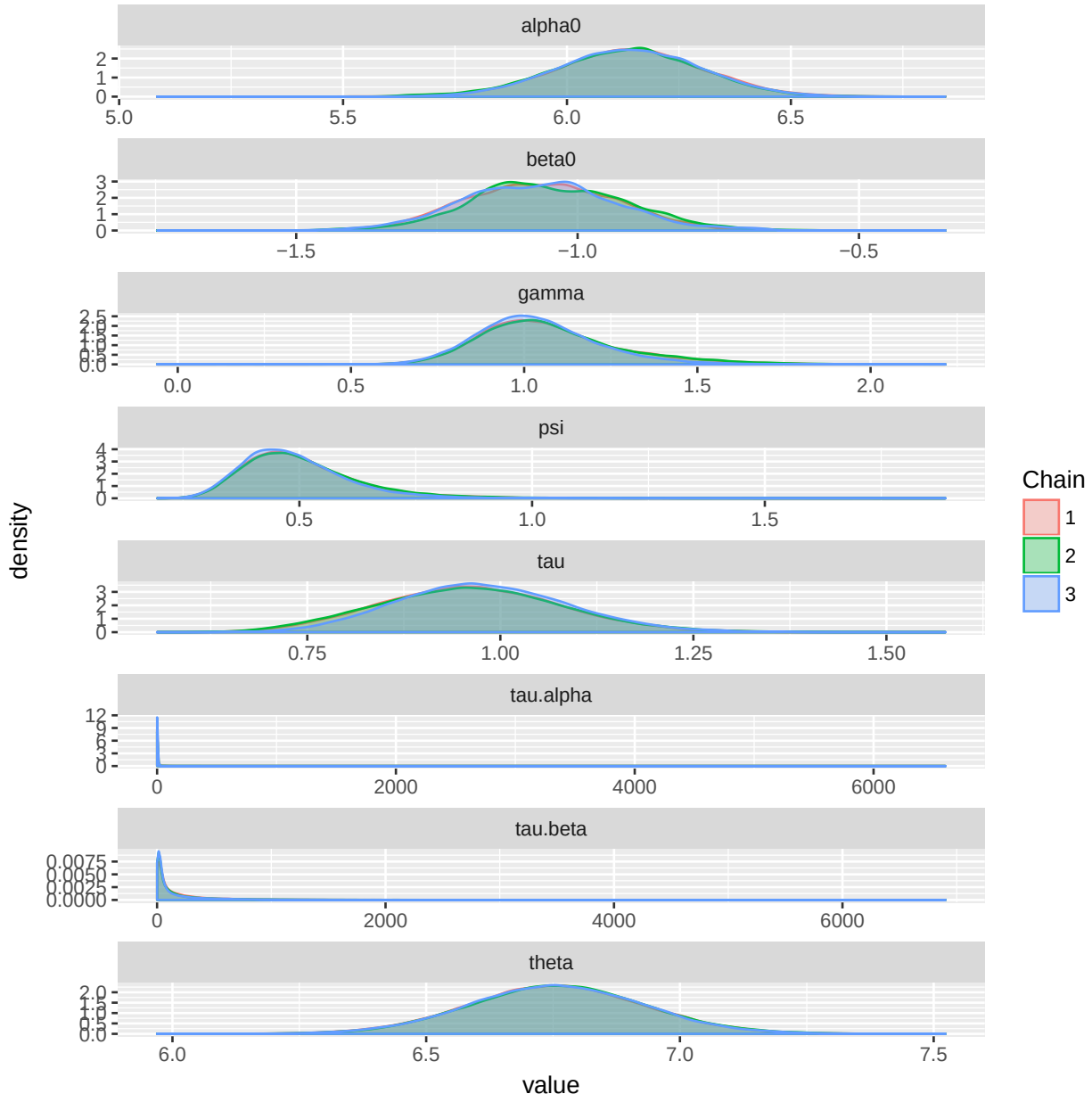

Figure 10: Simulation Density

Simulation Traceplot:

```
ggs_traceplot(hep2.results)
```

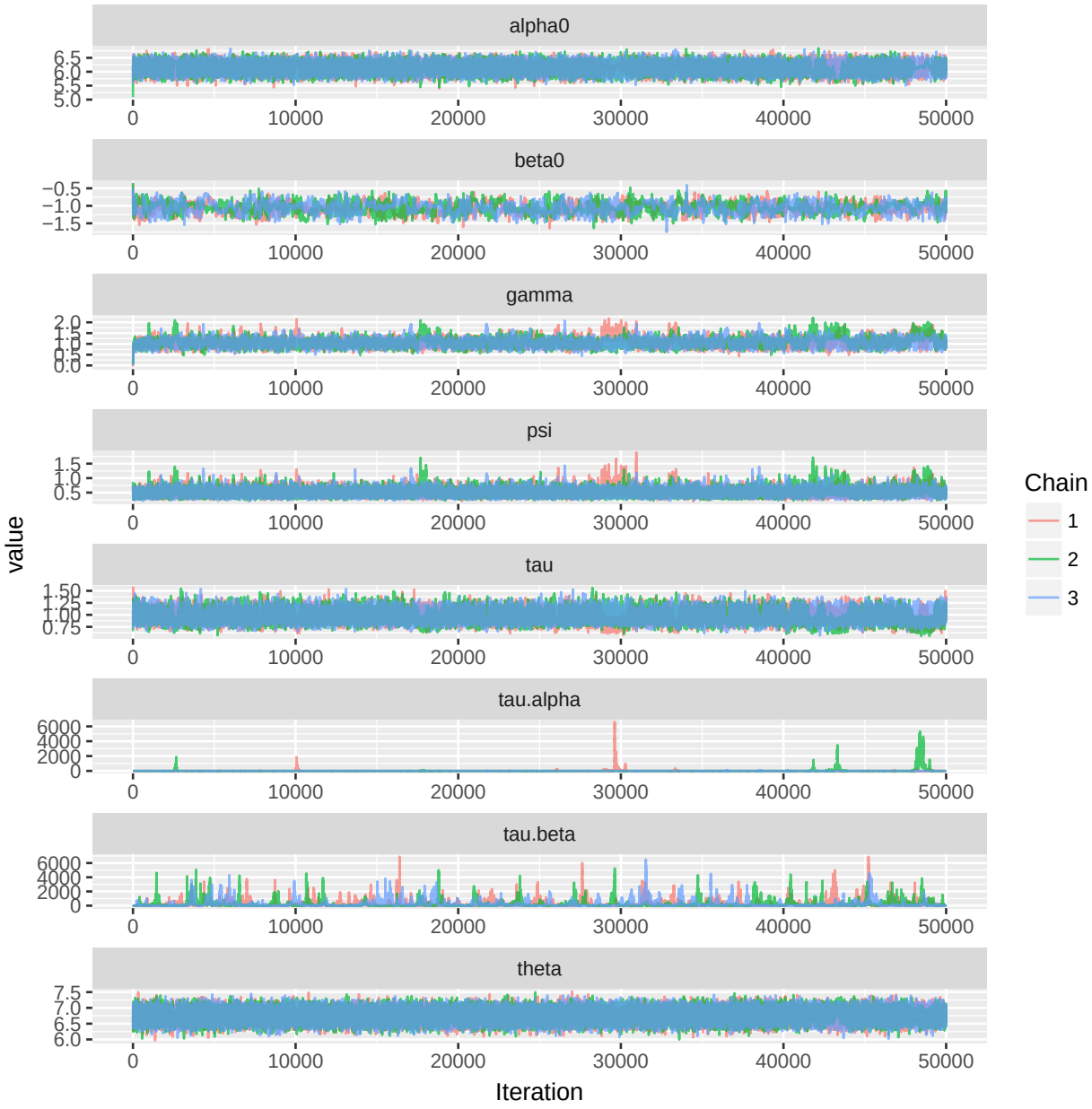

Figure 11: Simulation Traceplot

Simulation Running:

```
ggs_running(hep2.results)
```

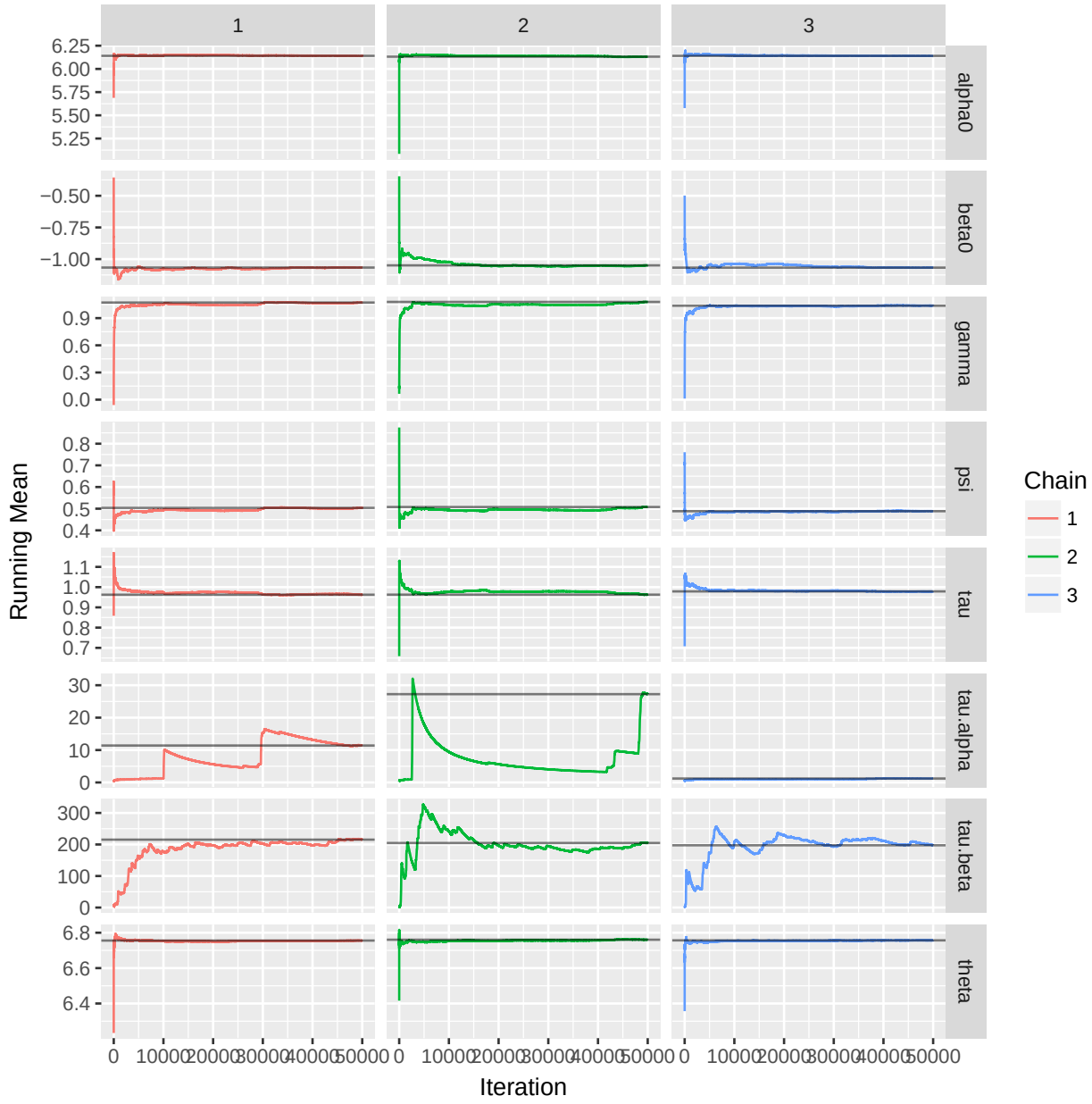

Figure 12: Simulation Runnings

Simulation Autocorrelation:

```
ggs_autocorrelation(hep2.results)
```

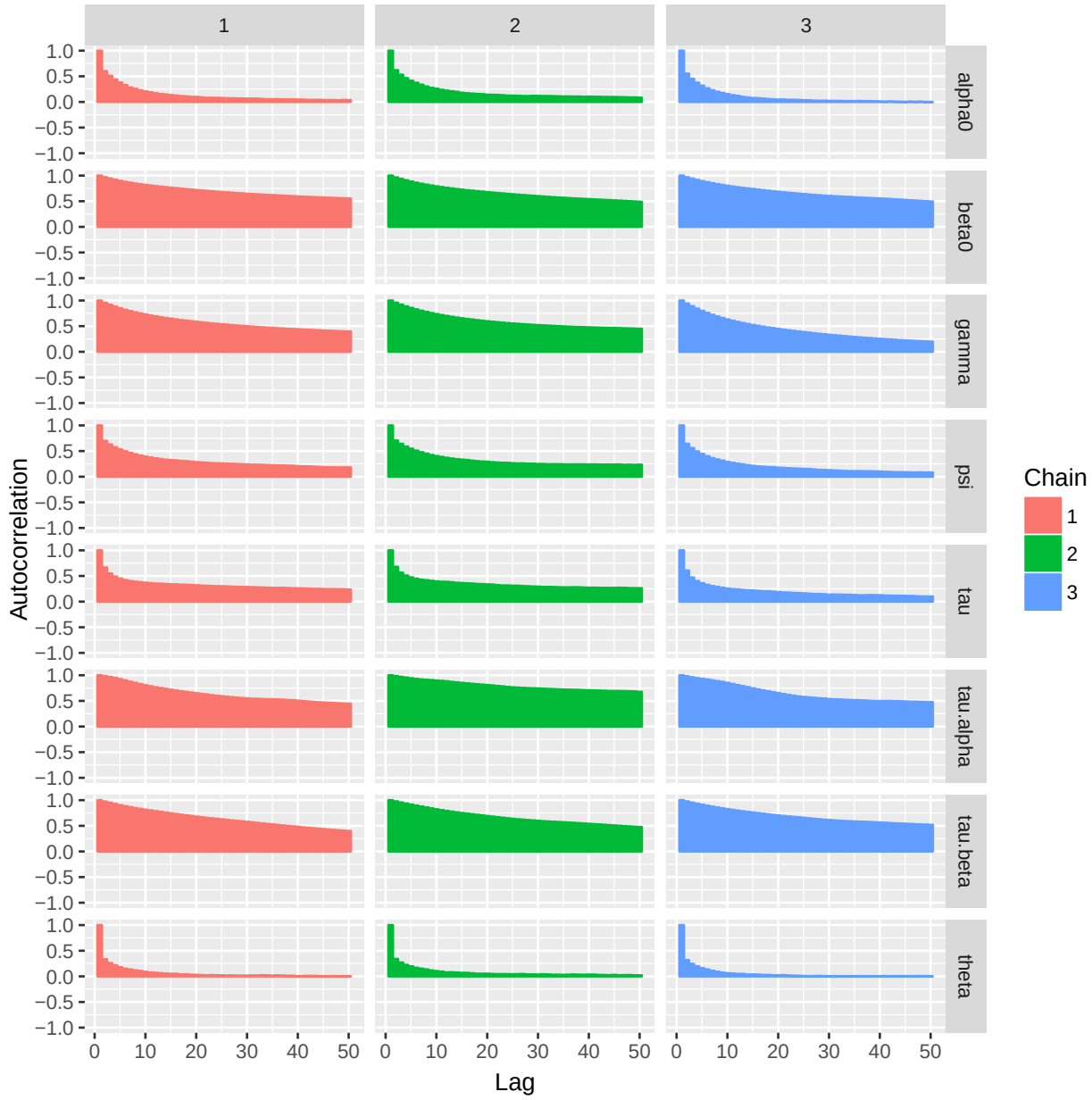

Figure 13:

Simulation Crosscorrelation

```
ggs_crosscorrelation(hep2.results)
```

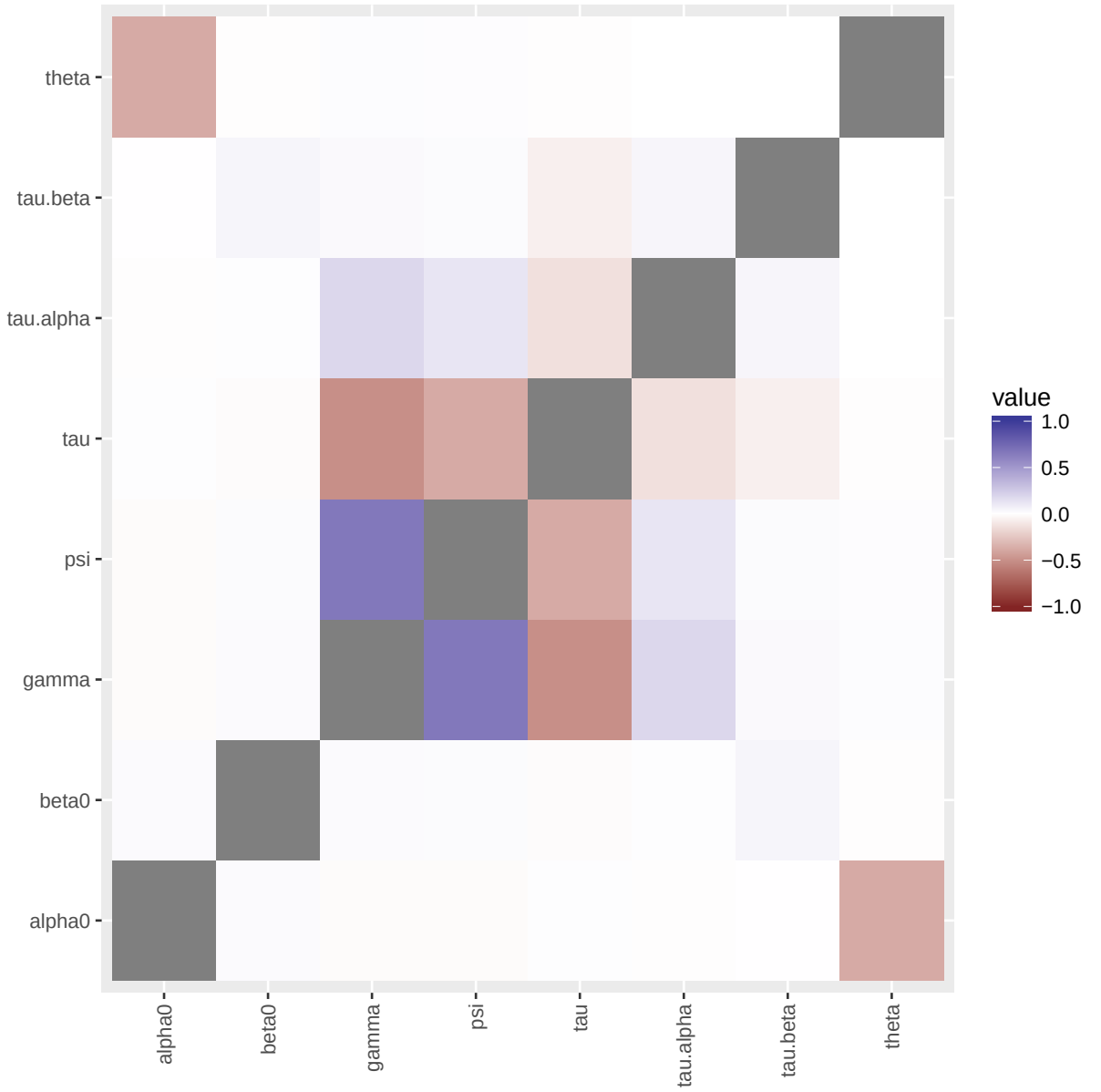

Figure 14: Simulation Crosscorrelations

In this case α has again a good value and it seems that β works better than the first model, but, in order to obtain these results, we had to set a high precision to the initial parameters (τ, τ_α,τ_β and ψ).

Despite this, from the trace-plots there are some anomalies, as well as in the running means, in particular for τ_α and τ_β.

It is in the autocorrelations that we find strange behaviours where almost all the values have difficulties to tend to zero.

In the crosscorrelations also there is perfect positive correlation between γ and ψ and an almost perfect negative correlation between τ and γ.
In conclusion we had to set a high precision in order to obtain discrete values for α and β but the model basically works very bad. This is probably due to the substitution of the covariate y_0 of the first model with a paramter that interferes with α.

Model 3

Due to the ambiguous behaviour of the previous model, this time we lighten our model simply avoiding random effects on α and β and not considering the adjustment covariate y_0.

$$Y_{ij} \sim \mathcal{N}(\mu_{ij}, \tau) \tag{30}$$

$$\mu_{ij} = \alpha + \beta(t_{ij} - t_{bar}) \tag{31}$$

where τ represents the precision (1/variance) of a normal distribution:

$$\tau \sim \Gamma(0.001, 0.001) \tag{32}$$

Priors:

$$\tau \sim \Gamma(0.001, 0.001) \tag{33}$$

$$\alpha \sim \mathcal{N}(0.0, 1.0\mathrm{E}{-6}) \tag{34}$$

$$\beta \sim \mathcal{N}(0.0, 1.0\mathrm{E}{-6}) \tag{35}$$

BUGS Directed Acyclic Graph (DAG)

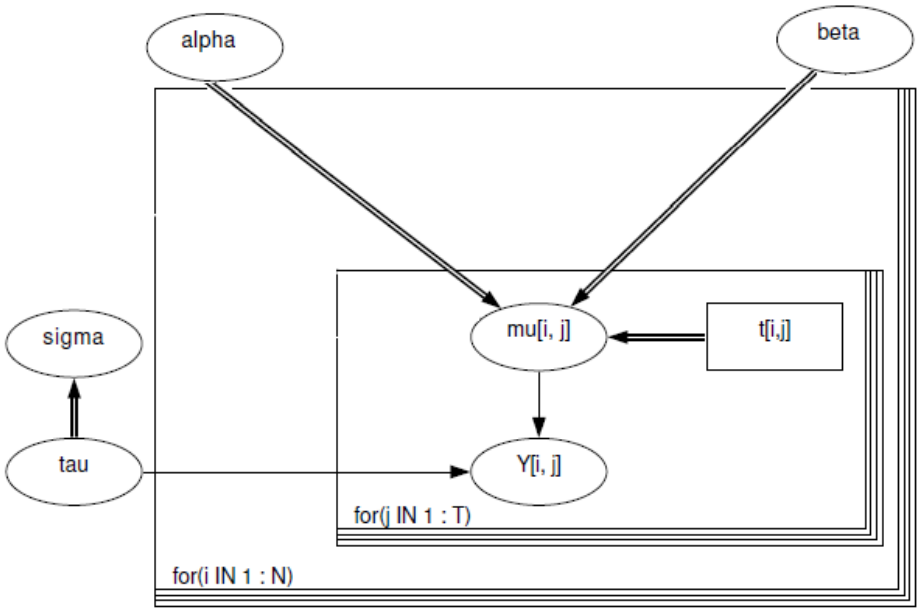

Figure 15: Hepatitis Model 3 DAG

BUGS Model code

```
model
{
  alpha ~ dnorm( 0.0,1.0E-6)
  beta ~ dnorm( 0.0,1.0E-6)
  for( i in 1 : N ) {
     for( j in 1 : n[i] ) {
        Y[i , j] ~ dnorm(mu[i , j],tau)
        mu[i , j] <- alpha + beta * (t[i , j] - 6.5)
     }
  }
  tau ~ dgamma(0.001,0.001)
  sigma <- 1 / sqrt(tau)
}
```

Model Evaluation

```
library(ggplot2)
library(rjags)
```

```
library(ggmcmc)

set.seed(123)

source("dataset.R")

N=106
J=3
t=data$t

sigma=0.8
tau=1/(sigma^2)
alpha=6
beta=1

Y=data.frame(matrix(ncol = J, nrow = N))

for( i in 1 : N ) {
  for( j in 1 : J ) {
    mu <- alpha + beta * (t[i,j] - 6.5)
    Y[i , j] = rnorm(n=1, mean=mu, sd=sigma)
  }
}

data3.eval = data
data3.eval$Y = Y
data3.eval$n=apply(!is.na(data3.eval$Y),1,sum)


hep3.eval.inits <- list(alpha=4, beta = 0, tau = 1)
hep3.eval.mod = jags.model("modelHEP3.txt", data=data3.eval, inits=hep3.eval.inits,
n.adapt=5000, n.chains=3)
#update(sim.model, n.iter=5000)
hep3.eval.mod.samp = coda.samples(model=hep3.eval.mod,
                                  variable.names=c("alpha", "beta","tau"),
                                  n.iter=50000,
                                  n.thin=100,
                                  n.burnin=5000)

summary(hep3.eval.mod.samp)
```

Model evaluation Summary:

```
Iterations = 1:50000
Thinning interval = 1
Number of chains = 3
Sample size per chain = 50000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

       Mean      SD  Naive SE Time-series SE
alpha 6.011 0.04502 1.162e-04      0.0001254
beta  1.019 0.03854 9.951e-05      0.0001071
tau   1.712 0.13651 3.525e-04      0.0003544

2. Quantiles for each variable:

        2.5%    25%   50%   75% 97.5%
alpha 5.9229 5.9810 6.011 6.042 6.099
beta  0.9436 0.9936 1.019 1.045 1.095
tau   1.4544 1.6183 1.708 1.802 1.989
```

Simulation

```
library(ggplot2)
library(rjags)
library(ggmcmc)

source("dataset.R")

hep3.inits <- list(alpha=4, beta = 0, tau = 1)

hep3.sim = jags.model("modelHEP3.txt", data=data, inits=hep3.inits, n.adapt=5000,
n.chains=3)

#update(sim.model, n.iter=5000)

hep3.mod.samp = coda.samples(model=hep3.sim,
                             variable.names=c("alpha", "beta","tau"),
                             n.iter=50000,
                             n.thin=100,
                             n.burnin=5000)

summary(hep3.mod.samp)
```

Simulation results

Simulation Summary:

```
Iterations = 1:50000
Thinning interval = 1
Number of chains = 3
Sample size per chain = 50000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

         Mean      SD  Naive SE Time-series SE
alpha  6.1031 0.12444 3.213e-04      3.206e-04
beta  -1.0887 0.27248 7.035e-04      6.996e-04
tau    0.2258 0.01891 4.883e-05      4.879e-05

2. Quantiles for each variable:

         2.5%     25%     50%     75%   97.5%
alpha  5.8585  6.0193  6.1032  6.1870  6.3465
beta  -1.6234 -1.2722 -1.0886 -0.9055 -0.5550
tau    0.1902  0.2128  0.2252  0.2382  0.2643
```

Simulation Density:

```
ggs_density(hep3.results)
```

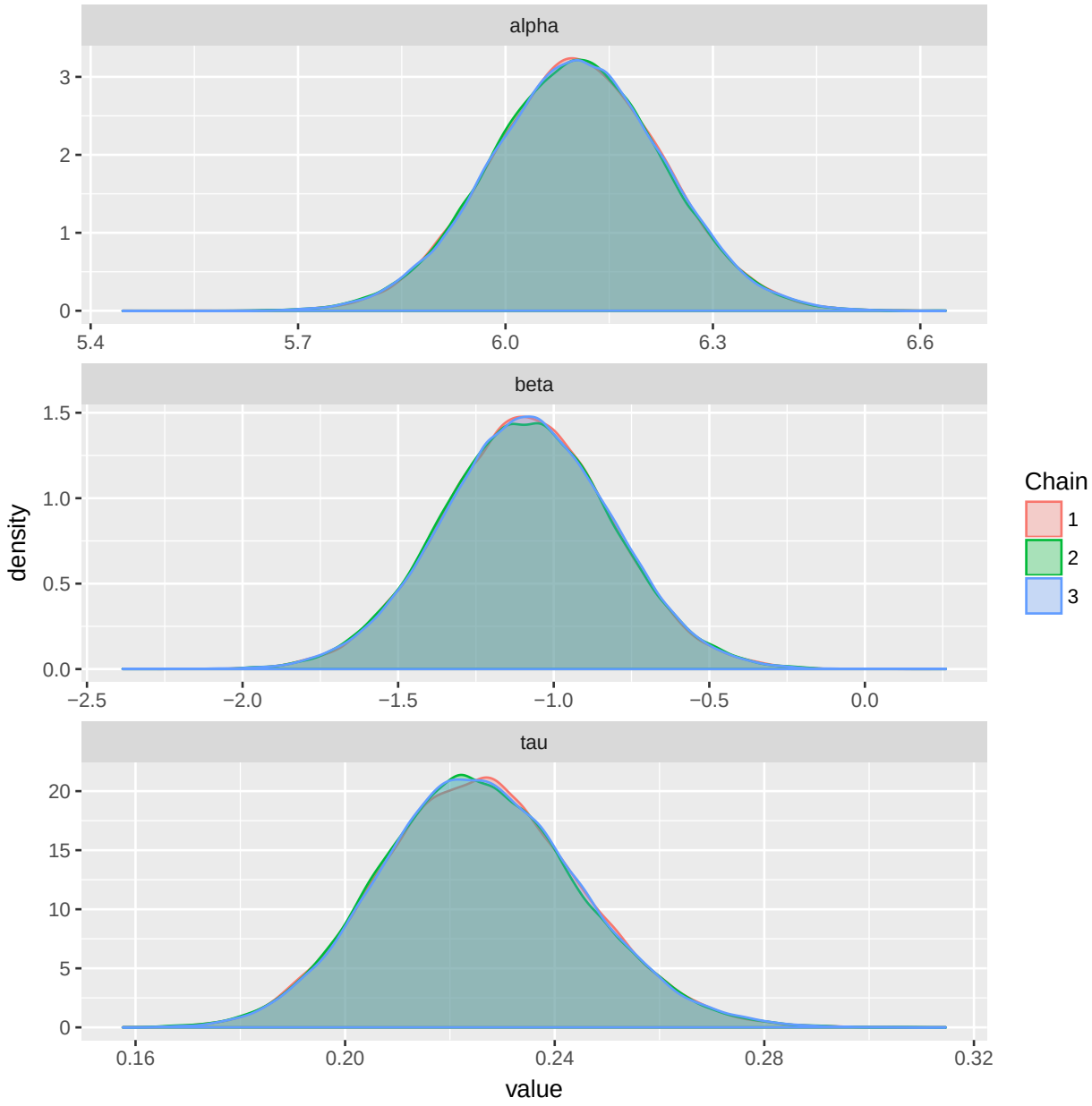

Figure 16: Simulation Density

Simulation Traceplot:

```
ggs_traceplot(hep3.results)
```

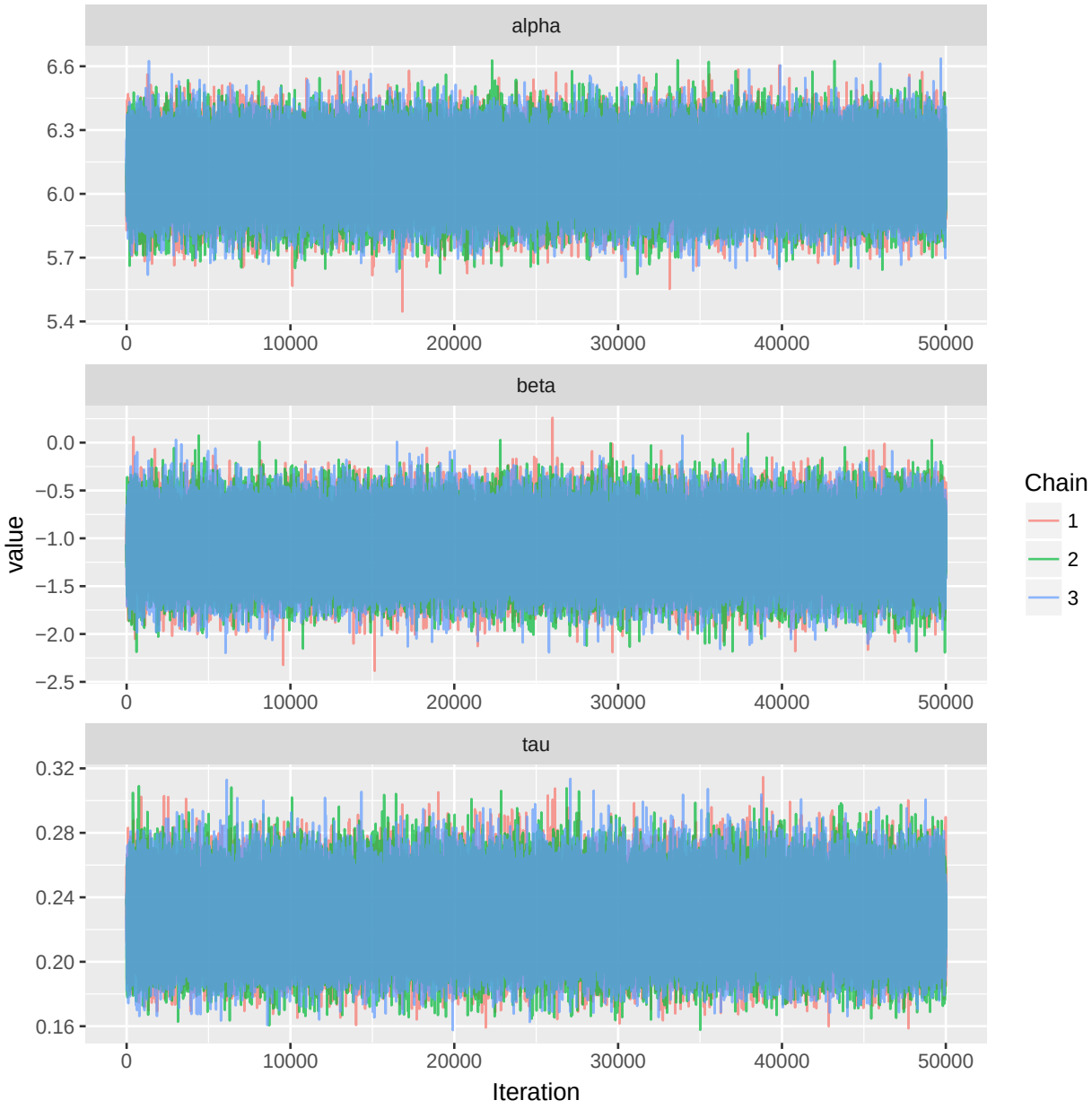

Figure 17: Simulation Traceplot

Simulation Running:

```
ggs_running(hep3.results)
```

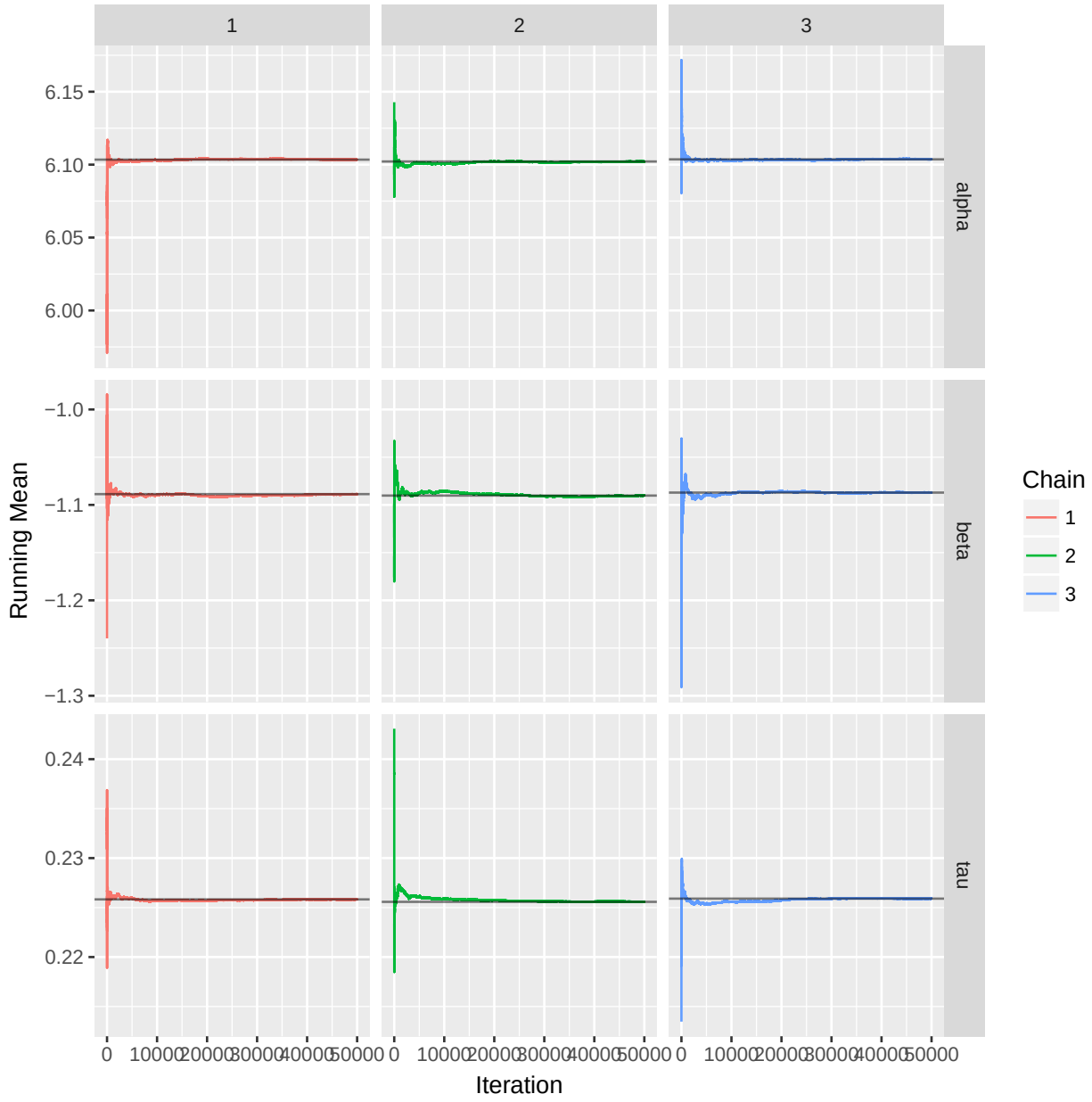

Figure 18: Simulation Runnings

Simulation Autocorrelation:

```
ggs_autocorrelation(hep3.results)
```

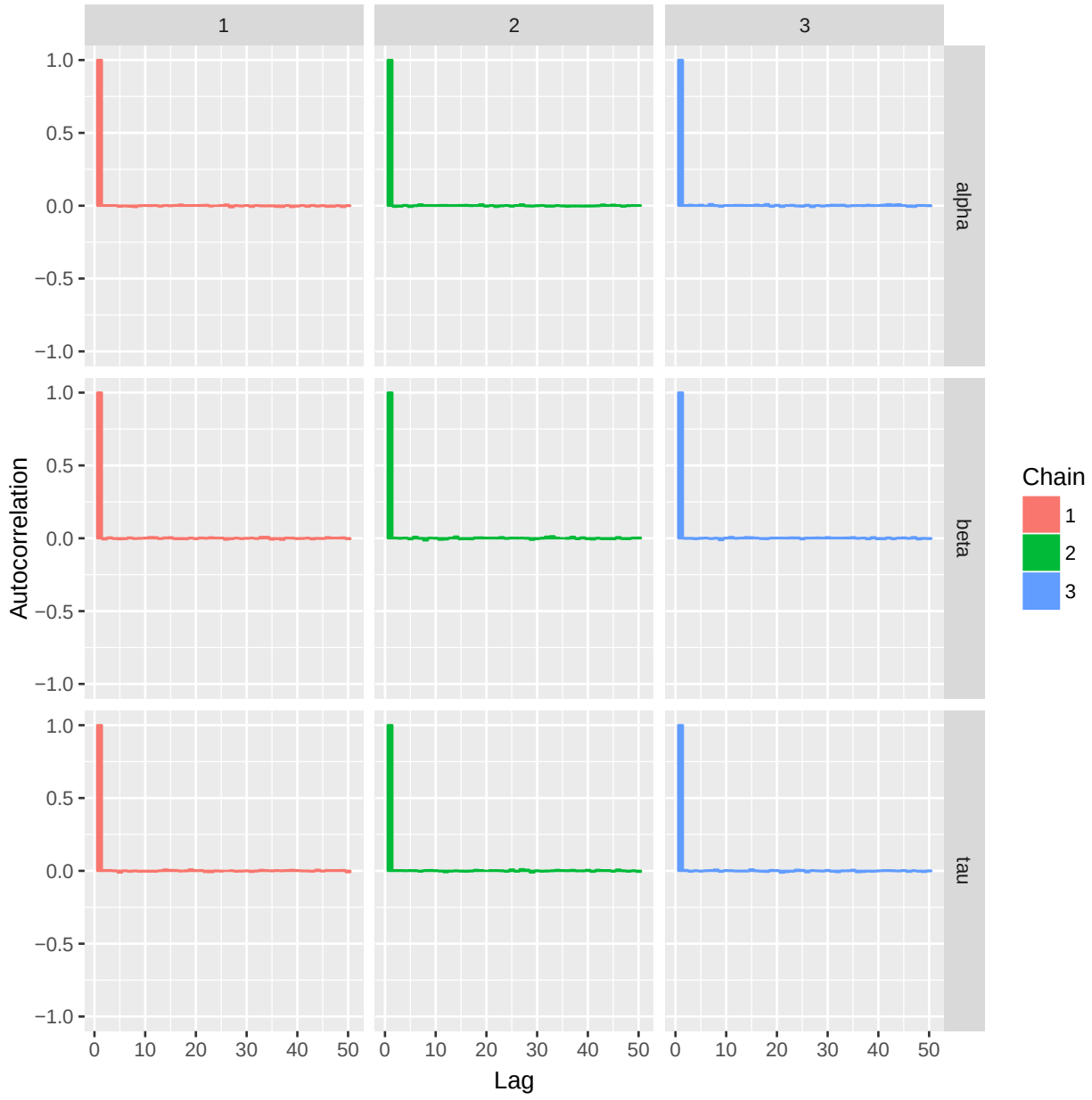

Figure 19: Simulation autocorrelations

Simulation Crosscorrelation

```
ggs_crosscorrelation(hep3.results)
```

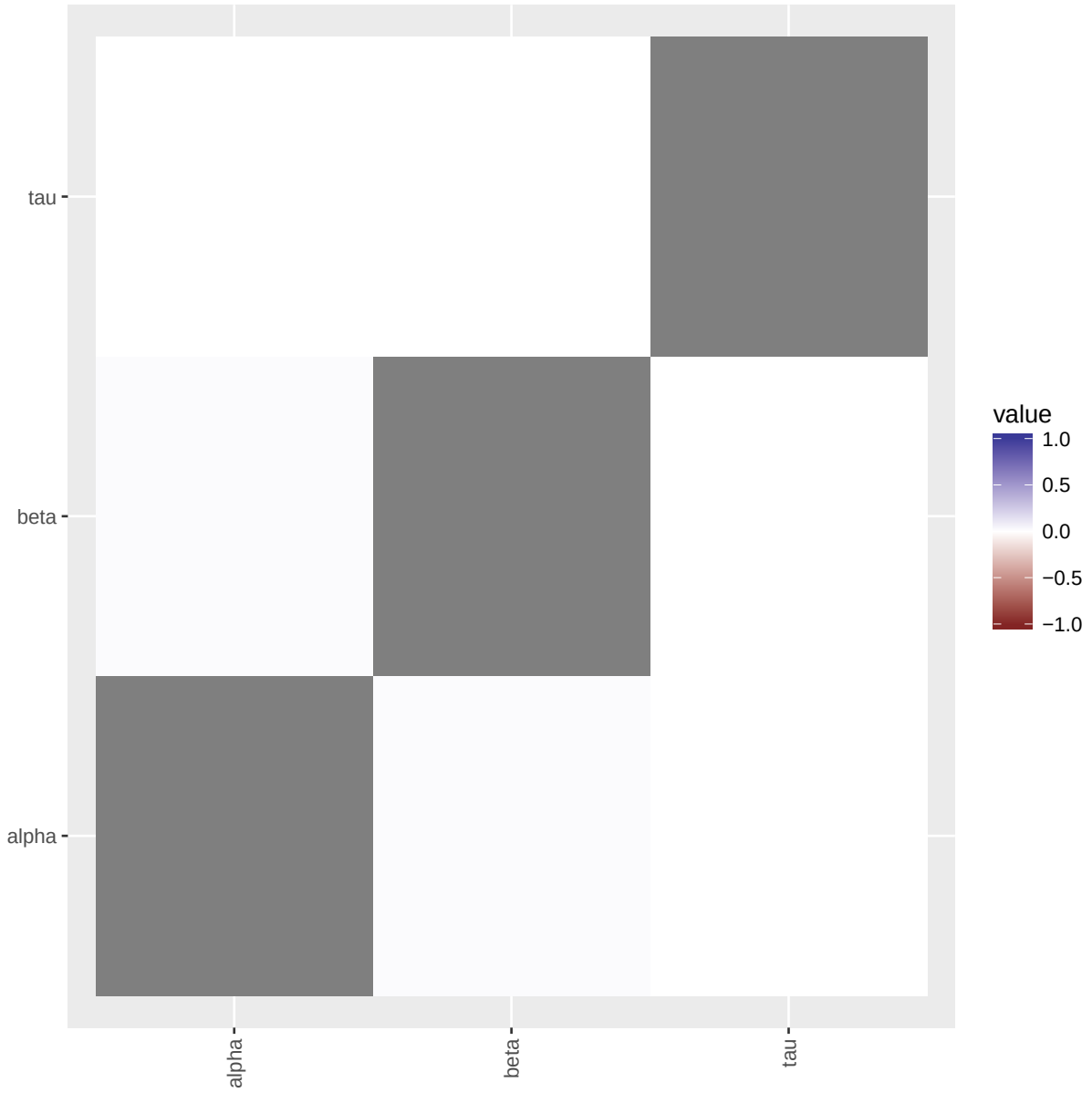

Figure 20: Simulation crosscorrelations

In this case the model works very good with optimal results to all the paramters.

In particular the autocorrelations tend incredibly to zero in a very few time.

The crosscorrelations show also a perfect uncorrelation of the parameters.

DIC comparison

Natural way to compare these models is to use criterion based on trade-off between the fit of the data to the model and the corresponding complexity of the model. Spiegelhalter et al (2002) proposed a Bayesian model comparison criterion based on this principle:

Deviance Information Criterion, DIC = "goodness of fit" + "complexity"

They measure fit via the deviance:

$$D(\theta) = 2logL(data|\theta) \tag{36}$$

Complexity measured by estimate of the "effective number of parameters":

$$p_D = E_{\theta|y}[D] - D(E_{\theta|y}[\theta]) = \bar{D} - D(\bar{\theta}) \tag{37}$$

i.e. posterior mean deviance minus deviance evaluated at the posterior mean of the parameters.

The DIC is then defined:

$$DIC = D(\bar{\theta}) + 2p_D = \bar{D} + p_D \tag{38}$$

Models with smaller DIC are better supported by the data.
We will compare 30 times the DIC of the first model with the third one using, first, the data generated by the first model and then the data generated by the third model.

Simulation from first model data

```
library(ggplot2)
library(rjags)
library(ggmcmc)

source("dataset.R")
#head(data1)

N=106
J=3
t=data$t

sigma=0.8
tau=1/(sigma^2)
alpha=6
beta=1

Y=data.frame(matrix(ncol = J, nrow = N))
```

```
inits1 <- list(alpha0 = 4, beta0 = 0, gamma = 0, tau.alpha = 1, tau.beta = 1,
tau = 1 )
inits3 <- list(alpha=4, beta = 0, tau = 1)

rip=30

set.seed(123)

dic1=rep(NA,rip)
dic3=rep(NA,rip)
dics=rep(NA,rip)

for(n in 1:rip){
  for( i in 1 : N ) {
    for( j in 1 : J ) {
      mu <- alpha + beta * (t[i,j] - 6.5)
      Y[i , j] = rnorm(n=1, mean=mu, sd=sigma)
    }
  }

  data3.eval = data
  data3.eval$Y = Y
  data3.eval$n=apply(!is.na(data3.eval$Y),1,sum)

  model1 = jags.model("modelHEP1.txt", data=data3.eval, inits=inits1, n.adapt=5000,
  n.chains=3)
  model3 = jags.model("modelHEP3.txt", data=data3.eval, inits=inits3, n.adapt=5000,
  n.chains=3)

  model1.dic = dic.samples(model1,20000)
  model3.dic = dic.samples(model3,20000)

  dic1[n] = sum(model1.dic$deviance)+sum(model1.dic$penalty)
  dic3[n] = sum(model3.dic$deviance)+sum(model3.dic$penalty)
  dics[n] = which.min(c(dic1[n], dic3[n]))

}
```

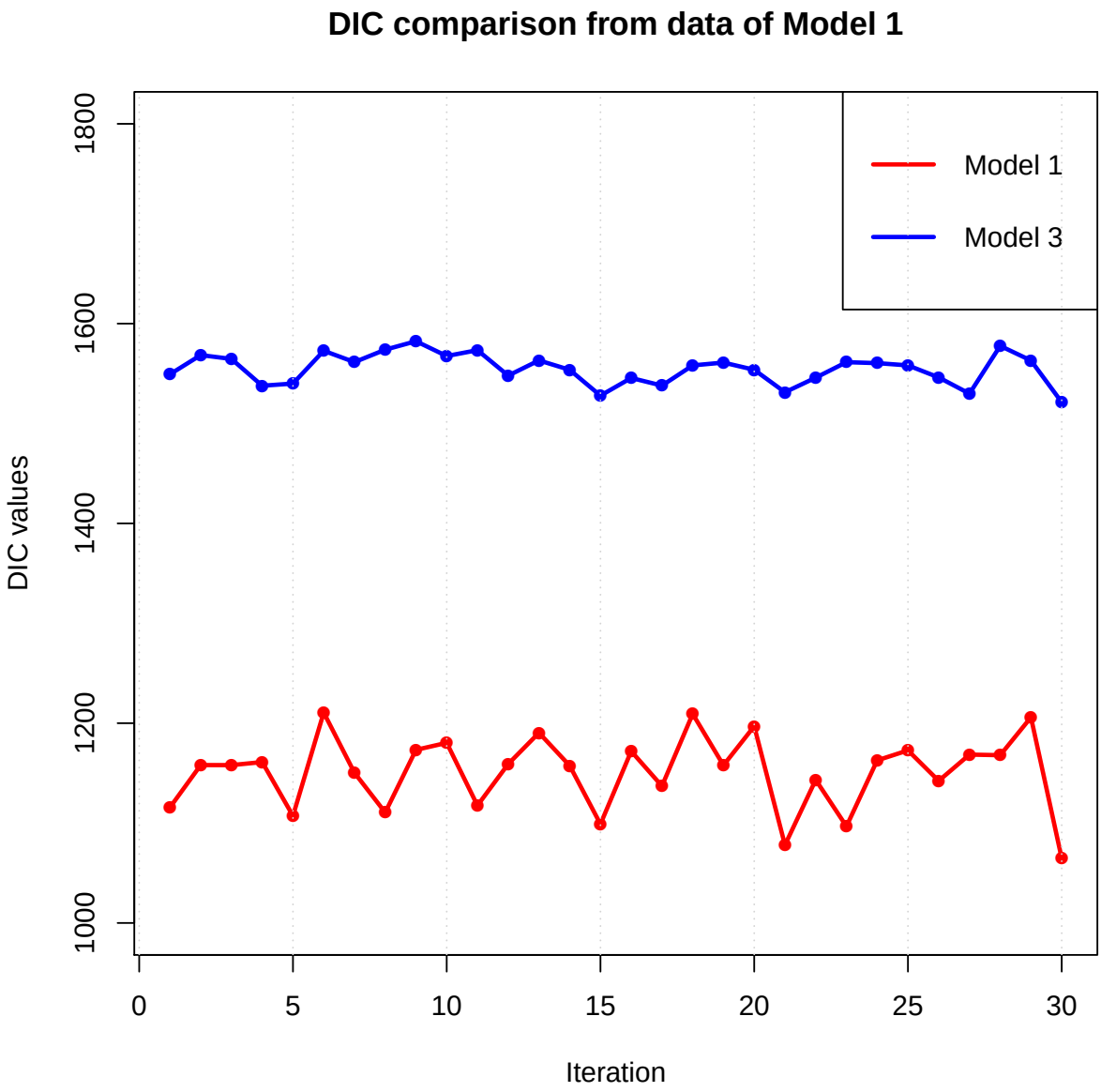

Figure 21: DIC comparison from data of Model 1

Great result for the first model where there has been a lower DIC in all the iterations. In few words model 1 perfectly fits better the data than the second one.

Simulation from third model data

```
library(ggplot2)
library(rjags)
library(ggmcmc)
```

```
source("dataset.R")

N=106
J=3
t=data$t

sigma=0.8
tau=1/(sigma^2)
alpha=6
beta=1

Y=data.frame(matrix(ncol = J, nrow = N))

inits1 <- list(alpha0 = 4, beta0 = 0, gamma = 0, tau.alpha = 1, tau.beta = 1, tau = 1 )
inits3 <- list(alpha=4, beta = 0, tau = 1)

rip=30

set.seed(123)

dic1=rep(NA,rip)
dic3=rep(NA,rip)
dics=rep(NA,rip)

for(n in 1:rip){
  for( i in 1 : N ) {
    for( j in 1 : J ) {
      mu <- alpha + beta * (t[i,j] - 6.5)
      Y[i , j] = rnorm(n=1, mean=mu, sd=sigma)
    }
  }

  data3.eval = data
  data3.eval$Y = Y
  data3.eval$n=apply(!is.na(data3.eval$Y),1,sum)

  model1 = jags.model("modelHEP1.txt", data=data3.eval, inits=inits1,
  n.adapt=5000, n.chains=3)
  model3 = jags.model("modelHEP3.txt", data=data3.eval, inits=inits3,
  n.adapt=5000, n.chains=3)

  model1.dic = dic.samples(model1,20000)
  model3.dic = dic.samples(model3,20000)
```

```
  dic1[n] = sum(model1.dic$deviance)+sum(model1.dic$penalty)
  dic3[n] = sum(model3.dic$deviance)+sum(model3.dic$penalty)
  dics[n] = which.min(c(dic1[n], dic3[n]))

}
```

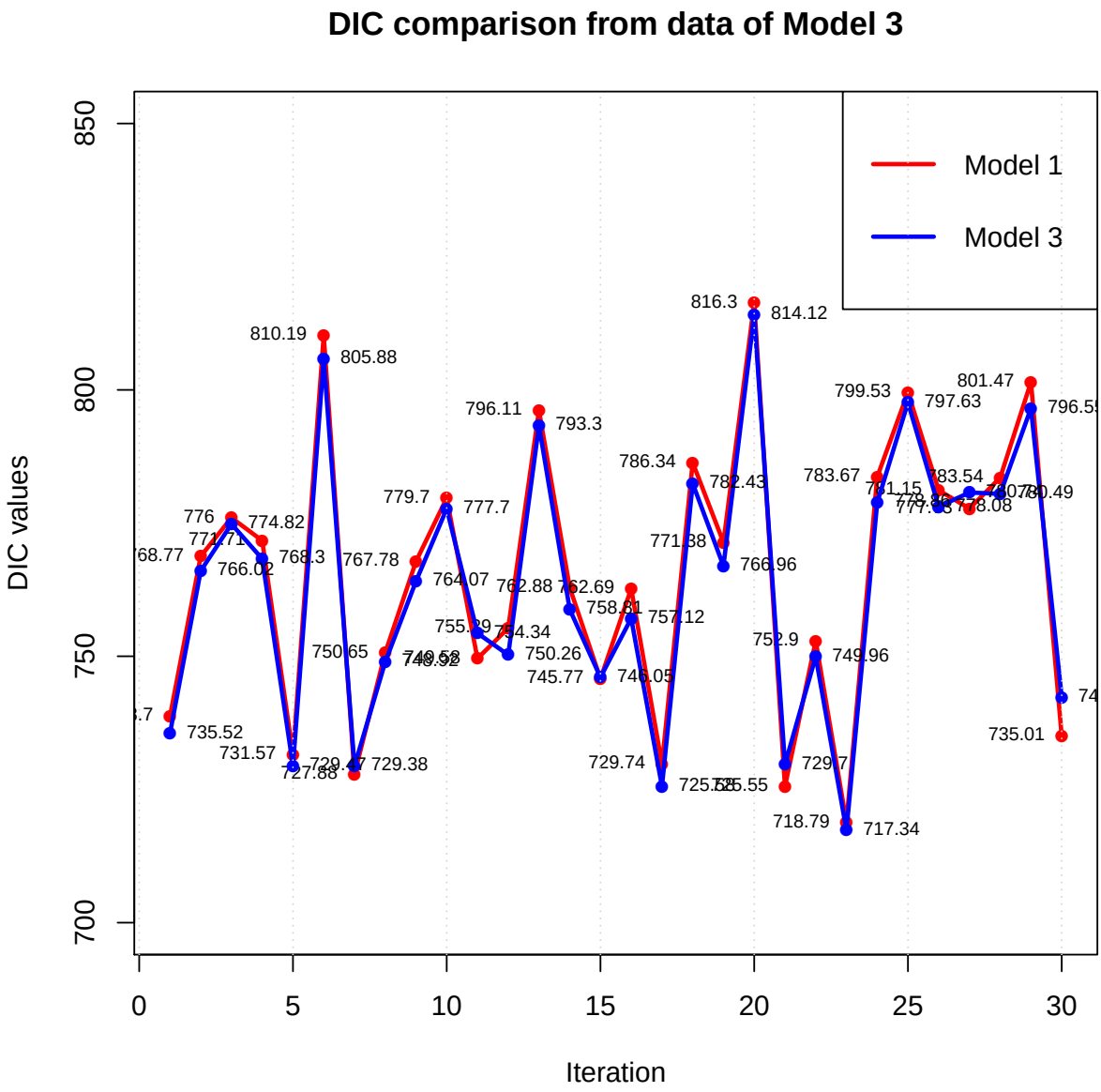

Figure 22: DIC comparison from data of Model 3

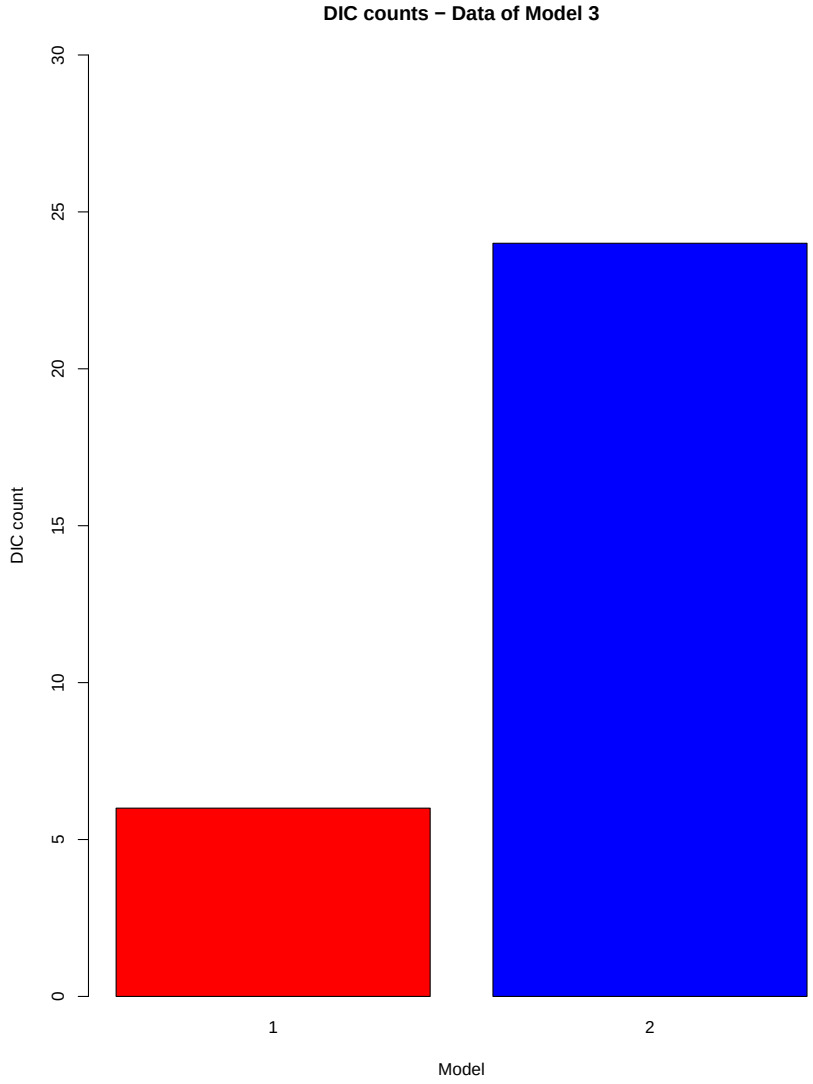

Figure 23: Counts comparison between the two models

In this case the values of the two models are instead pretty close making it difficult to identify the best model with a line plot.

In the count barplot we can find out that model 3 is certainly better, but not always, than the model 1.