Stochastic Processes and Time Series: Project II

Andrea Silvi
Politecnico di Torino
andrea.silvi@studenti.polito.it

Ulysse Marquis
Politecnico di Torino
s293793@studenti.polito.it

Abstract

In this report we implement a Gibbs sampler including a Metropolis Hastings sampler to estimate the posterior distributions of the parameters for a model estimating rat's weights over 14 weeks during an experiment. We then evaluate our model visually and compare it to a smaller model that we come up with.

1. Introductory Analysis

The goal of this report is to analyze a dataset consisting of the evolution of rats weights over time. We assume that data are noisy measurements of a Gompertz curve

$$w_i(t_j) = A_i \exp\left(-\exp\left(-b_i(t_j - c_i)\right)\right) + \epsilon_{i,j}, \tag{1}$$

where $\epsilon_{i,j} \sim \mathcal{N}(0, \frac{1}{\tau})$. We take the logarithms of each observation, that are then assumed to be distributed as $\log Y_i(t_j) \sim \mathcal{N}(\log w_i(t_j), \frac{1}{\tau})$.

We report the hierarchical model scheme in Figure 1. We assume the parameters of the Gompertz curve to be distributed conditionally normally with respect to population parameters $log(a), log(b), log(c), \alpha, \beta, \gamma$. We assume these to be distributed either as $\mathcal{N}(0, 100)$ for averages either as $\Gamma(0.1, 0.1)$ for precisions. We also assume the overall precision τ prior to be distributed as $\Gamma(0.1, 0.1)$. More specifically,

$$\begin{aligned}
\log A_i | a, \alpha &\sim \mathcal{N}(\log a, 1/\alpha) \\
\log b_i | b, \beta &\sim \mathcal{N}(\log b, 1/\beta) \\
\log c_i | c, \gamma &\sim \mathcal{N}(\log c, 1/\gamma)
\end{aligned}$$

In order to infer these parameters, we leverage a Gibbs Sampler algorithm using Metropolis Hastings to sample from the full conditional distributions. We use as proposal a random walk defined as $x_j^{(i)} = x_j^{(i-1)} + \omega_j$, where $\omega_j \sim \mathcal{N}(0, \sigma_j)$ and σ_j is specifically tuned for each parameter.

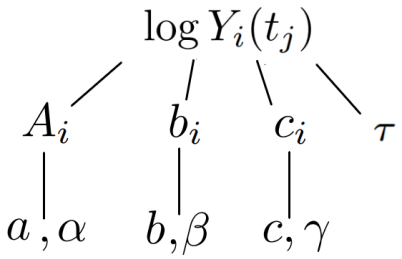

Figure 1: Hierarchical model.

After sampling at time step i the new candidate x_j for parameter j, Metropolis Hastings accepts it with a probability

$$\alpha(x_j^{(i-1)}, x_j^{(i)}) = \min\{1, \frac{\pi(x_j^{(i)}, x_{-j})}{\pi(x_j^{(i-1)}, x_{-j})}\}, \tag{2}$$

where $\pi(x_j^{(i)}, x_{-j})$ is the posterior of our model that at the numerator receives the new candidate $x_j^{(i)}$, while at the denominator the one from the previous iteration $x_j^{(i-1)}$, while x_{-j} in both represent all the other parameters except x_j that are kept the same, and the proposal does not appear since it is symmetric.

In order to compute the posterior $\pi(x)$, we also define the log-likelihood as

$$\log \mathcal{L}(\{\log Y_i(t_j)\}_{(i=1\ldots n, j=1\ldots 14)} | \{A_i, b_i, c_i\}_{i=1}^n, \tau) \propto -\frac{\tau}{2} \sum_{i,j} (\log Y_i(t_j) - \log w_i(t_j))^2 \tag{3}$$

2. Results

We run the MCMC algorithm with a burn-in period of 200 iterations and a total of 1000 iterations. In order to decrease autocorrelation, we also use a thinning factor of 10. We obtain acceptance rates for all 481 parameters in the interval $[0.1973, 0.3494]$, with the mean of the acceptance rates being 0.29 ± 0.02.

We report the chains of the parameters of the first twelve rats in Figure 2, 3 and 4. We also report the chains of the population parameters $(a, \alpha, b, \beta, c, \gamma)$ and the overall precision τ in Figure 5. As we can see, while the chains of the first two sets of parameters converge fairly quickly and are not noisy, the chains of the third set of parameters and their population mean c and precision γ are fairly noisy and do not really seem to converge. Actually, since the average of the logarithm of these parameters lies around -10, this indicates us that they are insignificant, since they are only compared to t in (1), which takes values in the range $[1, 14]$, thus not contributing to the overall weight.

We also use the obtained parameters samples to estimate the evolution of the weight of the first nine rats in the first 14 weeks, and we compare it with the actual measurements. We report these results in Figure 6. Visually, the estimated curves seem to follow the data quite accurately.

Since we find the parameters c_i not to be influent, we also try to fit a new model, similar to the one presented in Figure 1 but without the c_i's. As a consequence, the data are now considered to be noisy measurements of the curve

$$w_i(t_j) = A_i \exp\left(-\exp\left(-b_i(t_j)\right)\right) + \epsilon_{i,j}. \tag{4}$$

We present the estimated evolution of the weights of the first nine rats with the parameters obtained from this second model in Figure 7. Again, visually the predicted curve fits the data nicely.

In order to compare the two models, we use the Deviance Information Criterion (DIC). It is defined as

$$DIC = -2 \log \mathcal{L}(Y|\bar{\theta}) + 2k_{eff}, \tag{5}$$

where k_{eff} is the effective number of parameters, defined as

$$k_{eff} = 2(\log \mathcal{L}(Y|\bar{\theta}) - \mathbb{E}_\pi[\log \mathcal{L}(Y|\theta)]). \tag{6}$$

For the first model, we obtain that $DIC = -7607$, while for the second model we obtain that $DIC = -7417$. This suggests us that even though the difference is not huge, still the model with the higher number of parameters seems to outperform the smaller one, even though from our simulations we assumed that the third set of parameters c_i did not have an influence in the overall weight.

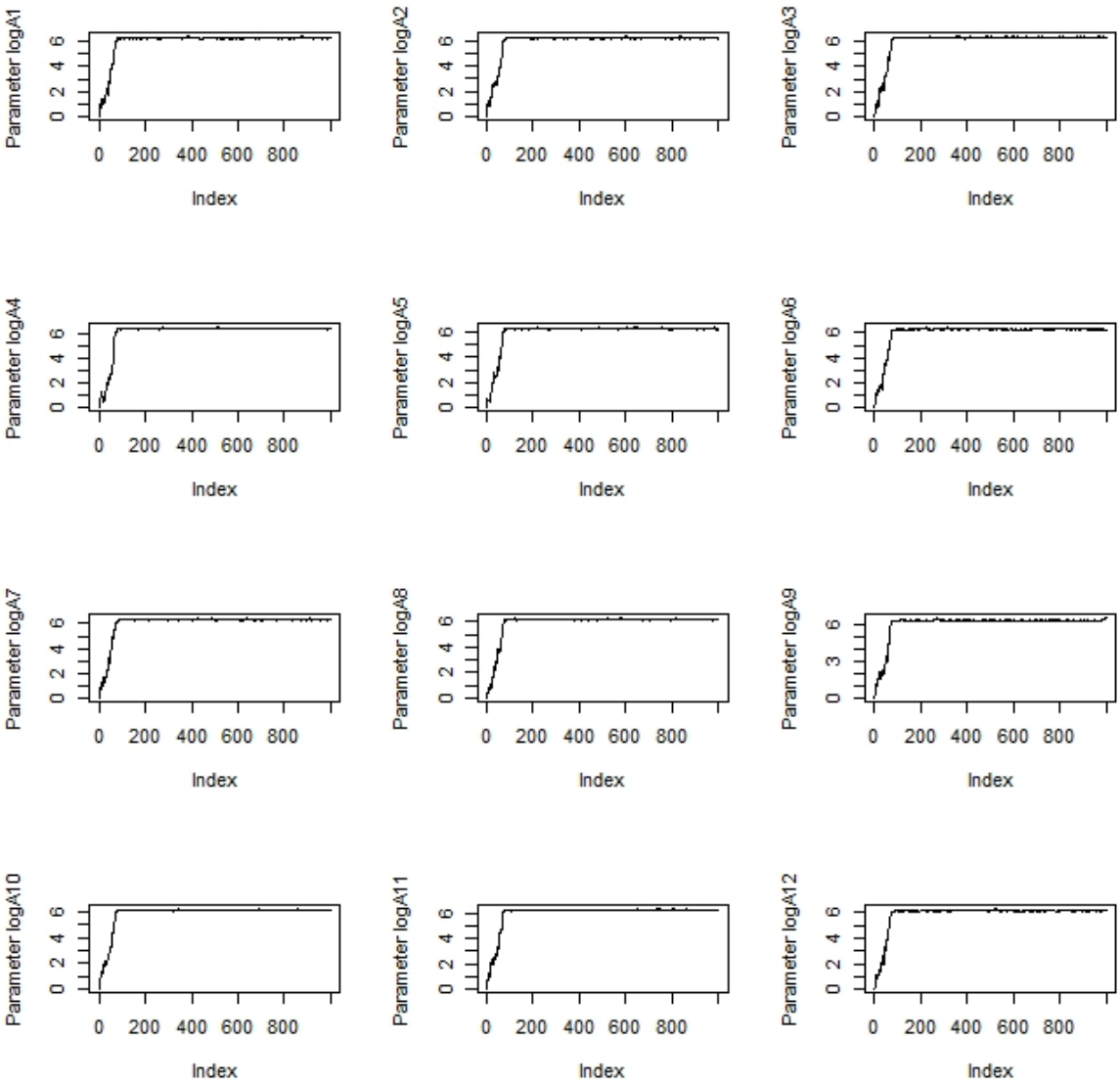

Figure 2: Chains of the first 12 $\log A_i$ parameters.

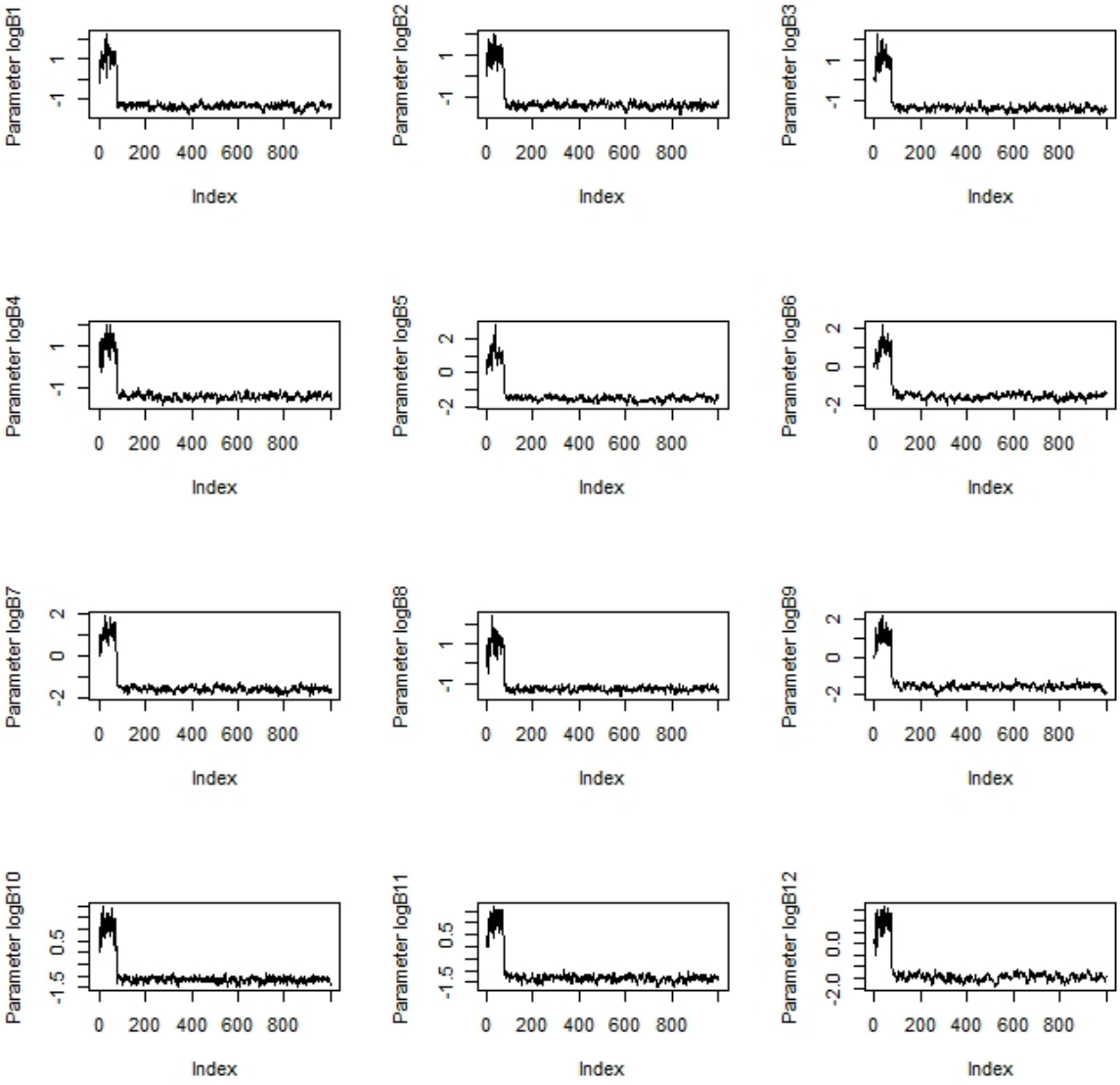

Figure 3: Chains of the first 12 $\log b_i$ parameters.

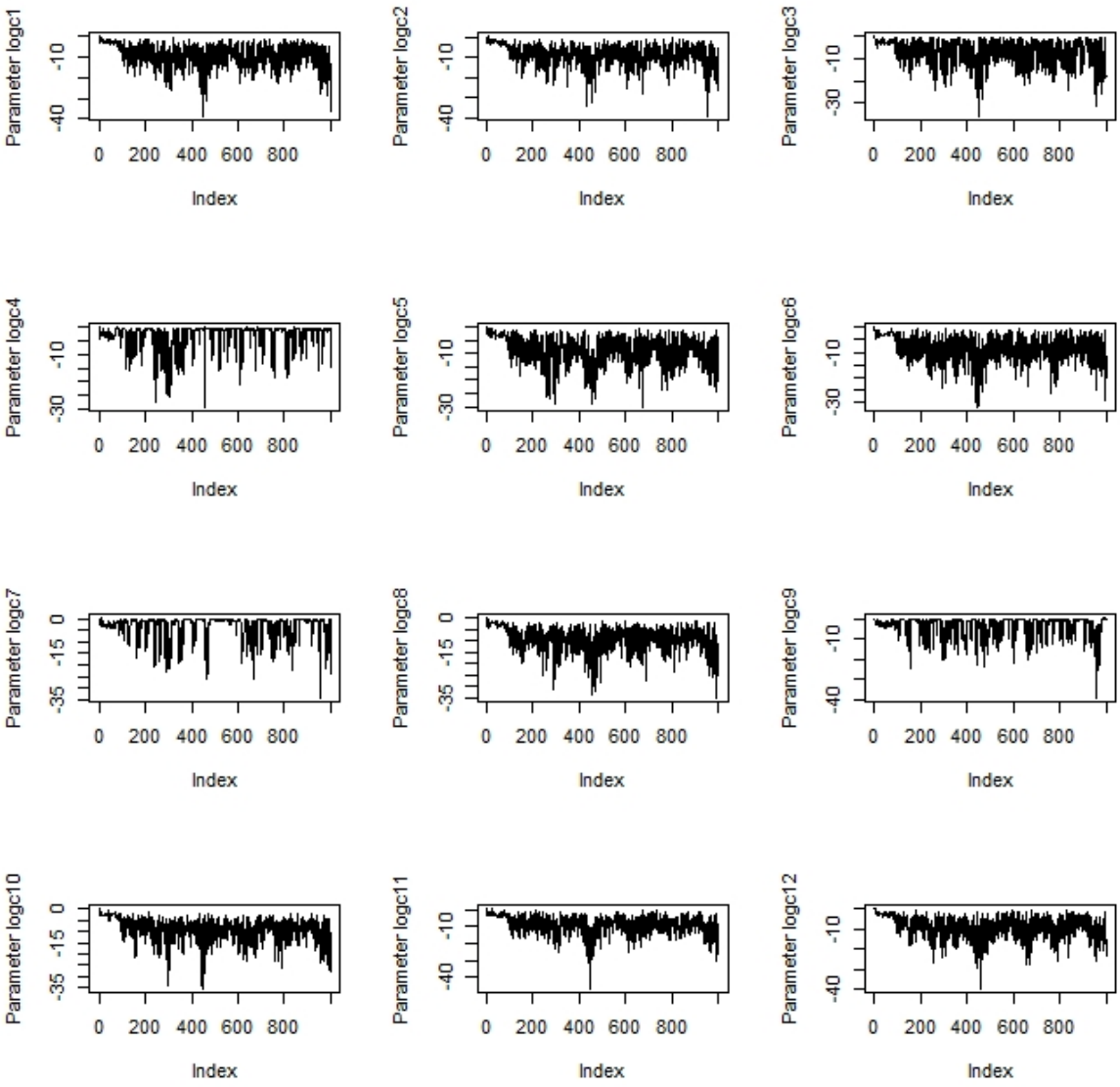

Figure 4: Chains of the first 12 $\log c_i$ parameters.

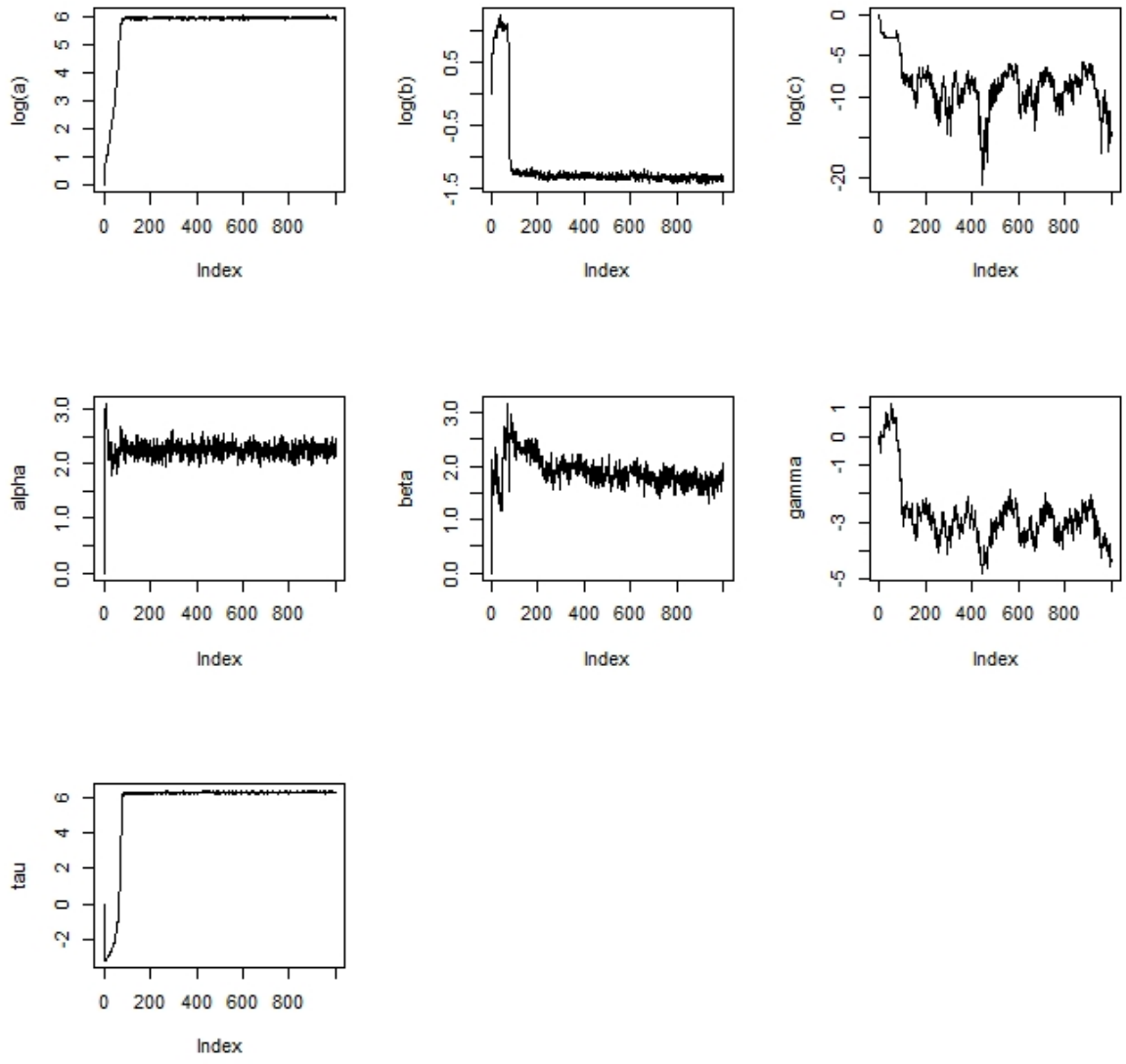

Figure 5: Chains of the populations parameters.

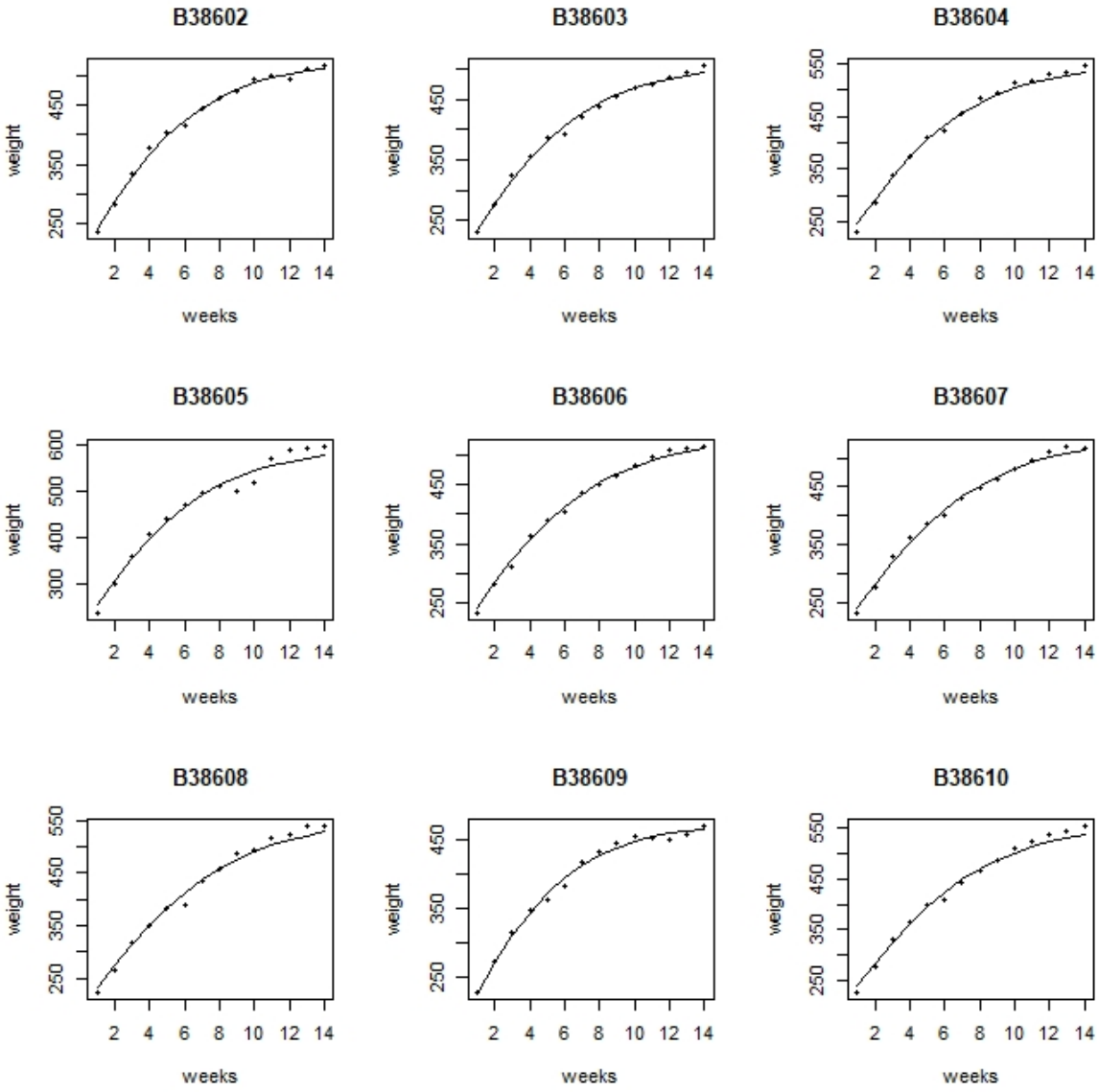

Figure 6: Comparison between the actual weights measurements and the curves obtained from the parameters found for the first 9 rats.

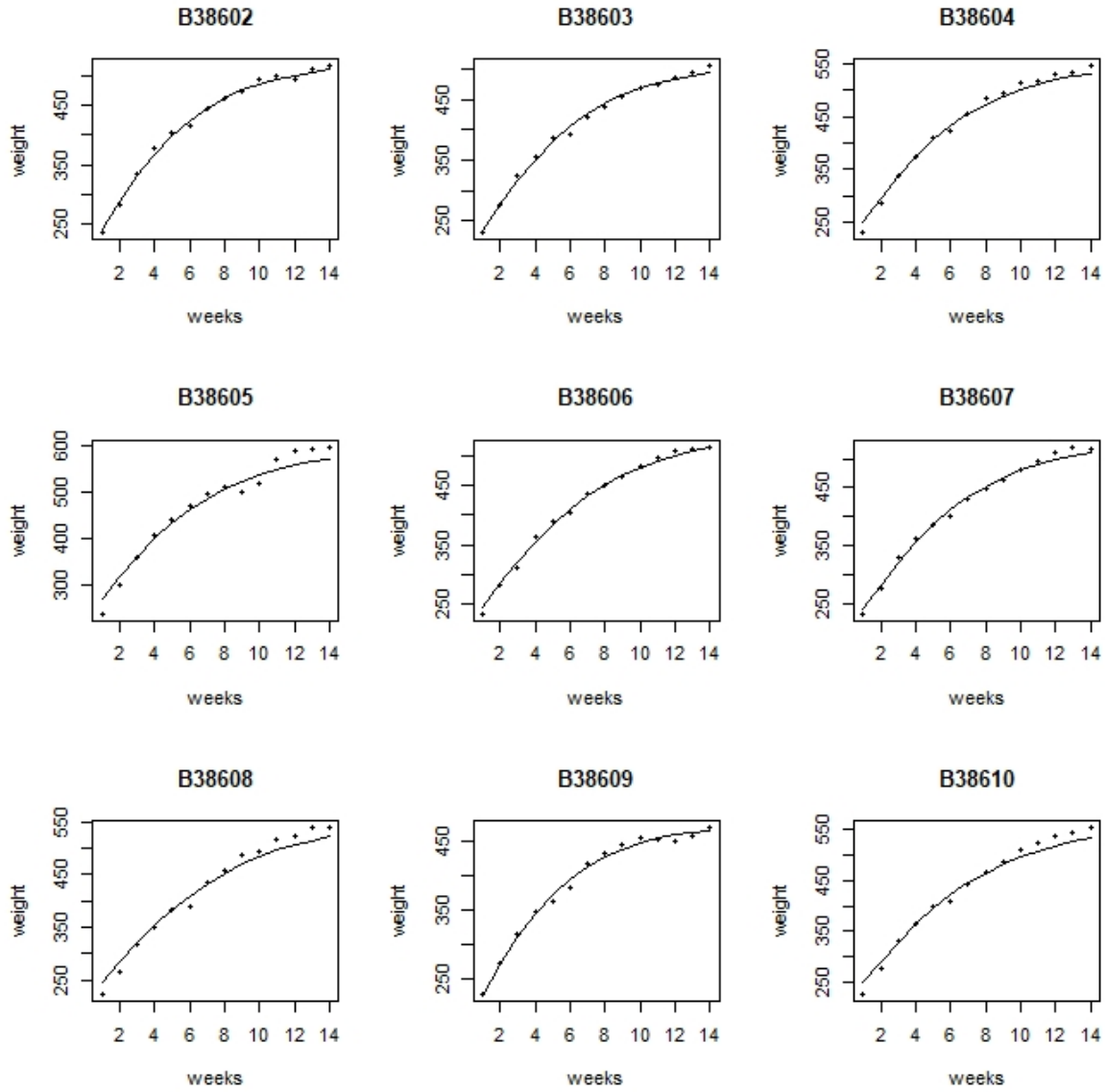

Figure 7: Comparison between the actual weights measurements and the curves obtained from the parameters found in the smaller model for the first 9 rats.

```
######### Code implementation


### Reading the data

data <- read.csv("ratWeight.csv")
attach(data)
n_params <- 158 * 3 + 3 * 2 + 1
n_rats <- 158
Y<-matrix(weight, nrow=n_rats, byrow=TRUE)
lY<-log(Y)
t<-matrix(week, nrow=n_rats, byrow=TRUE)
rats<-matrix(id ,nrow=n_rats, byrow=TRUE)
post.samp.size<-1000
thin<-10

### Tuning of the sd's

sd.proposal<-rep(1, n_params)
sd.proposal[1:158]<-0.065

sd.proposal[159:316]<-0.19

sd.proposal[168]<-0.2
sd.proposal[180]<-0.202
sd.proposal[195:196]<-0.205
sd.proposal[201]<-0.2
sd.proposal[210]<-0.2
sd.proposal[238]<-0.265
sd.proposal[239:240]<-0.215
sd.proposal[241]<-0.275
sd.proposal[244]<-0.25
sd.proposal[246:250]<-0.27
sd.proposal[249]<-0.29
sd.proposal[252]<-0.275
sd.proposal[253:254]<-0.25
sd.proposal[255:256]<-0.205
sd.proposal[258:262]<-0.25
sd.proposal[264]<-0.22
sd.proposal[265:268]<-0.25
sd.proposal[269]<-0.21
sd.proposal[270]<-0.23
sd.proposal[271]<-0.21
sd.proposal[272]<-0.225
sd.proposal[273:276]<-0.215
sd.proposal[277]<-0.265
sd.proposal[278]<-0.21
sd.proposal[279:284]<-0.25
sd.proposal[281]<-0.21
sd.proposal[285]<-0.21
sd.proposal[286]<-0.26
sd.proposal[287]<-0.262
```

```
sd.proposal[289:290]<-0.23
sd.proposal[292:296]<-0.24
sd.proposal[298]<-0.25
sd.proposal[299]<-0.2
sd.proposal[300]<-0.25
sd.proposal[303]<-0.25
sd.proposal[305]<-0.22
sd.proposal[306:307]<-0.24
sd.proposal[309]<-0.225
sd.proposal[310]<-0.205
sd.proposal[312:315]<-0.25

sd.proposal[317:474]<-15
sd.proposal[319]<-12.5
sd.proposal[320]<-4.85
sd.proposal[323]<-5.1
sd.proposal[325]<-6.45
sd.proposal[337]<-2
sd.proposal[340]<-0.68
sd.proposal[342]<-9.65
sd.proposal[344]<-7.25
sd.proposal[352]<-7.3
sd.proposal[360]<-3.75
sd.proposal[364]<-1.32
sd.proposal[365]<-1.18
sd.proposal[372]<-0.9
sd.proposal[374]<-4.9
sd.proposal[380]<-1.25
sd.proposal[475]<-0.1
sd.proposal[476]<-0.12
sd.proposal[477]<-1.4
sd.proposal[478:480]<-0.5
sd.proposal[n_params]<-0.12

### Utils

chain<-matrix(rep(0,num_params*post.samp.size), byrow= TRUE, nrow=post.samp.size)

lw <- function(t, la, lb, lc) la - exp(-exp(lb)*(t-exp(lc))) # logarithm of weight
likelihood <- function(lY, t, lAi, lbi, lci, ltau)
        sum(dnorm(lY, mean=lw(t, lAi, lbi, lci),sd= sqrt(1/exp(ltau)),log=TRUE))
condPriorAi <- function(lAi,la,logalpha)
        sum(dnorm(lAi,mean=la,sd=sqrt(1/exp(logalpha)),log=TRUE))
condPriorbi <- function(lbi,lb,logbeta)
        sum(dnorm(lbi,mean=lb,sd=sqrt(1/exp(logbeta)),log=TRUE))
condPriorci <- function(lci,lc,loggamma)
        sum(dnorm(lci,mean=lc,sd=sqrt(1/exp(loggamma)),log=TRUE))
priorla<-function(la) dnorm(la,mean=0,sd=sqrt(100),log=TRUE)
priorlb<-function(lb) dnorm(lb,mean=0,sd=sqrt(100),log=TRUE)
priorlc<-function(lc) dnorm(lc,mean=0,sd=sqrt(100),log=TRUE)
#this is the log of the prior of the log precisions, lalpha, lbeta, lgamma,ltau
logpriorloggamma<-function(precision) precision*0.1-0.1*exp(precision)
```

```
posterior<-function(parameters){
  lAi<-matrix(parameters[1:158], byrow=FALSE, ncol=14, nrow=n_rats)
  lbi<-matrix(parameters[159:316], byrow=FALSE, ncol=14, nrow=n_rats)
  lci<-matrix(parameters[317:474], byrow=FALSE, ncol=14, nrow=n_rats)
  lk<-likelihood(lY, t, lAi,lbi,lci, parameters[n_params])
  cAi<-condPriorAi(parameters[1:158], parameters[158 * 3 + 1], parameters[158 * 3 + 4])
  cbi <-condPriorbi(parameters[159:316], parameters[158 * 3 + 2], parameters[158 * 3 + 5])
  cci<-condPriorci(parameters[317:474], parameters[158 * 3 + 3], parameters[158 * 3 + 6])
  return
   (lk+cAi+cbi+cci+priorla(parameters[158 * 3 + 1])+priorlb(parameters[158 * 3 + 2])
   +priorlc(parameters[158 * 3 + 3])+logpriorloggamma(parameters[158 * 3 + 4])
   +logpriorloggamma(parameters[158 * 3 + 5])+logpriorloggamma(parameters[158 * 3 + 6])
   +logpriorloggamma(parameters[n_params]))
}

#####################################

print("Main loop")

acceptances<-rep(0, n_params)

old<-new<-chain[1,]
for (iter in 2:post.samp.size){
  print(iter)
  for (inner in 1:thin){
    old<-new
    for (i in 1: n_params){
      updated.pars<-new
      candidate<-old[i]+rnorm(1, mean=0, sd=sd.proposal[i])
      updated.pars[i]<-candidate
      log.acc.ratio<- posterior(updated.pars)-posterior(new)
      if (log(runif(1))<log.acc.ratio){new[i]<-candidate
      acceptances[i]<-acceptances[i]+1
      } else new[i]<-old[i]
    }
    chain[iter,]<-new
  }    }
final_chain<-chain
acceptances/(post.samp.size*thin)
post.means<-apply(chain,2,mean)


### Chains plots

dev.new()
par(mfrow=c(4, 3))
for (i in 1:12){
  plot(chain[,i],type="l",ylab=sprintf("Parameter logA%d", i))
}
dev.new()
par(mfrow=c(4, 3))
for (i in 159:(159+11)){
```

```
  plot(chain[,i],type="l",ylab=sprintf("Parameter logb%d", i-158))
}
dev.new()
par(mfrow=c(4,3))
for (i in (2*158+1):(2*158+12)){
  plot(chain[,i],type="l",ylab=sprintf("Parameter logc%d", i-2*158))
}
dev.new()
par(mfrow=c(3,3))
for (i in (158 * 3 + 1): n_params){
  plot(chain[,i],type="l",ylab="Population parameters")
}

dev.new()
par(mfrow=c(3,3))
plot(chain[,158*3+1],type="l",ylab="log(a)")
plot(chain[,158*3+2],type="l",ylab="log(b)")
plot(chain[,158*3+3],type="l",ylab="log(c)")
plot(chain[,158*3+4],type="l",ylab="alpha")
plot(chain[,158*3+5],type="l",ylab="beta")
plot(chain[,158*3+6],type="l",ylab="gamma")
plot(chain[,158*3+7],type="l",ylab="tau")


### Comparison between obtained curves and actual measurements

sequence<-201:post.samp.size
llss<-length(sequence)

trajectories<-array(0,dim=c(n_rats,14,llss))
for (iter in 1:llss) {
  lA<-matrix(chain[sequence[iter],1:158],byrow=FALSE,ncol=14,nrow=n_rats)
  lb<-matrix(chain[sequence[iter],159:316],byrow=FALSE,ncol=14,nrow=n_rats)
  lc<-matrix(chain[sequence[iter],317:474],byrow=FALSE,ncol=14,nrow=n_rats)
  trajectories[,,iter]<-exp(lw(t, lA, lb, lc))
}
medians<-apply(trajectories,c(1,2),median)

dev.new()
par(mfrow=c(3,3))
for (rat in 1:9){
  plot(x=t[rat,], y=Y[rat,], xlab="weeks", ylab="weight", main=id[1+14*(rat-1)], pch=20)
  lines(x=t[rat,], y=medians[rat,])
}


### Calculating the DIC

lAi_bar<-matrix(apply(chain[201:1000, 1:158], 2, mean), byrow=FALSE, ncol=14, nrow=n_rats)
lbi_bar<-matrix(apply(chain[201:1000, 159:316], 2, mean), byrow=FALSE, ncol=14, nrow=n_rats)
lci_bar<-matrix(apply(chain[201:1000, 317:474], 2, mean), byrow=FALSE, ncol=14, nrow=n_rats)
ltau_bar<-mean(chain[201:1000, n_params])
lk_bar<-likelihood(lY, t, lAi_bar, lbi_bar, lci_bar, ltau_bar)
```

```
avg_lk<-0
for (iter in 1:llss){
  lA_i<-matrix(chain[sequence[iter],1:158],byrow=FALSE,ncol=14,nrow=n_rats)
  lb_i<-matrix(chain[sequence[iter],159:316],byrow=FALSE,ncol=14,nrow=n_rats)
  lc_i<-matrix(chain[sequence[iter],317:474],byrow=FALSE,ncol=14,nrow=n_rats)
  ltau<-chain[sequence[iter], n_params]
  avg_lk<-avg_lk + likelihood(lY, t, lA_i, lb_i, lc_i, ltau)
}
avg_lk<-avg_lk/llss
pd<- -2*(avg_lk)+2*lk_bar
DIC<- pd-2*(avg_lk)
```