SAPIENZA
UNIVERSITÀ DI ROMA

SDS II Project: Hepatitis B Study

MADIYAR TURAR

30 March 2019

Contents

Introduction

In this project, we are going to consider The Gambian Hepatitis Intervention Study (GHIS). This study is gambian national vaccination programm to reduce Hepatitis B (HB) carriers rate in population. This project aims to analyze a level of surface-antibody of HB, which is called anti-HB-titre, over time. Other studies found that after vaccination the level of HB antibody decreases with following realationship (Coursaget et al. 1991):

$$anti - HB - titre \sim \frac{1}{time}$$

where $time$ is time since final vaccination. Rewriting the formula above in log:

$$log(anti - HB - titre) = \alpha_i - log(time)$$

here, α_i is constant after final dose of vaccination and it differs for each infant i. However, gradient of $log(time)$, which is -1, was found to be common for all infants according to Coursaget study. So, in this project we are going to validate finding of Coursaget for α_i and gradient -1.

Data

The GHIS data contains HB measurement data of 106 children. After HB vaccination, blood samples from each infants were collected for checking level of anti-HB-titre. Overall, three HB tests were done with approximate interval of six months between tests. The first measurement was done right after vaccination. At the first and second test, blood samples from all 106 children were tested, and at the third test only 76 children were checked. So, there are overall 288 data points for HB (Yvec1), which is measured in milli-International-Units, and for time (tvec1),measured as days, and both are converted into log-scale.

Data,log scale

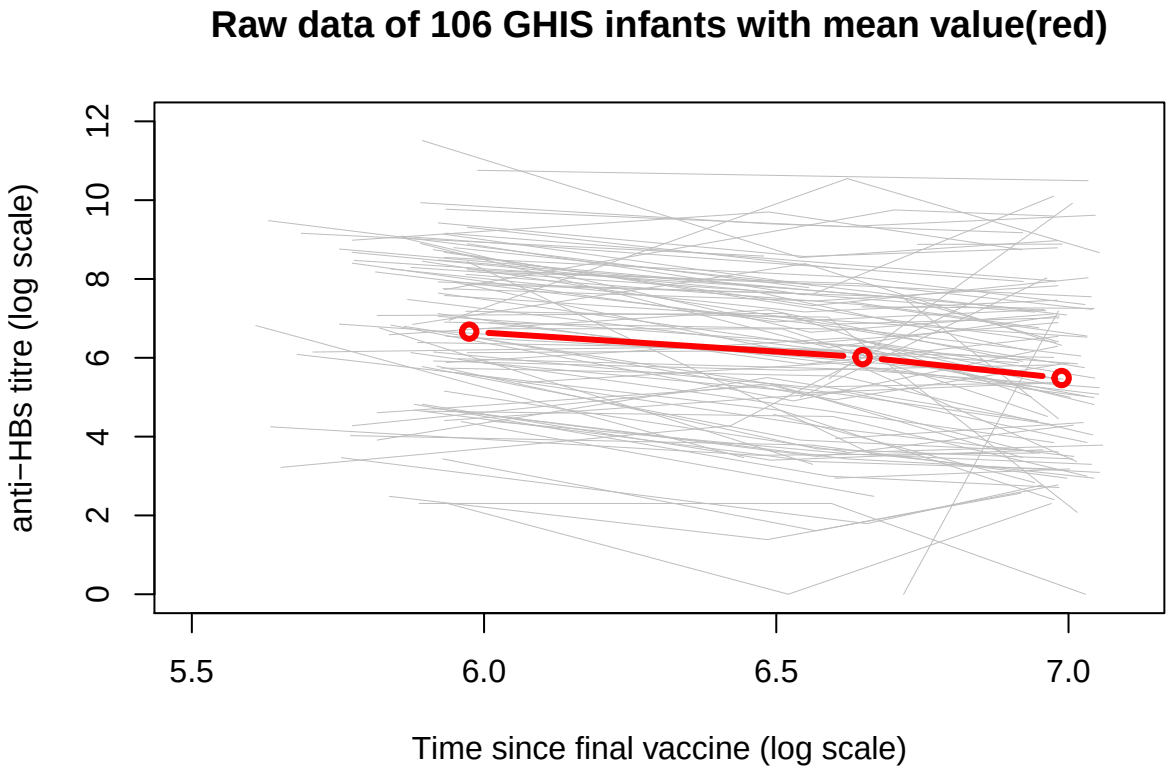
$InfantID$	$time_1$	$HBtitre_1$	$time_2$	$HBtitre_2$	$time_3$	$HBtitre_3$
1	6.54	5.00	6.96	8.03	10	NA
2	5.84	6.83	6.53	4.91	6.98	6.30
3	6.60	3.95	7.02	4.35	10	NA
...	...	...	...	...	...	...
106	5.86	9.09	6.92	7.94	10	NA

Data can be found by this link: https://github.com/stan-dev/example-models/blob/master/bugs_examples/vol3/hepatitis/hepatitis.data.R

First of all, we will plot raw data of 106 GHIS infants with seperate line for each infants anti-HB-titre measurement. Furthermore, average value based straight line was added into the plot.

```
y_mean=c(0,0,0)
t_mean=c(0,0,0)

plot(0,0,xlim = c(5.5,7.1),ylim = c(0,12),type = "n", xlab="Time since final vaccine (log scale)",
     ylab="anti-HBs titre (log scale)", main="Raw data of 106 GHIS infants with mean value(red)")
#cl <- rainbow(5)
HB_log=c()
t_log=c()
nn=c()
for (i in 1:106){
  n=2
  yvec=Yvec1[c(i,i+106)]
```

```
  time=tvec1[c(i,i+106)]


  y_mean[1]=y_mean[1]+yvec[1]
  y_mean[2]=y_mean[2]+yvec[2]

  t_mean[1]=t_mean[1]+time[1]
  t_mean[2]=t_mean[2]+time[2]
  HB_log=append(HB_log, yvec)
  t_log=append(t_log, time)

  if (i %in% idxn1[213:288])
  {
    loc=match(i,idxn1[213:288])
    yvec=append(yvec,Yvec1[212+loc])
    time=append(time, tvec1[212+loc])

    y_mean[3]=y_mean[3]+yvec[3]
    t_mean[3]=t_mean[3]+time[3]

    HB_log=append(HB_log, yvec[3])
    t_log=append(t_log, time[3])
    n=3
  }



  else{

    HB_log=append(HB_log, NA)
    t_log=append(t_log, 10)
  }

  nn=append(nn, n)


  lines(time,yvec,col='grey',type = 'l', lwd=0.5)
}
y_mean[1]=y_mean[1]/106
y_mean[2]=y_mean[2]/106
y_mean[3]=y_mean[3]/76
t_mean[1]=t_mean[1]/106
t_mean[2]=t_mean[2]/106
t_mean[3]=t_mean[3]/76
lines(t_mean,y_mean,col='red',type = 'b', lwd=3, cex=1)
```




```
#creating matrix from list for HB and time:
hb_matrix=matrix(HB_log, byrow=TRUE, nrow=106 )
time_matrix=matrix(t_log, byrow=TRUE, nrow=106)
data=list( N = 106,
           Y =hb_matrix,
           t = time_matrix,
           y0 = y0,
           times=nn)     #data created

print ("HB titre matrix:")
```

```
## [1] "HB titre matrix:"
```

```
head(data$Y)
```

```
##          [,1]     [,2]     [,3]
## [1,] 4.997000 8.028000       NA
## [2,] 6.830000 4.905000 6.295000
## [3,] 3.951244 4.356709       NA
## [4,] 6.797940 5.273000 4.317488
## [5,] 4.718499 3.496508 4.290459
## [6,] 5.789960 3.688879 3.433987
```

```r
print ("Time matrix:")
```

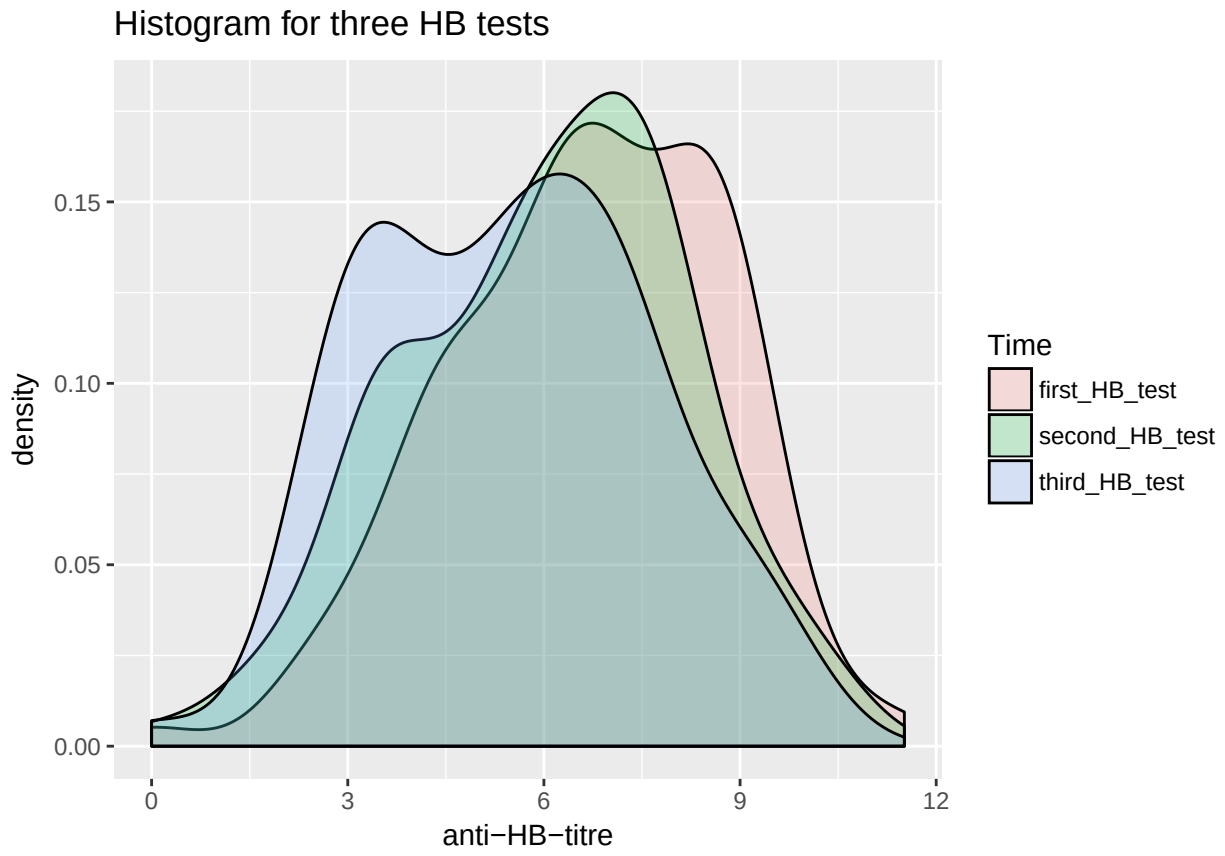
```
## [1] "Time matrix:"
```

```r
head(data$t)
```

```
##          [,1]     [,2]      [,3]
## [1,] 6.541000 6.963000 10.000000
## [2,] 5.840642 6.529419  6.981935
## [3,] 6.601230 7.025538 10.000000
## [4,] 5.863631 6.517671  6.967909
## [5,] 5.913503 6.569481  7.008505
## [6,] 5.918894 6.569481  7.007601
```

It is also possible to show as histogram the distribution of anti-HB-titre among 106 infants at each test.

```r
library(ggplot2)

first_HB_test=data.frame(Yvec1[1:106])
second_HB_test=data.frame(Yvec1[106:212])
third_HB_test = data.frame(Yvec1[213:288])

first_HB_test$n = 'first_HB_test'
second_HB_test$n = 'second_HB_test'
third_HB_test$n = 'third_HB_test'

colnames(first_HB_test)='Test'
colnames(second_HB_test)='Test'
colnames(third_HB_test)='Test'


tests <- rbind(first_HB_test, second_HB_test,third_HB_test)
colnames(tests)[2]='Time'
ggplot(tests, aes(Test, fill = Time)) + geom_density(alpha = 0.2)+ggtitle("Histogram for three HB tests
```



Model 1

In the first model, we apply random effect on α_i and gradient, since these values have variability in data set, and thus causing overdispersion. We expect that anti-HB-titre is represented as normal distribution, for infant i and at measurement j:

$$y_{ij} \sim N(\mu_{ij}, \sigma^2)$$

with mean value represented as:

$$\mu_{ij} = \alpha_i + \beta_i * (log(time_{ij}) - mean(time))$$

here, we subtract $time_{ij}$ by mean value to standartize for numerical stability. For interception α_i and gradient β_i:

$$\alpha_i \sim N(\alpha_0, \sigma_\alpha^2)$$
$$\beta_i \sim N(\beta_0, \sigma_\beta^2)$$

Considering priors, we would like the priors to be not too influential for our model, because for we don't have fairly informative priors for this data. Therefore, we model them as follows:

$$\alpha_0, \beta_0 \sim N(0, 10000)$$

and the precisions (1/variance) of all normal distributions have priors as gamma distributions:

$$\sigma^{-2}, \sigma_\alpha^{-2}, \sigma_\beta^{-2} \sim \Gamma(0.001, 0.001)$$

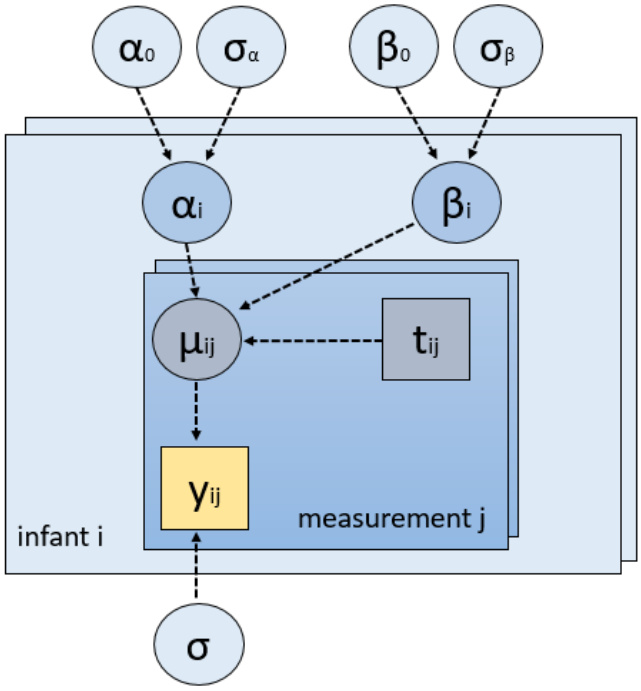


Figure 1: DAG of model 1

Bugs Code of Model 1

```
  model
{
   for( i in 1 : N ) {
      for( j in 1 : times[i] ) {
         Y[i , j] ~ dnorm(mu[i , j],tau)
         mu[i , j] <- alpha[i] + beta[i] * (t[i,j] - 6.48)

      }
      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-4)
   tau.alpha ~ dgamma(0.001,0.001)
   beta0 ~ dnorm(0.0,1.0E-4)
   tau.beta ~ dgamma(0.001,0.001)

}
```

Simulation of Model 1

Using JAGS packages, we simulate our model 1. During MCMC, we are interested in five parameters of the model. Chains number is selected as 2, so model parameteres will be checked for convergence twice. Furthermore, number of iterations is chosen as 11000 with 1000 burn-in, which allows to get more stationary results.

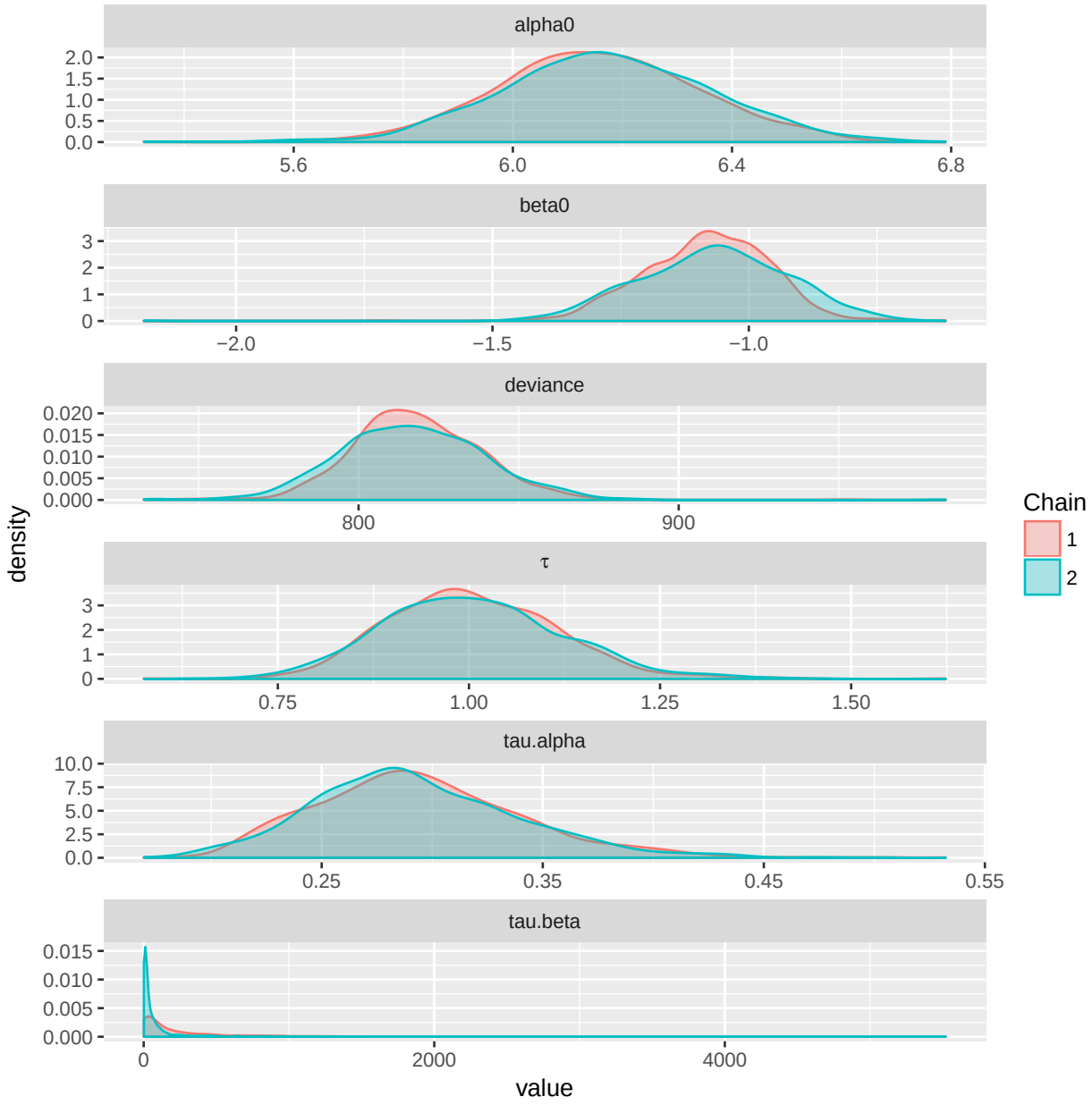
```
initial1=list(alpha0 = 3, beta0 = -2,  tau.alpha = 2, tau.beta = 2, tau = 2)
initial2=list(alpha0 = 4, beta0 = -3,  tau.alpha = 1, tau.beta = 1, tau = 1)
inits=list(initial1, initial2)
set.seed(123)
parameters=c("alpha0","beta0","tau.alpha","tau.beta","tau")
model1=jags(data=data, inits=inits,
            parameters.to.save=parameters,
            model.file="model1.txt",
            n.chains=2,
            n.burnin=1000,
            n.iter=11000)
output=head(model1)
```

After the simulation, α_0 has mean value 6.2 and β_0 is -1.1, which is close to finding of Coursaget. Furthermore, it should be noticed that alpha has higher variance meaning that interception coefficient differs for each infants, while gradient coefficient has higher precision indicating to be a common for all infants. Other important values are Rhat (potential scale reduction factor, for convergence <1.001) and n.eff (measure of effective sample size). The parameter τ_β shows non-converged behaviour over iterations according to Rhat. So, let's explore further what happens with each parameter during the simulation.

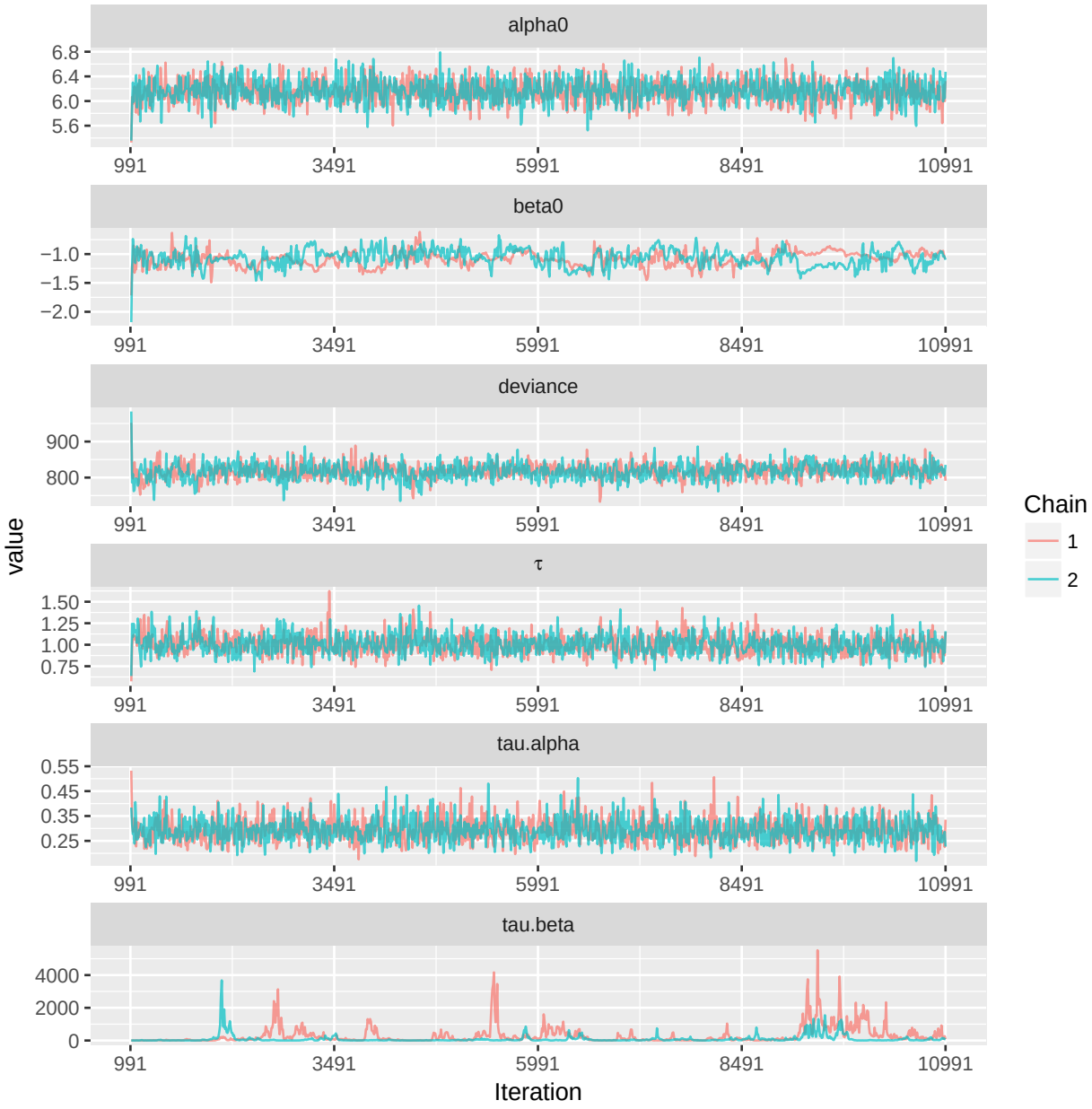
```
output$BUGSoutput
```

```
## Inference for Bugs model at "model1.txt", fit using jags,
##  2 chains, each with 11000 iterations (first 1000 discarded), n.thin = 10
##  n.sims = 2000 iterations saved
##            mean    sd  2.5%   25%   50%   75%  97.5% Rhat n.eff
## alpha0      6.2   0.2   5.8   6.0   6.2   6.3    6.5  1.0   540
## beta0      -1.1   0.1  -1.3  -1.2  -1.1  -1.0   -0.8  1.0   410
## deviance  817.4  21.6 778.2 803.3 816.6 831.4  860.8  1.0   890
## tau         1.0   0.1   0.8   0.9   1.0   1.1    1.2  1.0  2000
## tau.alpha   0.3   0.0   0.2   0.3   0.3   0.3    0.4  1.0  2000
## tau.beta  195.4 422.2   2.2  10.6  44.2 168.8 1320.8  1.3    10
##
## 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 = 234.2 and DIC = 1051.6
## DIC is an estimate of expected predictive error (lower deviance is better).
```

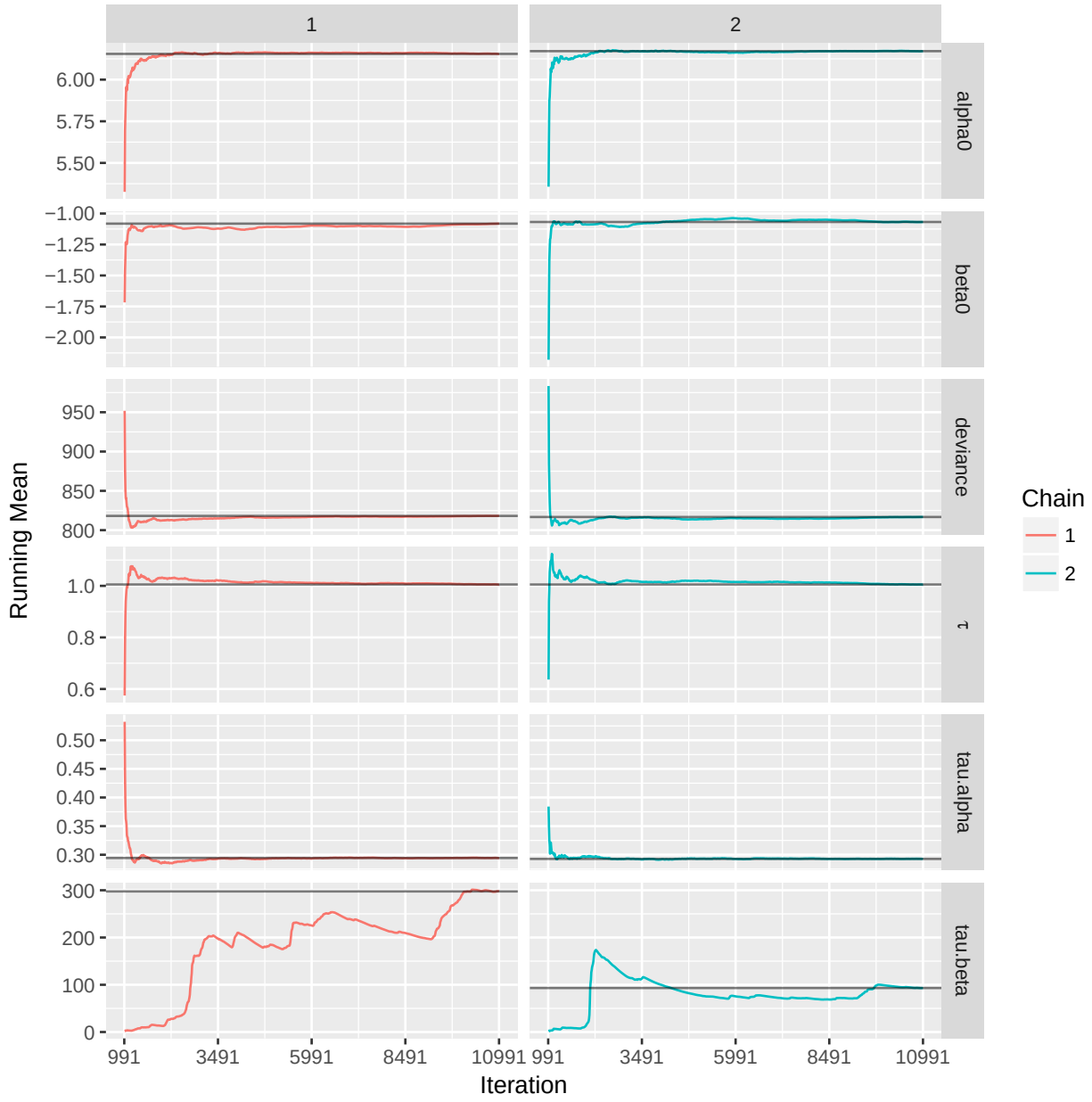
```
data_plots = ggs(as.mcmc(model1))
ggs_density(data_plots, greek=T)
```



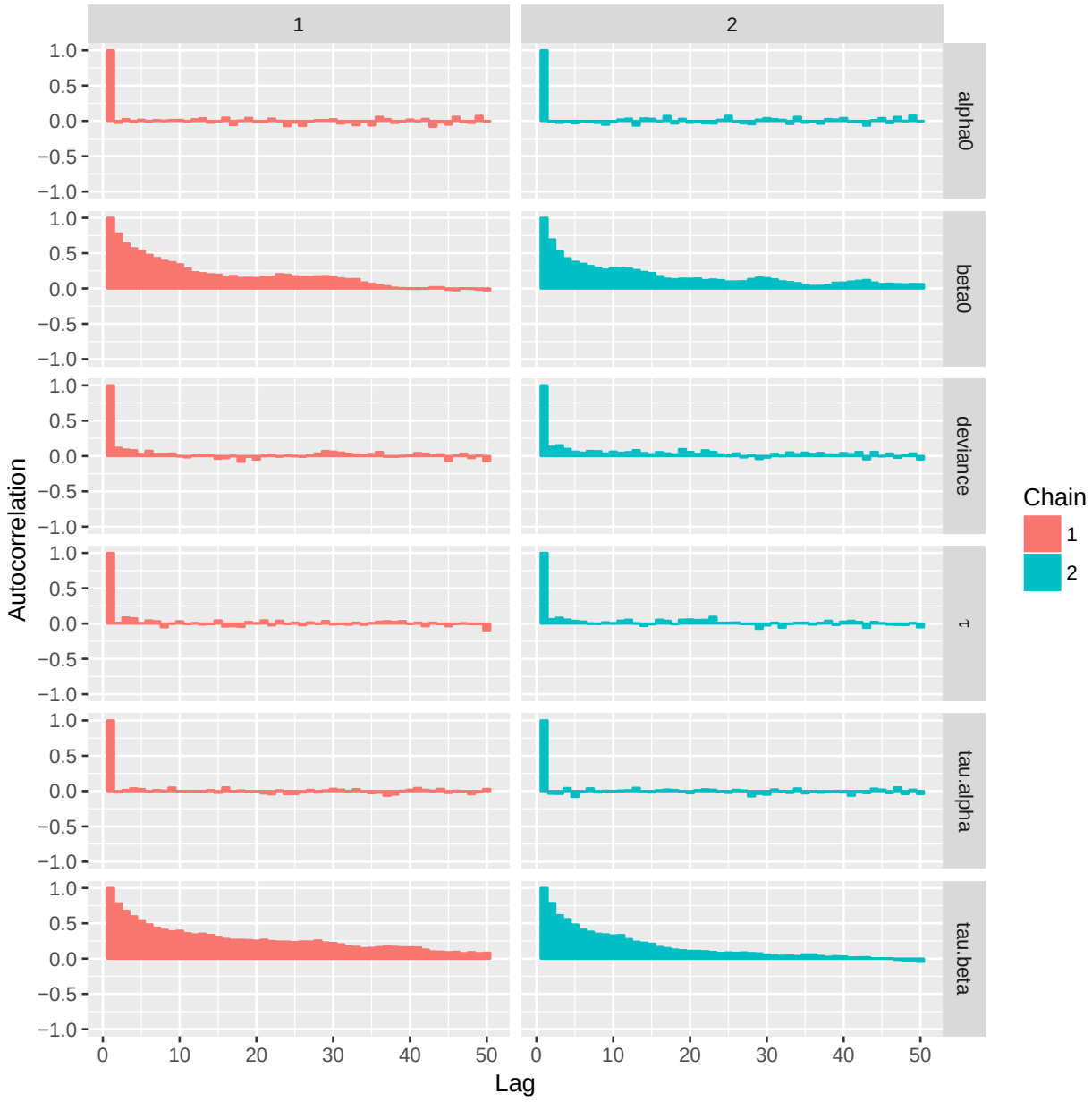

```
ggs_traceplot(data_plots, greek=T)
```



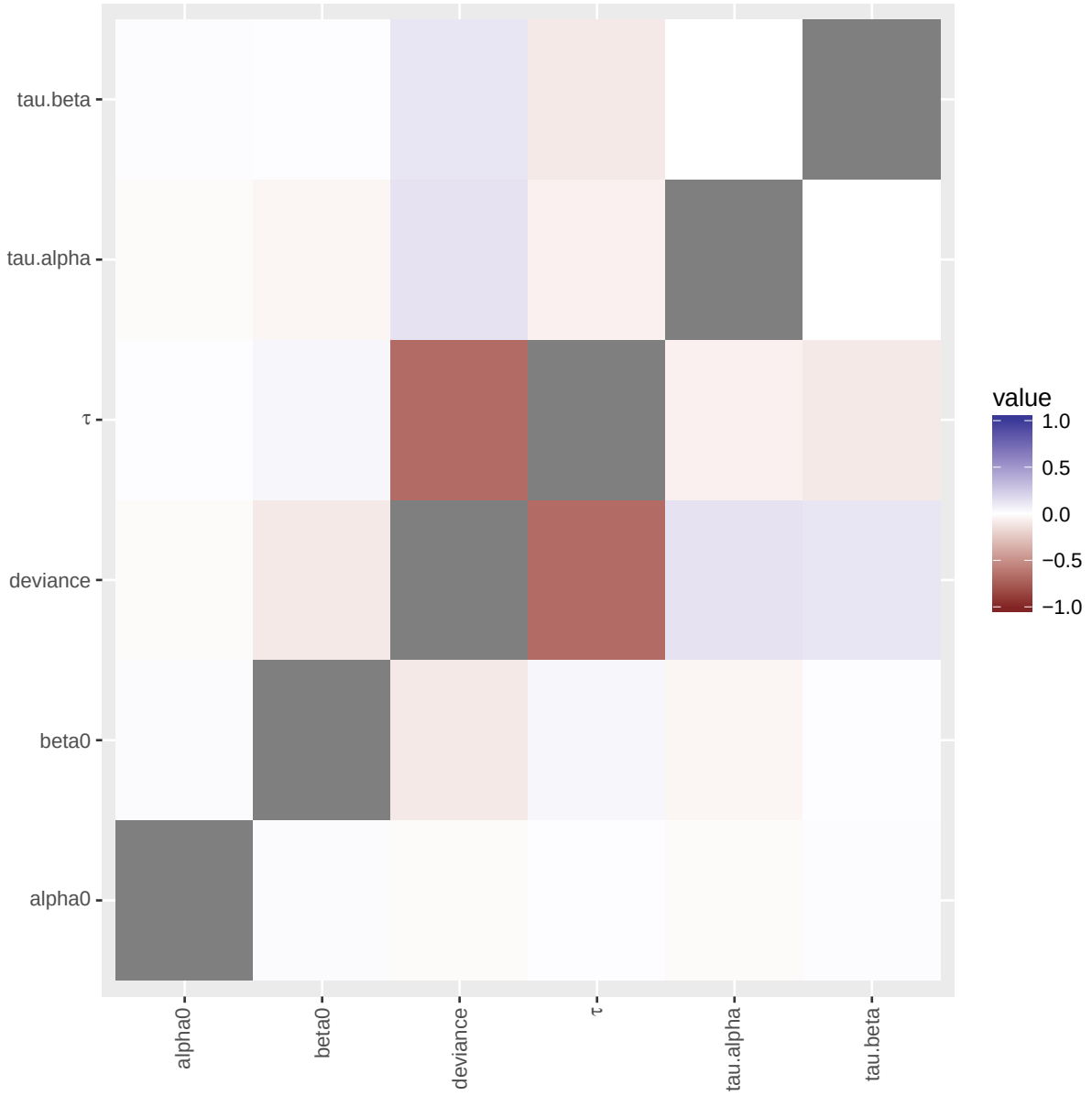

```
ggs_running(data_plots, greek=T)
```




```
ggs_autocorrelation(data_plots,greek=T)
```




```
ggs_crosscorrelation(data_plots, greek=T)
```



Trace-plot shows convergence of most of the parameters to their stationary point after burn-in, except τ_β, which has some fluctuations.

Running plot displays how mean value of the parameters changed over iterations. In this plot, we can see that τ_β had some difficulties in approaching stationary mean value, while others quickly converged.

Autocorrelation plot shows how the model parameters are Markovians, meaning no long memory process. In this case, τ_β and β_α has some noticable correlations.

In crosscorelation plot, all parameters don't have significant correlations between each other.

Frequentist Analysis

In frequency analysis, we apply a linear regresion model for each infant i:

```r
sdd=mean(tvec1)
sd_time=data$t-sdd
intercepts <- rep(NA, N)
slopes <- rep(NA, N)
for(i in 1:N){
  fit <- lm(data$Y[i,]~ sd_time[i,])
  intercepts[i] <- coefficients(fit)[[1]]
  slopes[i] <- coefficients(fit)[[2]]
}
mean_alpha <- mean(intercepts)
mean_beta <- mean(slopes)

print("Mean alpha:")
```

```
## [1] "Mean alpha:"
```

```r
print(mean_alpha)
```

```
## [1] 5.968885
```

```r
print("Mean beta:")
```

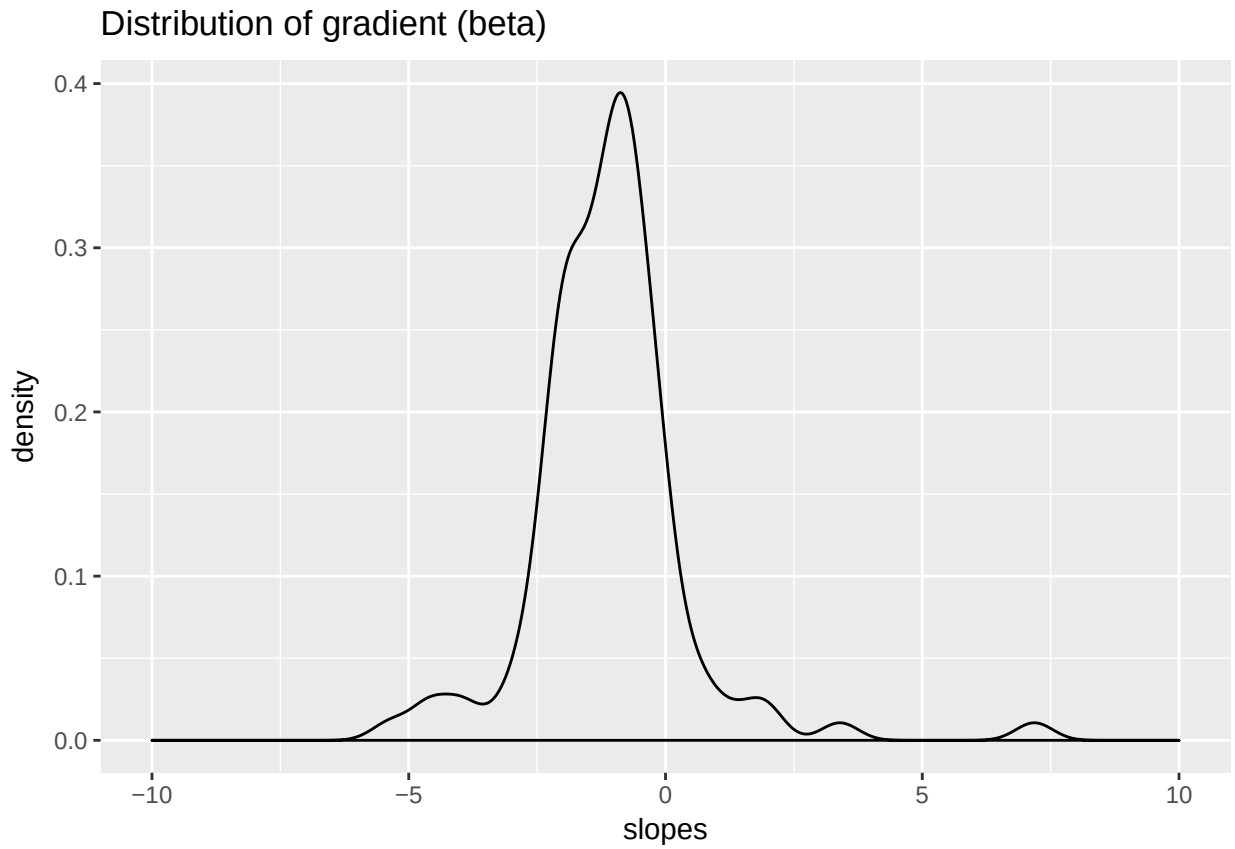
```
## [1] "Mean beta:"
```

```r
print(mean_beta)
```

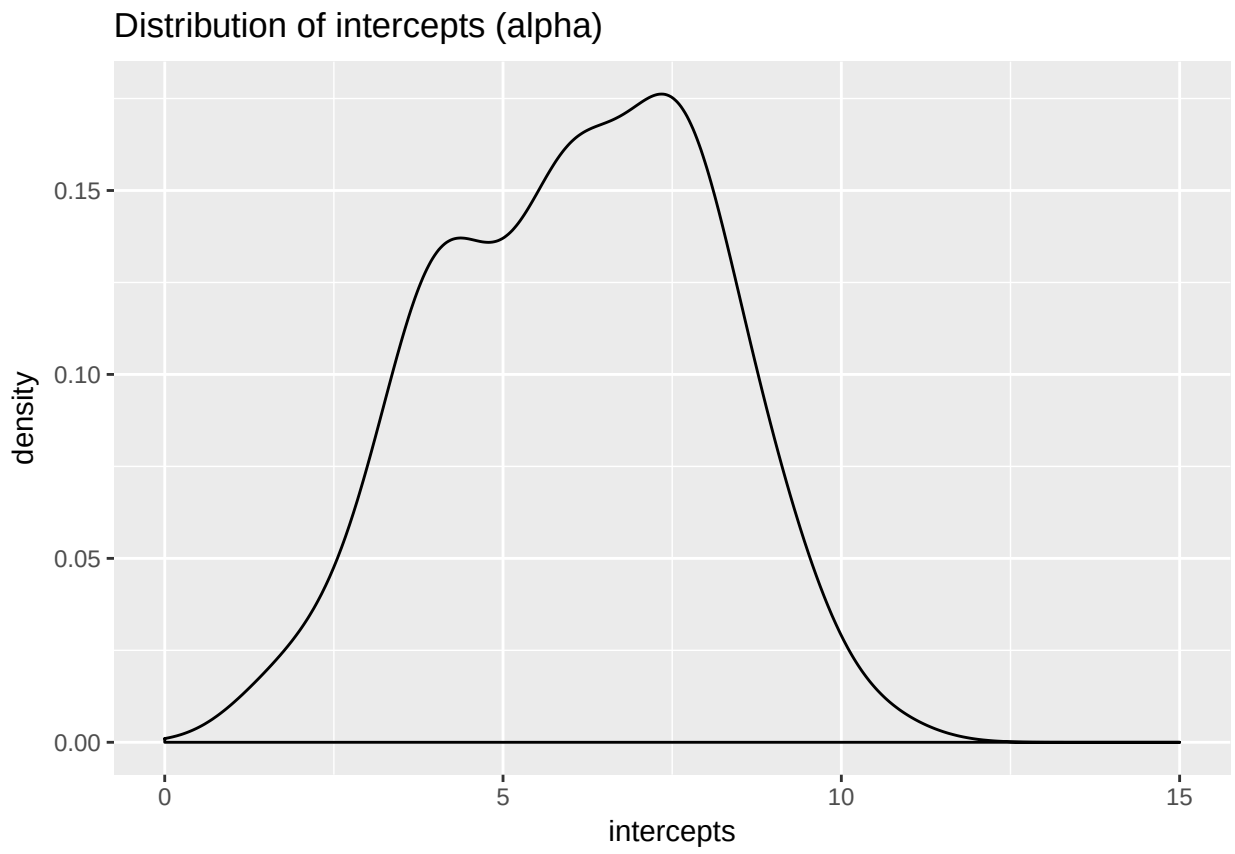
```
## [1] -0.7280098
```

Now, let's see distribution of slopes:

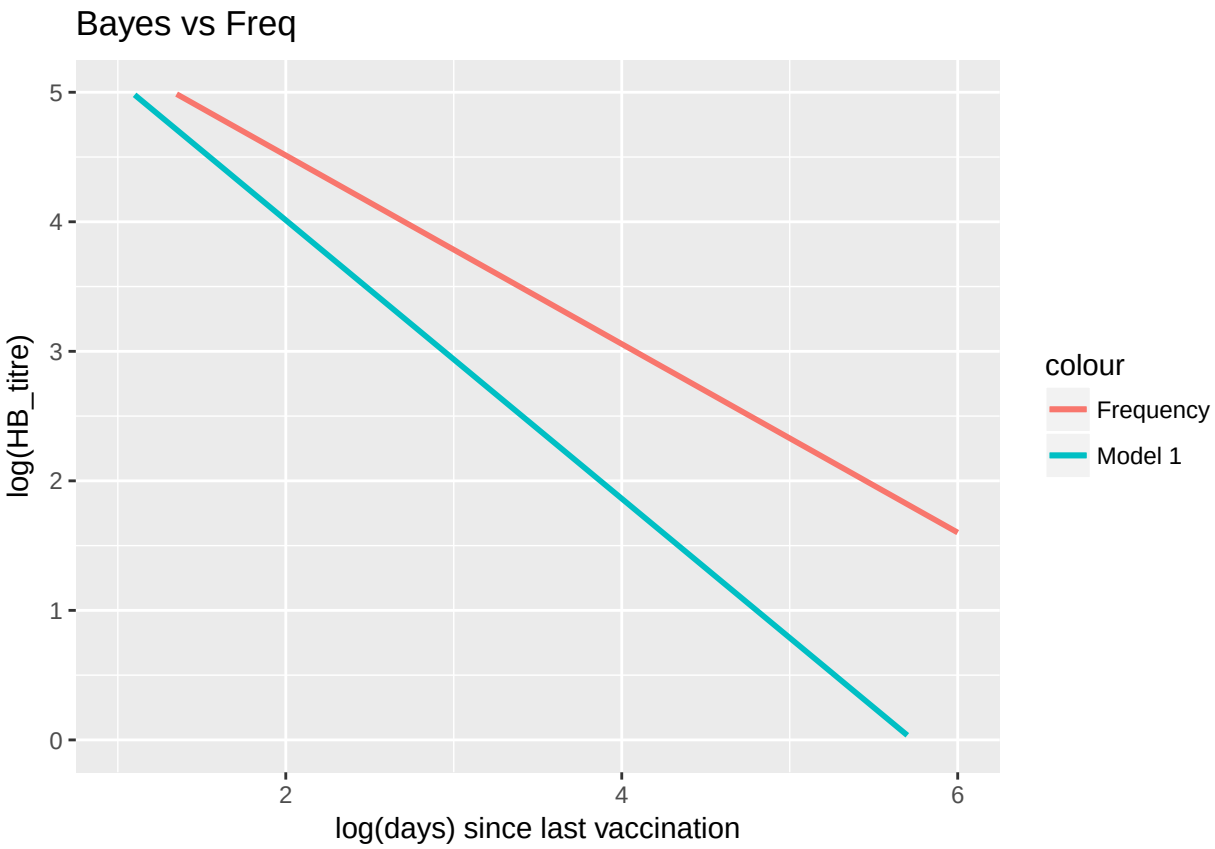
```r
qplot(slopes, geom="density", xlim=c(-10,10),main="Distribution of gradient (beta) ")
```




```
qplot(intercepts, geom="density", xlim=c(0,15),main="Distribution of intercepts (alpha) ")
```

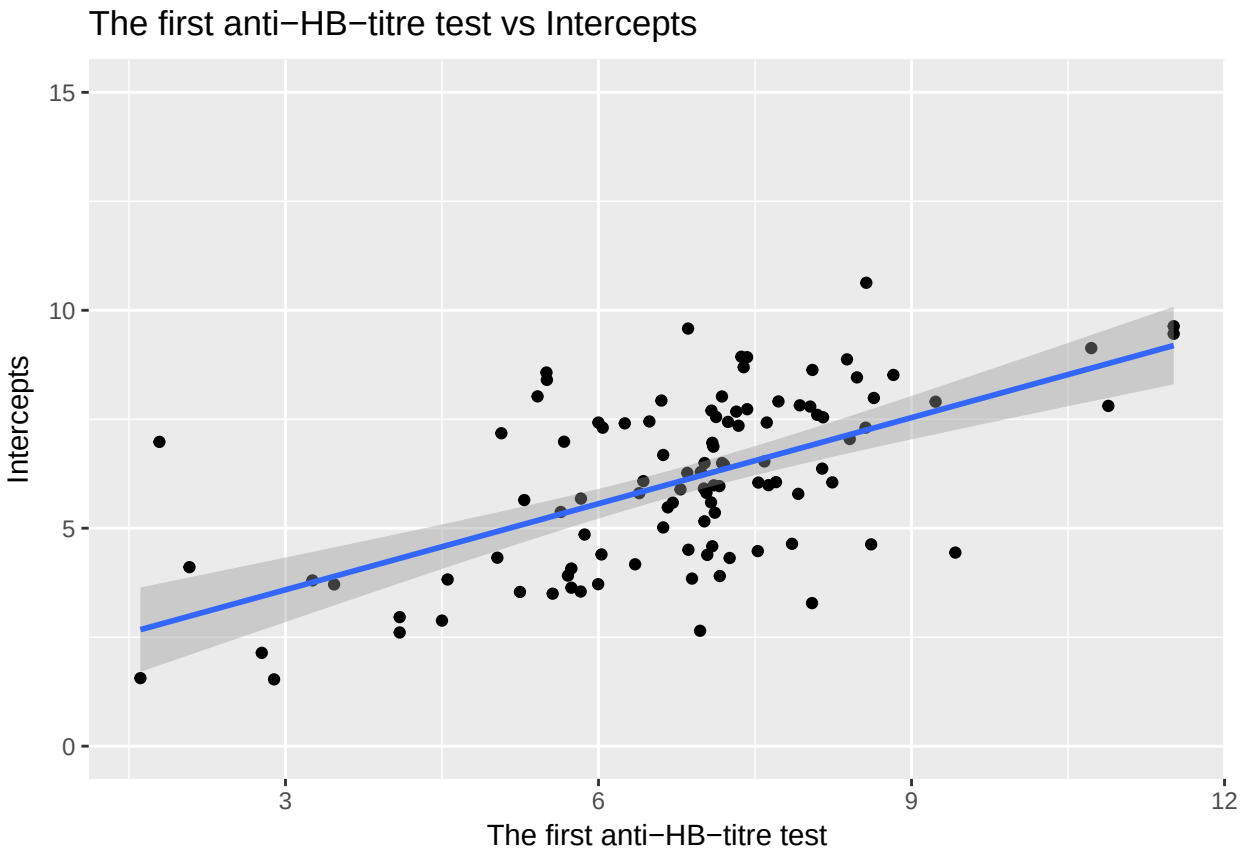



```
plt <- ggplot(NULL) + aes(x=c(1,6)) +
  theme_grey() +
  labs(x='log(days) since last vaccination',y='log(HB_titre)', title='Bayes vs Freq')+ylim(0,5)
regression = function(x, alpha, beta) alpha+beta*x

plt <- plt + stat_function(fun=regression,
                           args=list(alpha=mean_alpha,beta=mean_beta),
                           size=1, alpha=1,aes(colour = "Frequency"))
plt <- plt + stat_function(fun=regression,
                           args=list(alpha=output$BUGSoutput$mean$alpha0,
                                     beta=output$BUGSoutput$mean$beta0),
                           size=1, alpha=1,aes(colour = "Model 1"))

plt
```



Let's observe relationship of intercepts and the first measurement results:

```
qplot(y0, intercepts, xlab="The first anti-HB-titre test", ylab="Intercepts", main = "The first anti-HB
```



From this plot, we see that Intercepts have approximately linear relationship with the first anti-HB-titre test results indicating to predisposition of base HB titre to high subequent titres. Considering it, new model can be proposed further.

Model 2

From the previous plot we saw that there is effect of base HB-titre result on subsequent titres results. Therefore, model 2 will have a following form:

$$y_{ij} \sim N(\mu_{ij}, \sigma^2)$$
$$\mu_{ij} = \alpha_i + \gamma * (y_{i0} - mean(y_0)) + \beta_i * (log(time_{ij}) - mean(time))$$

here y_{i0} is base HB-titre result of infant i, and $mean(y_0)$ is the mean value of all base HB-tire results. Subtraction between them is proposed from a point that it will reduce correlation between γ and other parameters of the model (Spiegelhalter et al.). Indeed, if we simulate MCMC without centering base HB-titre, we find that α_0 which is term for α_i, has high negative correlation with γ (see figure below). Correlation between parameters are not prefered in Gibbs Sampling, since it will cause slow mixing.

The interception and gradient will have the same distribution as in model 1:

$$\alpha_i \sim N(\alpha_0, \sigma_\alpha^2)$$
$$\beta_i \sim N(\beta_0, \sigma_\beta^2)$$

Their priors will be the same, and also prior for γ will have non-informative normal distribution:

$$\alpha_0, \beta_0, \gamma \sim N(0, 10000)$$
$$\sigma^{-2}, \sigma_\alpha^{-2}, \sigma_\beta^{-2} \sim \Gamma(0.001, 0.001)$$

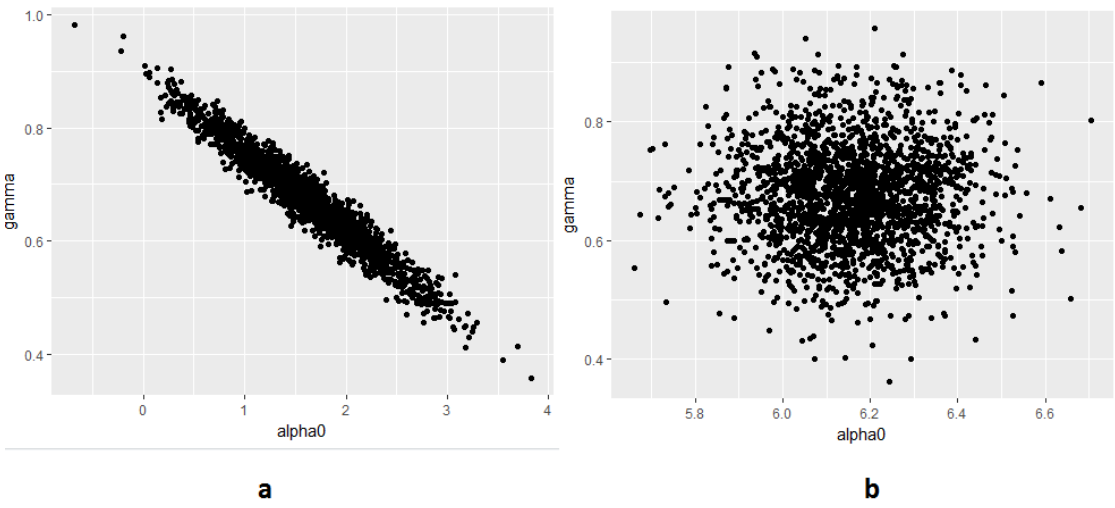


Figure 2: Joint Distribution of alpha0 and gamma: a- before centering, b- after centeting

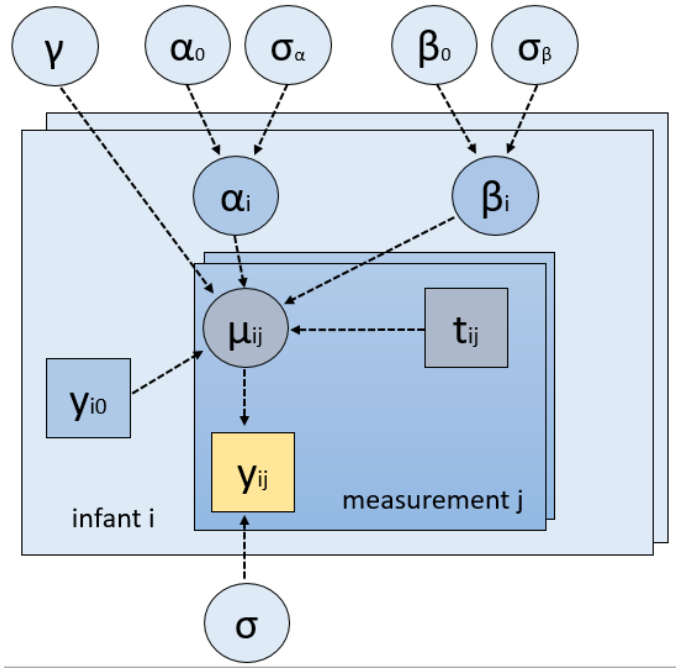


Figure 3: DAG of model 2

Bugs Code of Model 2

```
  model
{
   for( i in 1 : N ) {
      for( j in 1 : times[i] ) {
         Y[i , j] ~ dnorm(mu[i , j],tau)
         mu[i , j] <- alpha[i] + beta[i] * (t[i,j] - 6.48) +
                     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-4)
   tau.alpha ~ dgamma(0.001,0.001)
   beta0 ~ dnorm(0.0,1.0E-4)
   tau.beta ~ dgamma(0.001,0.001)
   gamma ~ dnorm(0.0,1.0E-4)
}
```

Simulation of Model 2

Simulation parameters will be the same as in model 1, that is, number of iterations is 11000 with 1000 burn-in, and 2 chains are set.

```
initial1_2=list(alpha0 = 3, beta0 = -2,  tau.alpha = 2, tau.beta = 2, tau = 2, gamma=2)
initial2_2=list(alpha0 = 4, beta0 = -3,  tau.alpha = 1, tau.beta = 1, tau = 1, gamma=1)
inits_2=list(initial1_2, initial2_2)
set.seed(111)
parameters_2=c("alpha0","beta0","tau.alpha","tau.beta","tau", "gamma")
model2=jags(data=data, inits=inits_2,
            parameters.to.save=parameters_2,
            model.file="model2.txt",
            n.chains=2,
            n.burnin=1000,
            n.iter=11000)
output2=head(model2)
```

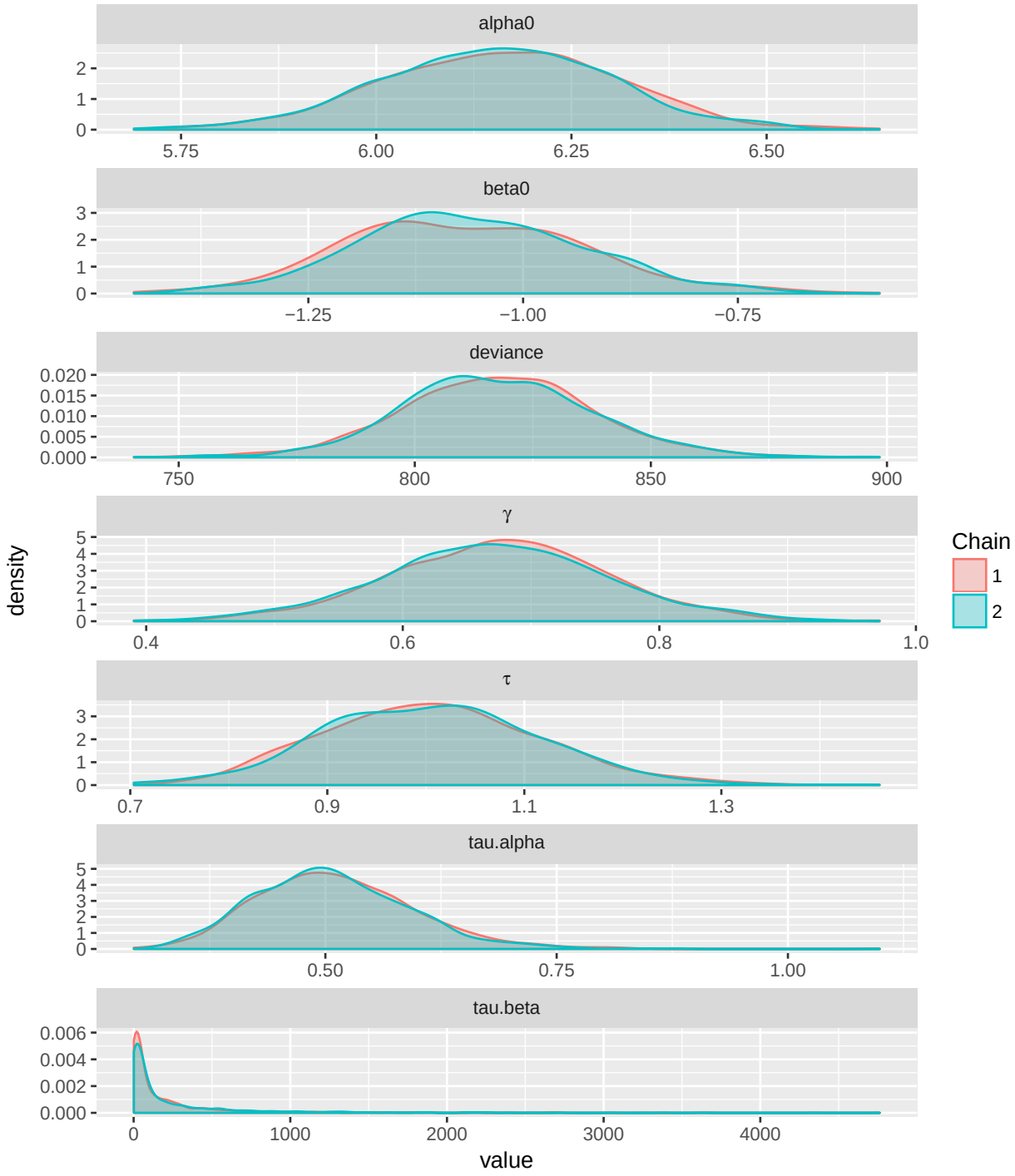
Frome summary of the simulation, α_0 and β_0 haven't changed being equal to the results in model 1. For τ_β we still see high precision, while τ_α has higher variance. Furthermore, one important improvement in model 2 is convergence of all values accroding to Rhat, since all of them are equal to 1. Good results also can be seen in n.eff outputs. From this step, let's investigate other results.

```
output2$BUGSoutput
```

```
## Inference for Bugs model at "model2.txt", fit using jags,
##  2 chains, each with 11000 iterations (first 1000 discarded), n.thin = 10
##  n.sims = 2000 iterations saved
##            mean    sd  2.5%   25%   50%   75%  97.5% Rhat n.eff
## alpha0      6.2   0.1   5.9   6.1   6.2   6.3    6.4    1  1000
## beta0      -1.1   0.1  -1.3  -1.2  -1.1  -1.0   -0.8    1  1400
## deviance  817.6  20.5 776.9 804.3 817.3 830.7  859.2    1  2000
## gamma       0.7   0.1   0.5   0.6   0.7   0.7    0.8    1  1900
## tau         1.0   0.1   0.8   0.9   1.0   1.1    1.2    1  2000
## tau.alpha   0.5   0.1   0.4   0.5   0.5   0.6    0.7    1   920
## tau.beta  215.4 422.4   2.3  13.3  50.3 223.6 1390.7    1   370
##
## 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)
```

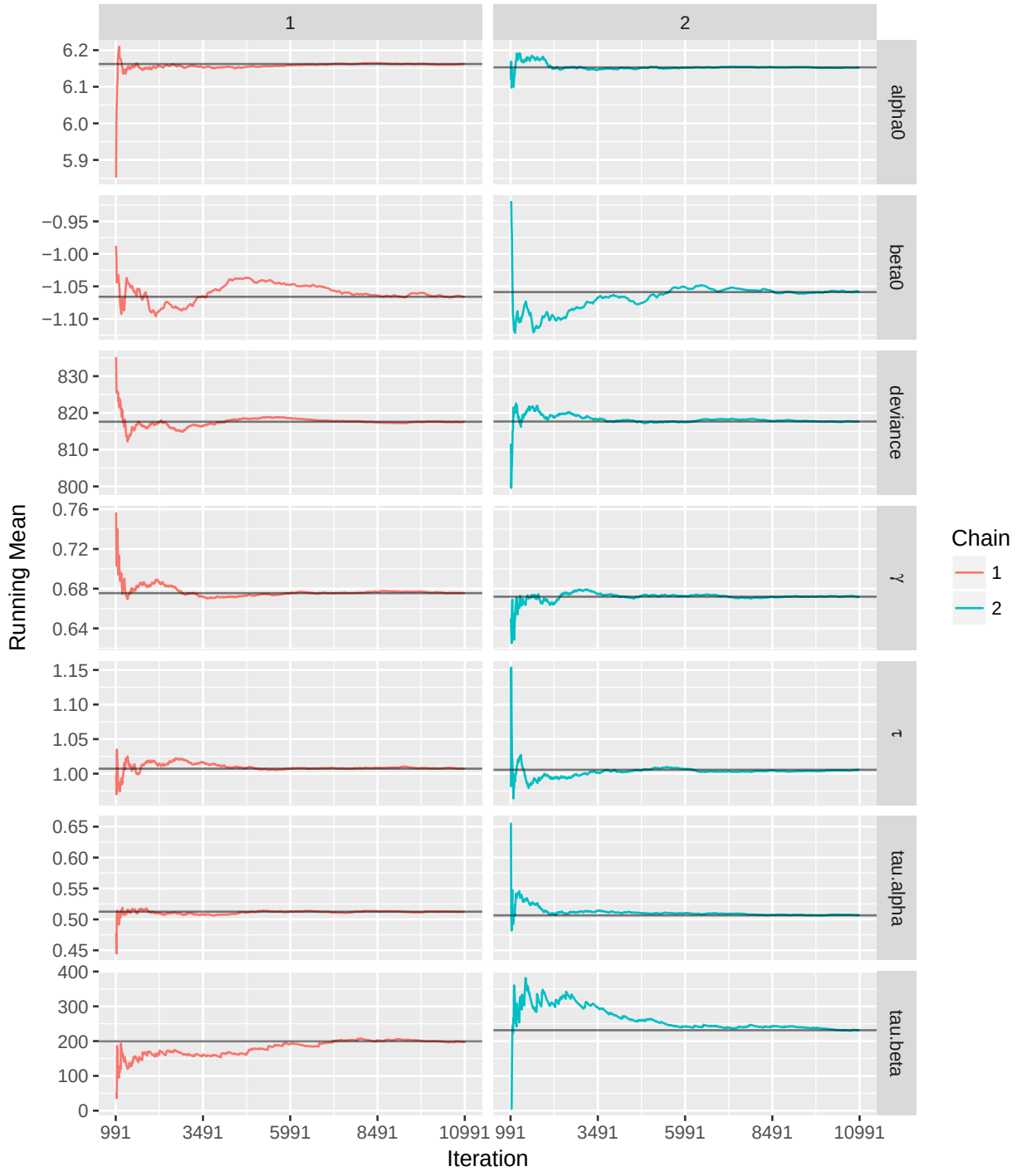
```
## pD = 210.4 and DIC = 1028.0
## DIC is an estimate of expected predictive error (lower deviance is better).
```

```
data_plots = ggs(as.mcmc(model2))
ggs_density(data_plots, greek=T)
```



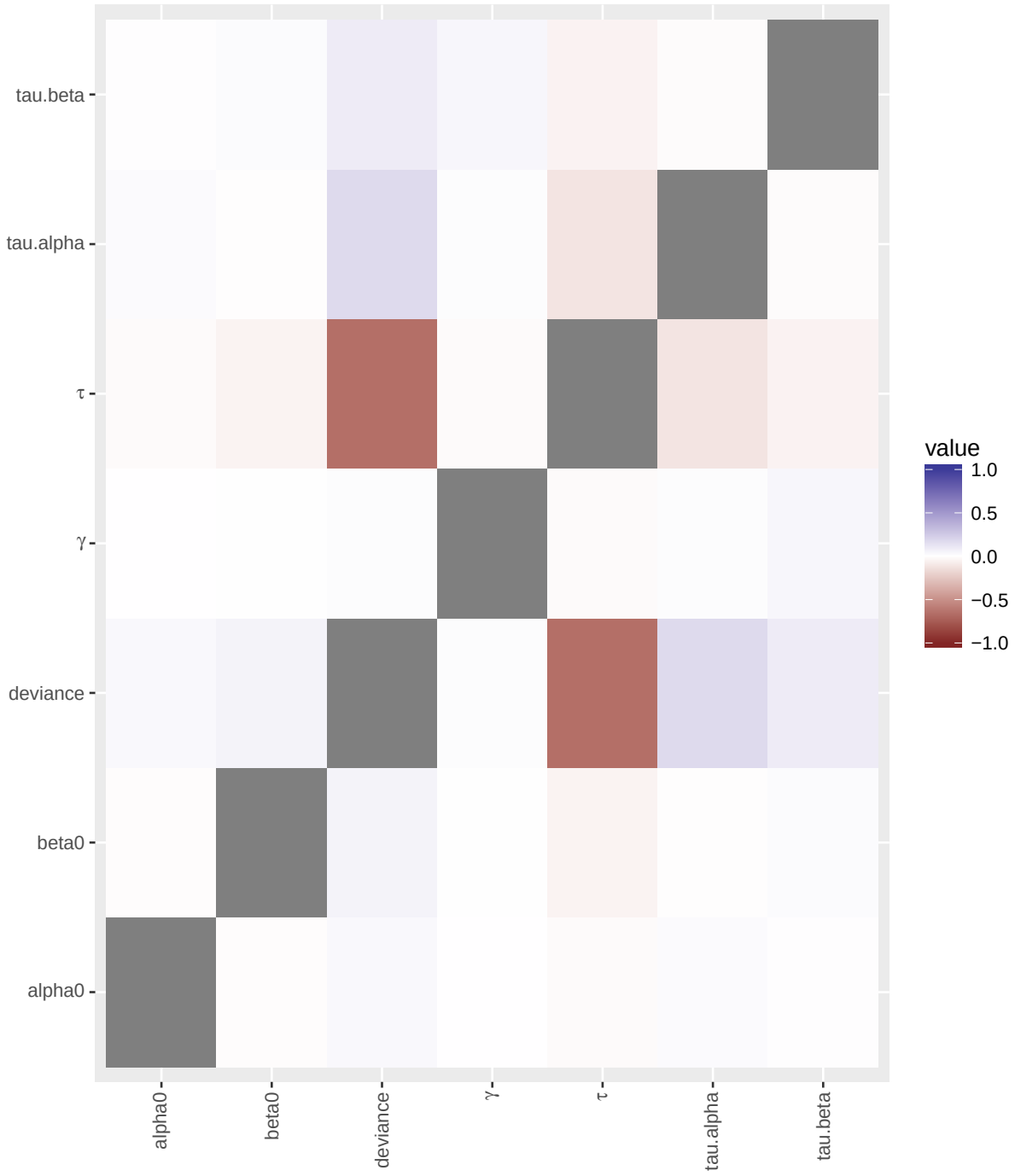
```
ggs_traceplot(data_plots, greek=T)
```


```
ggs_running(data_plots, greek=T)
```




```
ggs_autocorrelation(data_plots,greek=T)
```


```
ggs_crosscorrelation(data_plots, greek=T)
```



In trace plot, we can see that all parameters stayed around at their stationary values, despite some fluctuations, especially for τ_{beta}.

In running plot, γ and τ_{beta} had some diffulties in converging at inital steps, but after 10000 iterations, they all converged succefully.

In autocorreltion plot, all values shows no long memory behaviour.

Significant correlation between parameters of the model are not observed in the last plot.

Comparing with model 1, all parameters of model 2 perform well.

Model 1 vs. Model 2

Since we are going to compare two bayesian models derived from MCMC simulation, good comparison method will be Deviance Information Criterion. This method is based on goodness of fit and model parameters complexity. Defining deviance (goodness of fit) as:

$$D(\theta) = -2log(L(data|\theta)) + C$$

here $L(data|\theta)$ is likelihood function for data and model parameters θ. Constant term is ignored during the DIC comparison since they cancel out. When we estimate expected values for model parameters, we have expected deviance as:

$$D(\hat{\theta}) = D[E(\theta)]$$

From that, we can calculate the DIC as:

$$DIC = D(\hat{\theta}) + 2 * p_D$$

where p_D is the indicator for complexity of model, which is calculated as:

$$p_D = E_\theta[D(\theta)] - D(E[\theta])$$

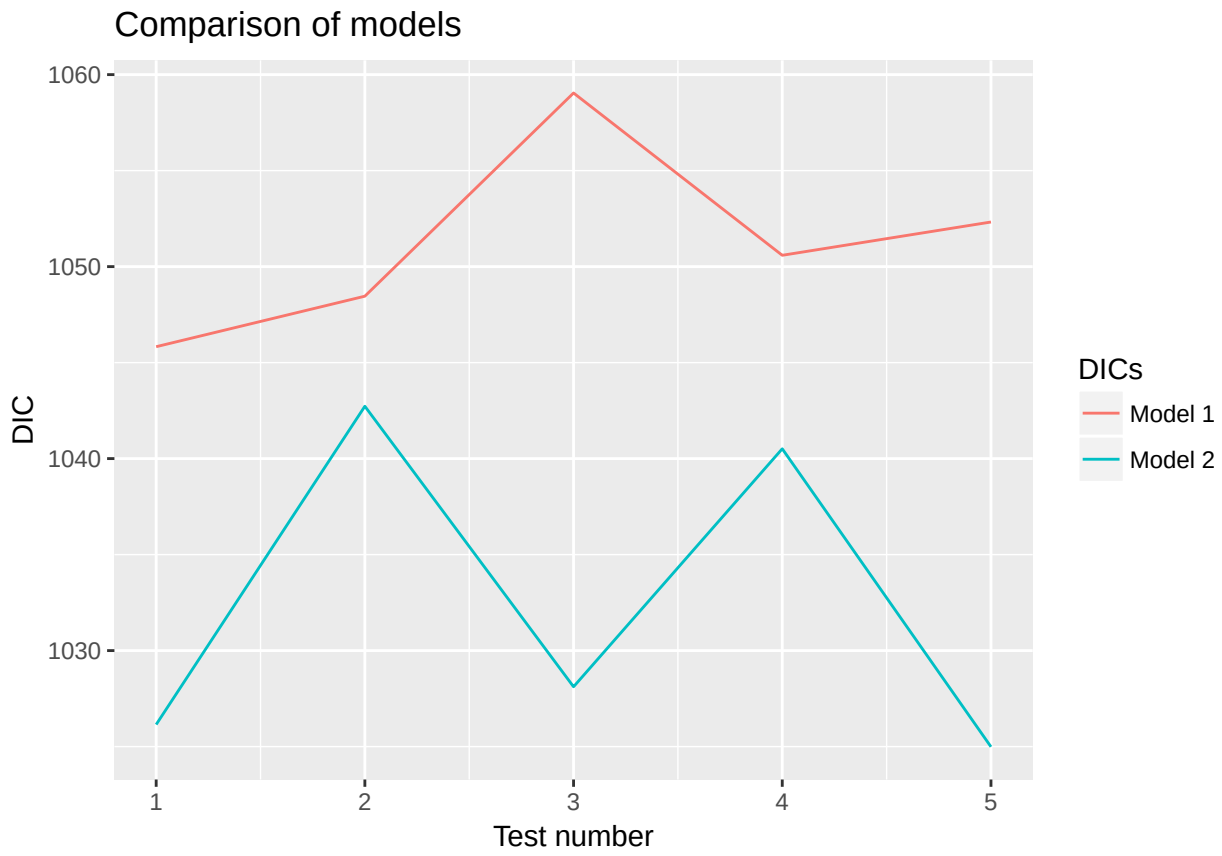
So, p_D is the penalization term for deviance. Indeed, a model fits easier when it has large number of parameters causing overfitting. Therefore, a model should have effective number of parameters. Finally, by comparing DICs of two models, we can say that a model, which has smaller DIC, is better.

We are going to check DIC of two models 5 times, in order to be sure about a good performing model.

```
test_number=5
dic1=rep(NA, test_number)
dic2=rep(NA, test_number)

initial1_1=list(alpha0 = 3, beta0 = -2,  tau.alpha = 2, tau.beta = 2, tau = 2)
initial2_1=list(alpha0 = 4, beta0 = -3,  tau.alpha = 1, tau.beta = 1, tau = 1)
inits_1=list(initial1_1, initial2_1)
parameters_1=c("alpha0","beta0","tau.alpha","tau.beta","tau")


initial1_2=list(alpha0 = 3, beta0 = -2,  tau.alpha = 2, tau.beta = 2, tau = 2, gamma=2)
initial2_2=list(alpha0 = 4, beta0 = -3,  tau.alpha = 1, tau.beta = 1, tau = 1, gamma=1)
inits_2=list(initial1_2, initial2_2)
parameters_2=c("alpha0","beta0","tau.alpha","tau.beta","tau", "gamma")

for (n in 1:test_number){

  model1=jags(data=data, inits=inits_1,
              parameters.to.save=parameters_1,
              model.file="model1.txt",
              n.chains=2,
              n.burnin=1000,
              n.iter=11000)
  model2=jags(data=data, inits=inits_2,
              parameters.to.save=parameters_2,
              model.file="model2.txt",
              n.chains=2,
              n.burnin=1000,
```

```
              n.iter=11000)

  output=head(model1)
  output_2=head(model2)

  dic1[n] = output$BUGSoutput$DIC
  dic2[n]=output_2$BUGSoutput$DIC
}
```

```
model1=data.frame(dic1)
model2=data.frame(dic2)
model1$n = 'Model 1'
model2$n = 'Model 2'
colnames(model1)='DIC'
colnames(model2)='DIC'
tests <- rbind(model1, model2)
colnames(tests)[2]='DICs'
tests$test_n=c(1,2,3,4,5,1,2,3,4,5)
ggplot(tests, aes(x=test_n, y=DIC, colour = DICs)) + ggtitle("Comparison of models") +geom_line()+ xlab
```



From the plot we can see that model 2 shows better result than model 1 for all 5 cases. Therefore, we choose model 2.

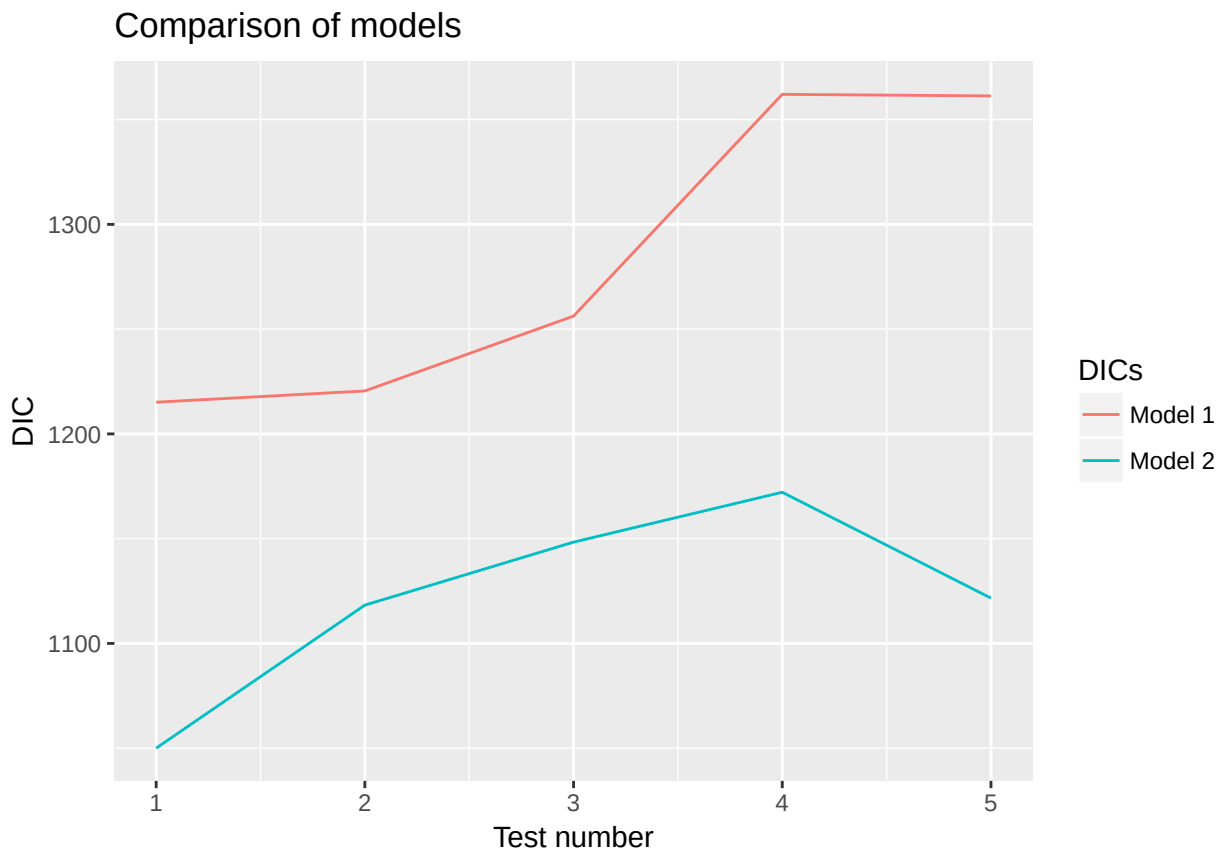
Ability of fully Bayesian analysis with simulated data

We check ability of fully bayesian analysis to recover model parameters from simulated data from chosen model. For a chosen we selected model 2 from previous part. After that, data will be generated at each iteration, and that data is used for simulation of two models. At each iteration we save corresponding DIC values of two models, to compare it later.

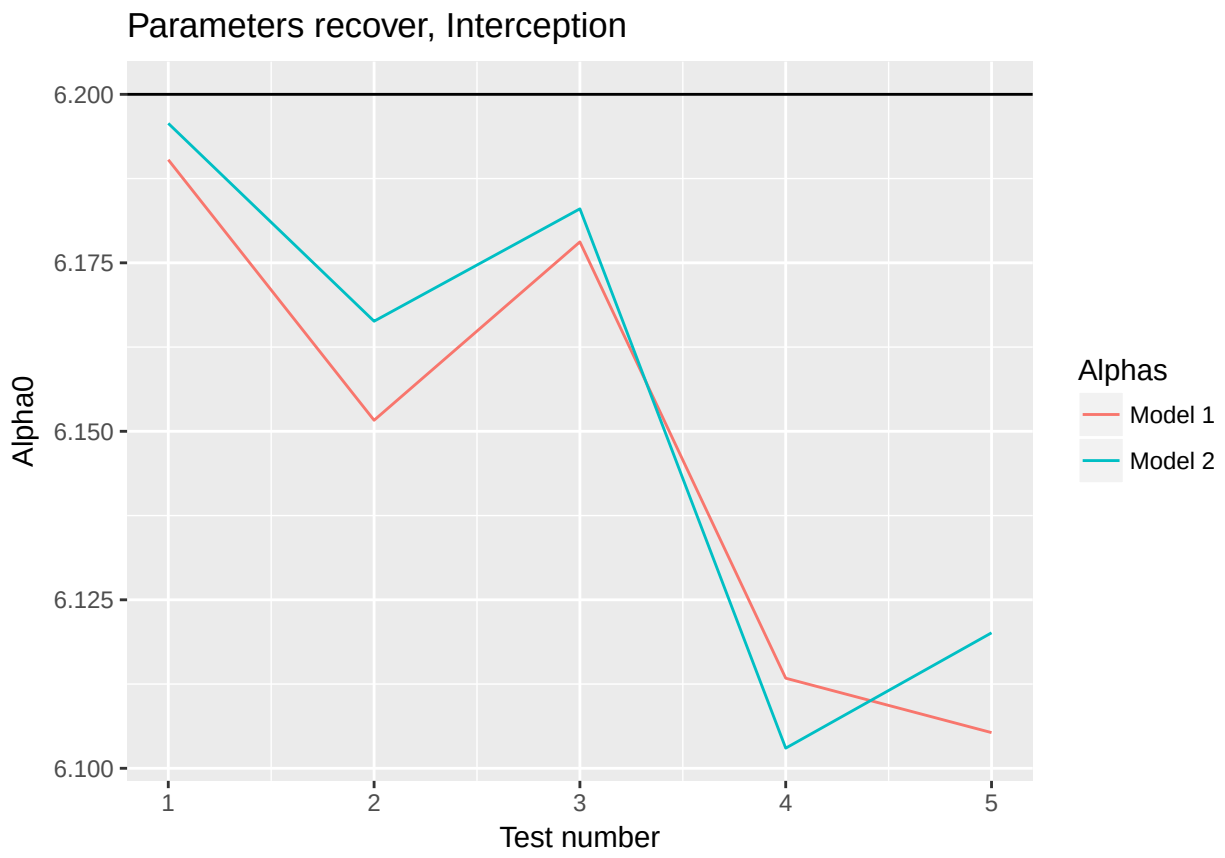
```
iter=5
alpha=6.2
beta=-1.1
gamma=0.7
tau=1
sigma=1/(sqrt(tau))
sdd=mean(tvec1)
y0_mean=mean(y0)

#we need create y0s list:

y02=append(y0, y0)
y03=append(y02, y0)
y03_mat=matrix(y03, byrow=TRUE, nrow=3 )
y03_list=as.vector(y03_mat)

dic1=rep(NA, iter)
dic2=rep(NA, iter)
alpha0_mod_1=rep(NA, iter)
alpha0_mod_2=rep(NA, iter)
beta0_mod_1=rep(NA, iter)
beta0_mod_2=rep(NA, iter)

initial1_1=list(alpha0 = 3, beta0 = -2,  tau.alpha = 2, tau.beta = 2, tau = 2)
initial2_1=list(alpha0 = 4, beta0 = -3,  tau.alpha = 1, tau.beta = 1, tau = 1)
inits_1=list(initial1_1, initial2_1)
parameters_1=c("alpha0","beta0","tau.alpha","tau.beta","tau")

initial1_2=list(alpha0 = 3, beta0 = -2,  tau.alpha = 2, tau.beta = 2, tau = 2, gamma=2)
initial2_2=list(alpha0 = 4, beta0 = -3,  tau.alpha = 1, tau.beta = 1, tau = 1, gamma=1)
inits_2=list(initial1_2, initial2_2)
parameters_2=c("alpha0","beta0","tau.alpha","tau.beta","tau", "gamma")

for (n in 1:iter){
  new_HB=c()
  for (i in 1:length(t_log)){
    mu=alpha+gamma*(y03_list[i]-y0_mean)+beta*(t_log[i]-sdd)
    y=rnorm(1,mean=rnorm(1, mean=mu,sd=sigma))
    new_HB=append(new_HB,y)
  }
  hb_matrix_new=matrix(new_HB, byrow=TRUE, nrow=106 )
  time_matrix_new=matrix(t_log, byrow=TRUE, nrow=106)
  new_data=list( N = 106,
                 Y =hb_matrix_new,
                 t = time_matrix,
                 y0 = y0,
                 times=nn)     #data created
```

```
  model1=jags(data=new_data, inits=inits_1,
              parameters.to.save=parameters_1,
              model.file="model1.txt",
              n.chains=2,
              n.burnin=1000,
              n.iter=11000)
  model2=jags(data=new_data, inits=inits_2,
              parameters.to.save=parameters_2,
              model.file="model2.txt",
              n.chains=2,
              n.burnin=1000,
              n.iter=11000)

  output=head(model1)
  output_2=head(model2)

  dic1[n] = output$BUGSoutput$DIC
  dic2[n]=output_2$BUGSoutput$DIC
  alpha0_mod_1[n]=output$BUGSoutput$mean$alpha0
  alpha0_mod_2[n]=output_2$BUGSoutput$mean$alpha0
  beta0_mod_1[n]=output$BUGSoutput$mean$beta0
  beta0_mod_2[n]=output_2$BUGSoutput$mean$beta0


}
```

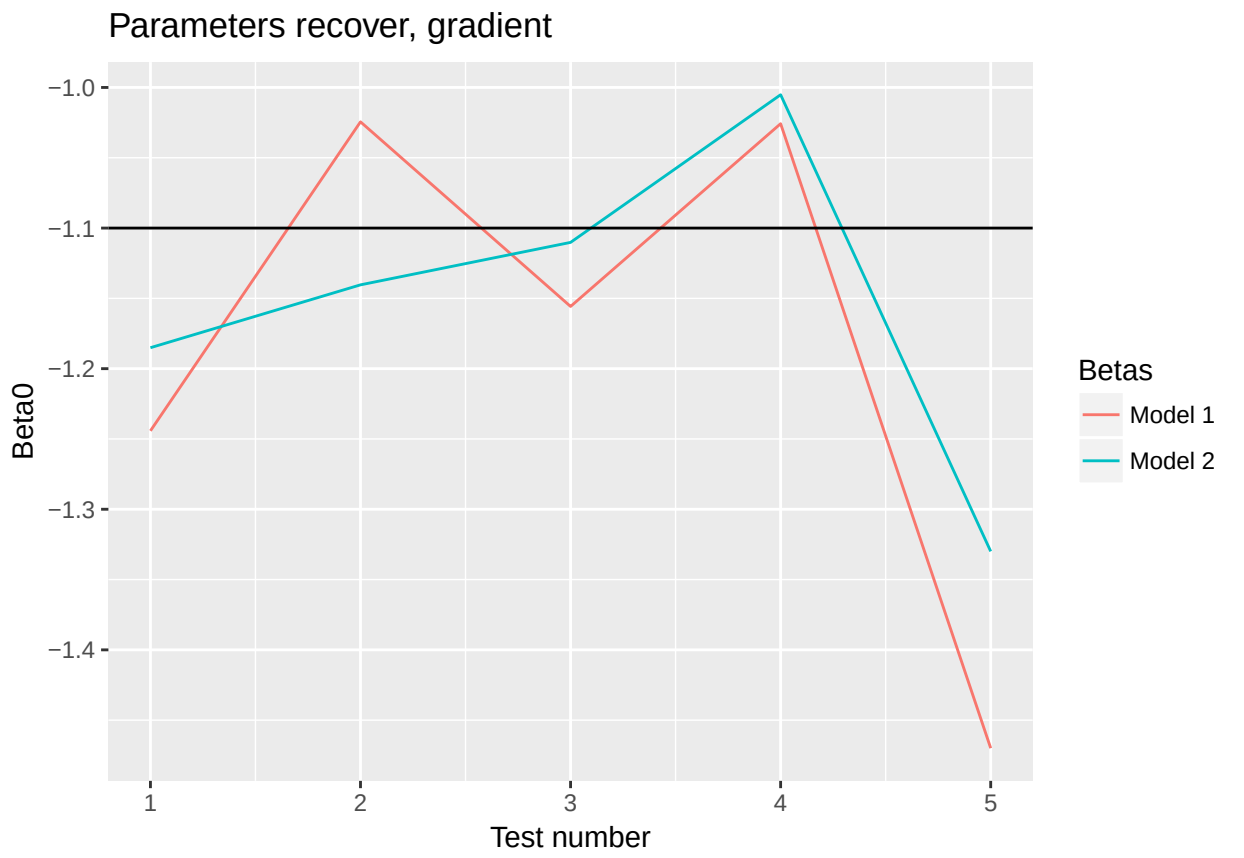
```
n=seq(1,iter)
n=append(n,n)

model1_dic=data.frame(dic1)
model2_dic=data.frame(dic2)
model1_dic$n = 'Model 1'
model2_dic$n = 'Model 2'
colnames(model1_dic)='DIC'
colnames(model2_dic)='DIC'
tests <- rbind(model1_dic, model2_dic)
colnames(tests)[2]='DICs'
tests$test_n=n
ggplot(tests, aes(x=test_n, y=DIC, colour = DICs)) + ggtitle("Comparison of models") +geom_line()+ xlab
```




```
model1_alpha=data.frame(alpha0_mod_1)
model2_alpha=data.frame(alpha0_mod_2)
model1_alpha$n = 'Model 1'
model2_alpha$n = 'Model 2'
colnames(model1_alpha)='Alpha0'
colnames(model2_alpha)='Alpha0'
tests <- rbind(model1_alpha, model2_alpha)
colnames(tests)[2]='Alphas'
tests$test_n=n
ggplot(tests, aes(x=test_n, y=Alpha0, colour = Alphas)) + ggtitle("Parameters recover, Interception") +g
```




```
model1_beta=data.frame(beta0_mod_1)
model2_beta=data.frame(beta0_mod_2)
model1_beta$n = 'Model 1'
model2_beta$n = 'Model 2'
colnames(model1_beta)='Beta0'
colnames(model2_beta)='Beta0'
tests <- rbind(model1_beta, model2_beta)
colnames(tests)[2]='Betas'
tests$test_n=n
ggplot(tests, aes(x=test_n, y=Beta0, colour = Betas)) + ggtitle("Parameters recover, gradient") +geom_li
```

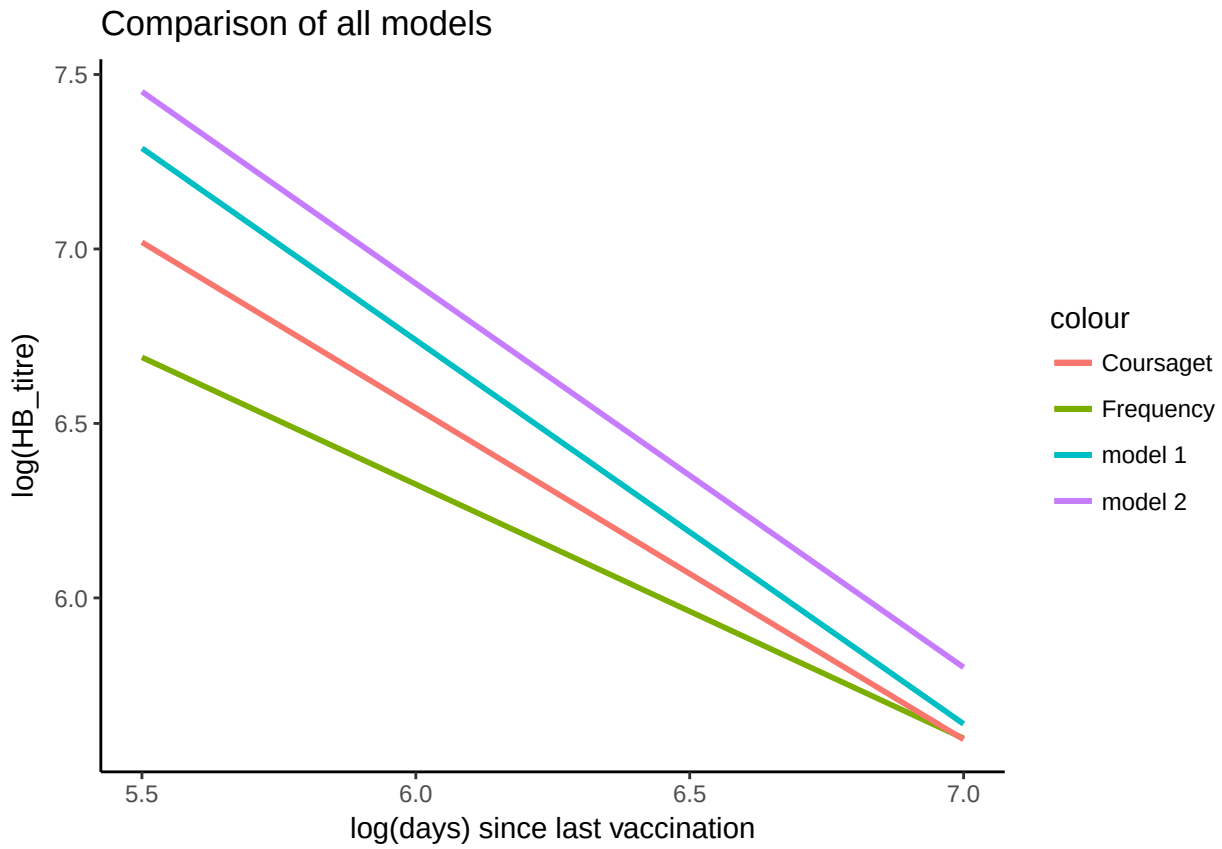


Final evaluation

Here, we evaluate models with finding of Coursaget et al. graphically. According to the study, model parameters are $\alpha_{mean} = 5.94$, $\beta_{mean} = -0.95$ and $\gamma_{mean} = 0.6$. Now, considering the range for $log(time) \in [5.5, 7]$, let's plot lines based on mean values:

```
#let's take a base log titre value for our model 2 and Coursaget model, computing mean
#at time log(5.5) for model 1 and frequency analysis:
tm=mean(tvec1)
y1=mean_alpha+mean_beta*(5.5-tm)
y2=6.2-1.1*(5.5-tm)
y0_test=(y1+y2)/2


plt <- ggplot(NULL) + aes(x=c(5.5,7)) +
  theme_classic() +
  labs(x='log(days) since last vaccination',y='log(HB_titre)', title='Comparison of all models')
regression = function(x, alpha, beta) alpha+beta*(x-tm)


plt <- plt + stat_function(fun=regression,
                           args=list(alpha=mean_alpha,beta=mean_beta),
                           size=1, alpha=1,aes(colour = "Frequency")) #from freq analysis
plt <- plt + stat_function(fun=regression,
                           args=list(alpha=6.2,
```

```
                                    beta=-1.1),
                            size=1, alpha=1,aes(color='model 1'))   #from model 1
plt <- plt + stat_function(fun=regression,
                            args=list(alpha=6.2+0.7*(y0_test-mean(y0)),
                                    beta=-1.1),
                            size=1, alpha=1,aes(color='model 2'))  #from model 2
plt <- plt + stat_function(fun=regression,
                            args=list(alpha=5.94+0.6*(y0_test-mean(y0)),
                                    beta=-0.95),
                            size=1, alpha=1,aes(color='Coursaget'))  #from model 2
plt
```



Reference

1. Coursaget et al. (1991). "Scheduling of revaccination against hepatitis B". The Lancet vol 337
2. Spiegelhalter et al.(1996). "Markov Chain Monte Carlo in Practice". Chapman and Hall
3. Tardella L. (2018). "Course: Statistical Methods for Data Science II, lecture notes".