Statistical Methods for Data Science Laboratory Computational Statistics (2015/2016)

Assignment # 3
Final Project

Paolo Tamagnini

July 2016

Contents

1 The data

This model, by Gelfand et al (1990), concerns 30 young rats whose weights were measured weekly for five weeks. Part of the data is shown below, where Y_{ij} is the weight of the i-th rat measured at age x_j.

	Weights Y_{ij} of rat i on day x_j				
	$x_j = 8$	15	22	29	36
Rat 1	151	199	246	283	320
Rat 2	145	199	249	293	354
.....					
Rat 30	153	200	244	286	324

$$i \in \{1; .. ; 30\}$$
$$j \in \{1; .. ; 5\}$$

2 Rats: a normal hierarchical model with linear expected value

$$Y_{ij} \sim N(\mu_{ij}, \tau_c)$$

$$\mu_{ij} = \alpha_i + \beta_i (x_j - \bar{x})$$
$$\bar{x} = \frac{1}{5} \sum_j x_j = 22$$

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

$$\alpha_c, \beta_c, \sim N(0, 10^{-4})$$
$$\tau_c, \tau_\alpha, \tau_\beta \sim G(0.001, 0.001)$$

The jags code for the model:

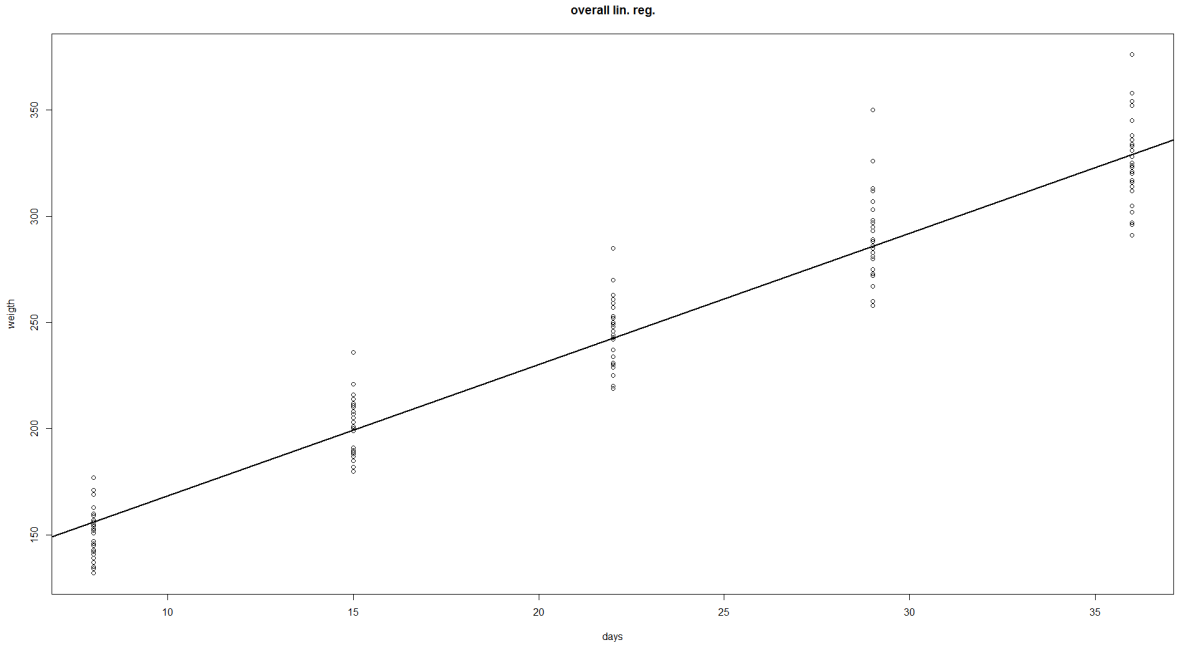
```
model {
    for (i in 1:N) {

      for (j in 1:T) {

        mu[i,j] <- alpha[i] + beta[i]*(x[j] - x.bar);
        Y[i,j]   ~ dnorm(mu[i,j],tau.c) }

      alpha[i] ~ dnorm(alpha.c,tau.alpha);
      beta[i]  ~ dnorm(beta.c,tau.beta); }

    alpha.c    ~ dnorm(0,1.0E-4);
    beta.c     ~ dnorm(0,1.0E-4);
    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);
    x.bar     <- mean(x[]); }
```

The R code:

```
nb=50000
nc= 2
ni = 100000
nsim=ni-nb

library(R2jags)
d <- read.jagsdata("rats-data.R")
initsFile <- read.jagsdata("rats-init.R")
inits=function(){
  initsFile }
attributes(initsFile)$names
params = c("tau.c","alpha","tau.beta","beta","tau.alpha","beta.c","alpha.c")
ratJags=jags(data=d, inits=inits, parameters.to.save=params, model.file="rats_paolo.txt",
             n.chains=nc, n.thin = 1, n.iter=ni, n.burnin =nb)

ratJags$model

alphaVect = ratJags$BUGSoutput$mean$alpha
betaVect = ratJags$BUGSoutput$mean$beta
alphaC = as.vector(ratJags$BUGSoutput$mean$alpha.c)
betaC = as.vector(ratJags$BUGSoutput$mean$beta.c)
xBar = mean(d$x)
```

Plotting now a scatter-plot with every pair (Y_{ij}, x_j) and using α_c and β_c to plot the overall linear regression using the line $\alpha_c + \beta_c (x - \bar{x})$.

```
X = t(replicate(30, d$x))
plot(X,d$Y, main='overall lin. reg.', ylab='weigth', xlab='days')
abline(a = alphaC-betaC*xBar, b=betaC, lw=2)
```



We can calculate the coefficient of determination for the regression of each rat.

```
R2all = calc_R2all_lin(d$Y, alphaVect, betaVect, d$x, xBar)

alphaChain1 = ratJags$BUGSoutput$sims.matrix[1:nsim,1:30]
```

```
alphaChain2 = ratJags$BUGSoutput$sims.matrix[(nsim+1):(nsim*2),1:30]
alphaChain = ratJags$BUGSoutput$sims.matrix[,1:30]

betaChain1 = ratJags$BUGSoutput$sims.matrix[1:nsim,32:61]
betaChain2 = ratJags$BUGSoutput$sims.matrix[(nsim+1):(nsim*2),32:61]
betaChain = ratJags$BUGSoutput$sims.matrix[,32:61]
```

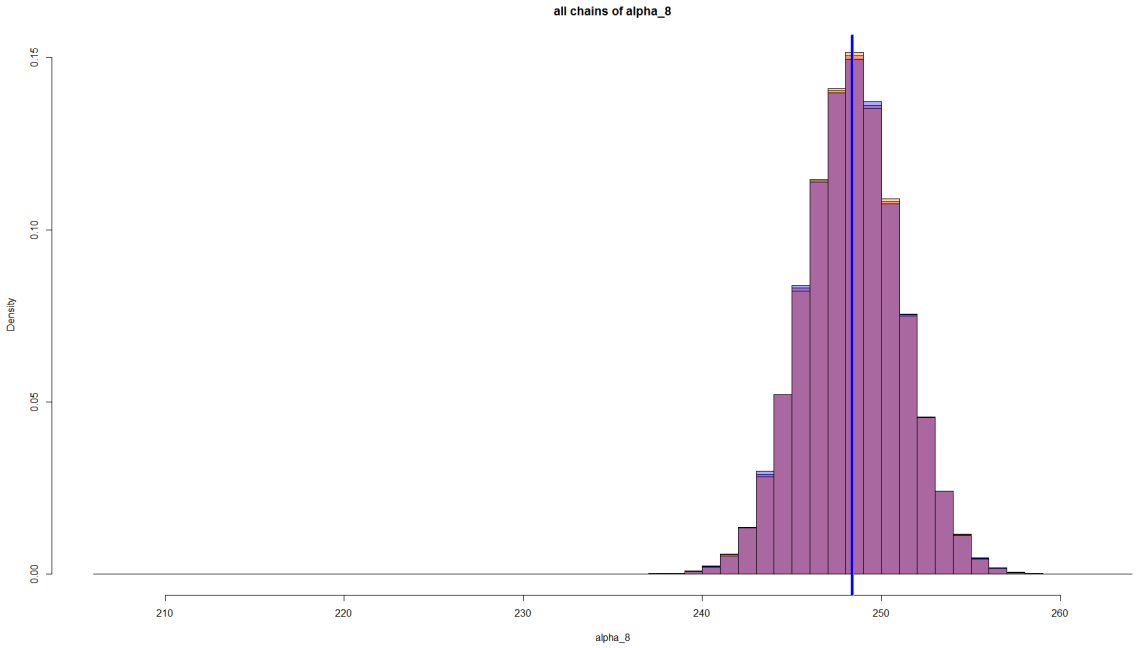
Then find the rat with the most linear behaviour.

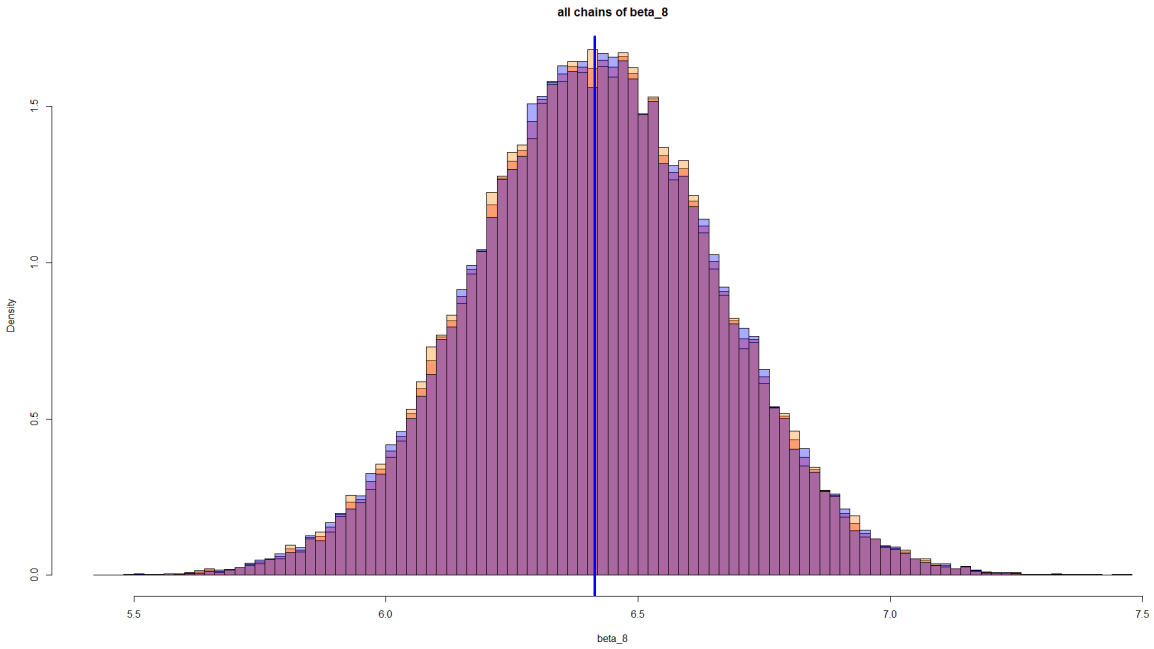
```
maxR2 = which.max(R2all)
maxR2Value = max(R2all)
i = unname(maxR2)

print('the best fit is achieved for:')
print(paste('rat_',i,sep = ""))
```

```
> the best fit is achieved for: rat_8
```

The rat_8 is then the rat with the best linear fit. Let's see the distribution of the chains that gave us alpha[8] and beta[8] on which the linear regression for rat_8 is achieved. We need to point out here that JAGS used 2 chains to estimate each parameter. We can plot 3 histograms in one single plot, where 2 are from the 2 different chain, and 1 is from a simulation where we merge the 2 chains.

```
threeHistMY(alphaChain1[,i],alphaChain2[,i],alphaChain[,i],mean(alphaChain1[,i]),mean(
    red↪ alphaChain2[,i]),alphaVect[i],F,0,paste('all chains of alpha_',i,sep=""),paste
    red↪ ('alpha_',i,sep=""))

threeHistMY(betaChain1[,i],betaChain2[,i],betaChain[,i],mean(betaChain1[,i]),mean(
    red↪ betaChain2[,i]),betaVect[i],F,0,paste('all chains of beta_',i,sep=""),paste('
    red↪ beta_',i,sep=""))
```

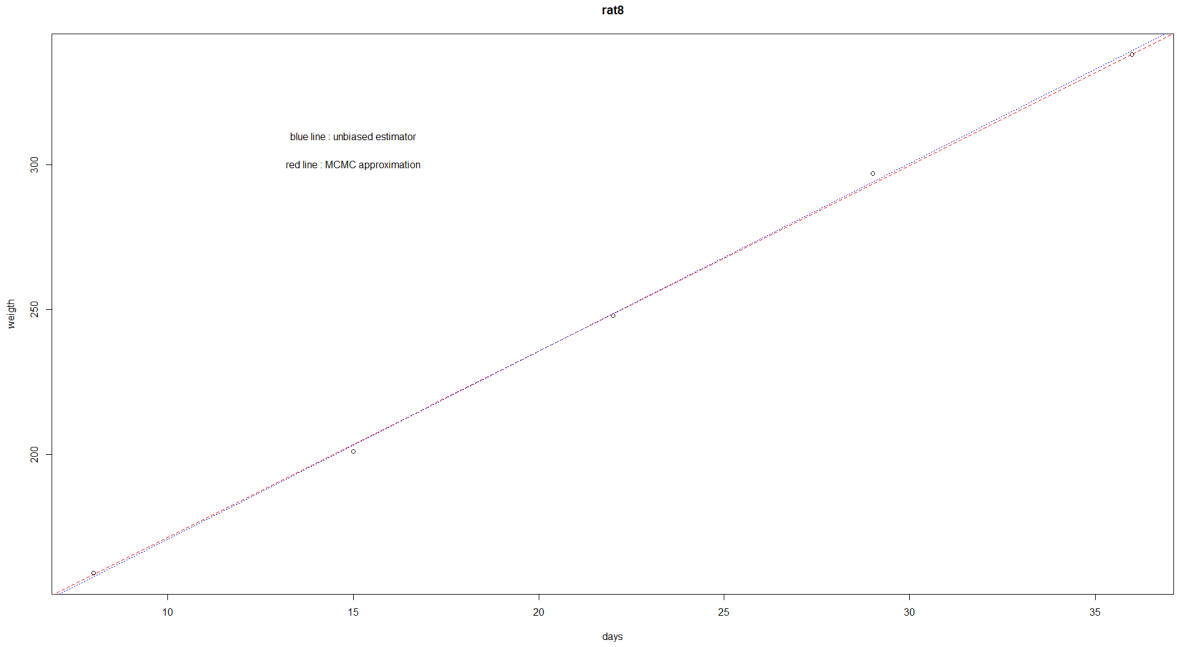




We can now finally plot the regression where we also plot the line with A and B, the usual estimator for linear regression. As we can see the 2 lines are really alike.

```
B = calc_B(d$x,d$Y[i,],xBar,length(d$x))
A = calc_A(d$Y[i,],B,xBar)

nameR = names(minR2)
plot(d$x,d$Y[i,], main=nameR, ylab='weigth', xlab='days')
abline(a = alphaVect[i]-betaVect[i]*xBar, b=betaVect[i], col='red', lty='dashed',lw=1)
abline(a = A, b=B, col='blue', lty='dotted',lw=1)
```

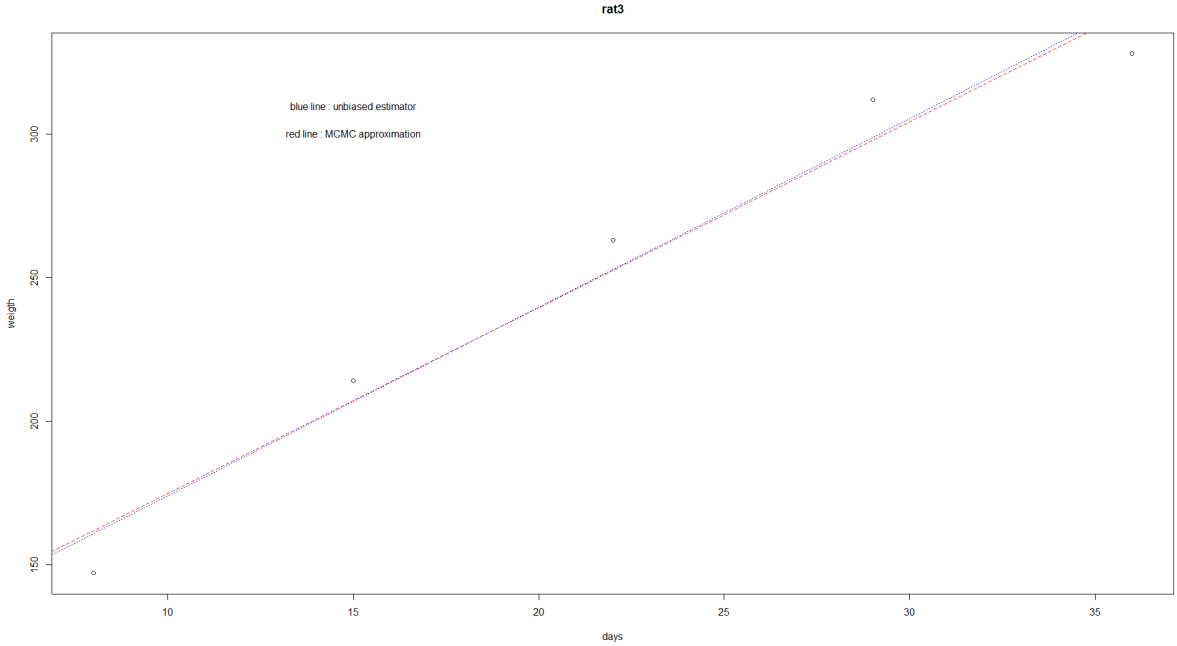
```
> A
[1] 105.9143
> alphaVect[i]-betaVect[i]*xBar
[1] 107.217
> B
[1] 6.485714
> betaVect[i]
[1] 6.416422
```



As we did with the best fit, we can now also find the rat with the worst linear regression. Those are the results:

```
minR2 = which.min(R2all)
minR2Value = min(R2all)
i = unname(minR2)
print('the worst fit is achieved for:')
print(paste('rat_',i,sep = ""))
```

```
> the worst fit is achieved for: rat_3
```



In the end we used MCMC method to approximate many parameters. Just considering α_i and β_i we have 30 approximation each. Each of those approximation have an error equal to the variance of the approximation. Such variance cannot be computed but just estimated in the following way:

$$\mathbb{E}\left[(\hat{I}_t - I)^2\right] = Var(\hat{I}_t) \simeq \hat{Var}(\hat{I}_t) = \frac{1}{t}\left(\hat{\gamma}_0 + 2\sum_{k=1}^{t-1}\hat{\gamma}_k\right)$$

$$\hat{\gamma}_k = \frac{1}{t-k}\sum_{i=1}^{t-k}\left(X_i - \hat{I}_t\right)\left(X_{i+k} - \hat{I}_t\right)$$

I computed such estimate for each of the 30 α_i and each of the β_i, then I normalized by dividing the absolute value of such error for the used approximation and then I computed the mean value as follows:

$$Err_\alpha = \frac{100}{30}\sum_i \frac{|\hat{Var}(\hat{I}_{\alpha_i})|}{\hat{I}_{\alpha_i}} = 0.063\%$$

$$Err_\beta = \frac{100}{30}\sum_i \frac{|\hat{Var}(\hat{I}_{\beta_i})|}{\hat{I}_{\beta_i}} = 0.19\%$$

Anyway as we can see the behaviour for this rat is not linear and might be exponential. We now implement a model that is able to use and exponential expected value.

3 Rats: a normal hierarchical model with exponential expected value

$Y_{ij} \sim N(\mu_{ij}, \tau_c)$

$\mu_{ij} = \alpha_i + \beta_i \gamma_i^{(x_j - \bar{x})}$
$\bar{x} = \frac{1}{5}\sum_j x_j = 22$

$\alpha_i \sim N(\alpha_c, \tau_\alpha)$
$\beta_i \sim N(\beta_c, \tau_\beta)$
$\gamma_i \sim U(0.5, 1)$

$\alpha_c, \beta_c, \sim N(0, 10^{-4})$
$\tau_c, \tau_\alpha, \tau_\beta \sim G(0.001, 0.001)$

The jags code for the model:

```
model {
        for (i in 1:N) {

                for (j in 1:T) {

                        mu[i,j] <- alpha[i] - beta[i]*gamma[i]^(x[j] - x.bar);
                        Y[i,j]   ~ dnorm(mu[i,j],tau.c) }

                alpha[i] ~ dnorm(alpha.c,tau.alpha);
                beta[i]  ~ dnorm(beta.c,tau.beta);
               gamma[i] ~ dunif(0.5, 1); }

        alpha.c    ~ dnorm(0,1.0E-4);
        beta.c     ~ dnorm(0,1.0E-4);
```

```
        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);
        sigma     <- 1.0/sqrt(tau.c);
        x.bar     <- mean(x[]); }
```

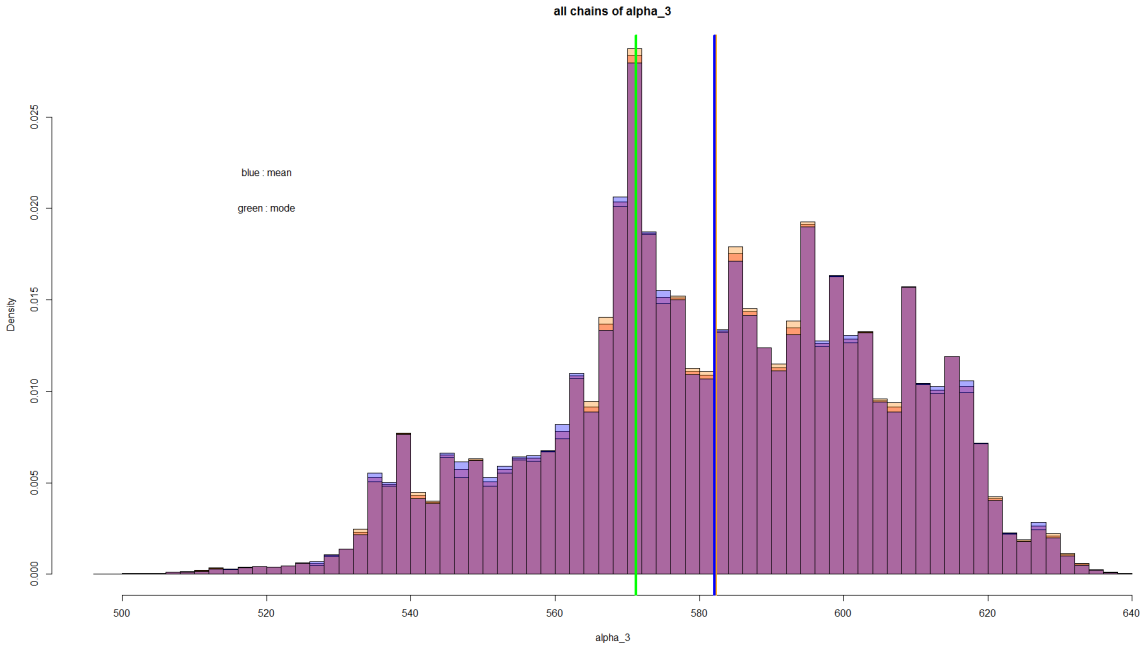
The R code:

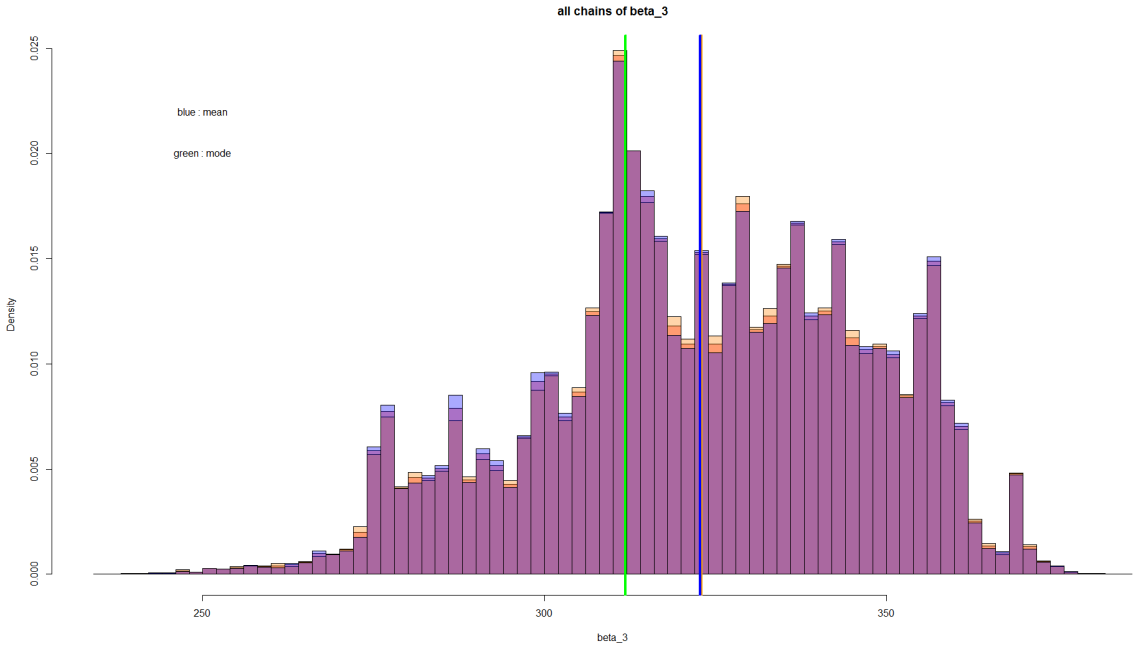
```
initsFileExp_i <- read.jagsdata("rats-init_exp_i.R")
initsExp_i=function(){
  initsFileExp_i }
attributes(initsFileExp_i)$names
paramsExp_i = c("tau.c","alpha","tau.beta","beta","tau.alpha","beta.c","gamma","alpha.c
    red↪ ")
ratJagsExp_i=jags(data=d,inits=initsExp_i,parameters.to.save=paramsExp_i,model.file="
    red↪ rats_paolo_exp_i.txt",
                  n.chains=nc,n.thin = 1,n.iter=ni,n.burnin =nb)

ratJagsExp_i$model

alphaVectExp_i = ratJagsExp_i$BUGSoutput$mean$alpha
betaVectExp_i = ratJagsExp_i$BUGSoutput$mean$beta
alphaCExp_i = as.vector(ratJagsExp_i$BUGSoutput$mean$alpha.c)
betaCExp_i = as.vector(ratJagsExp_i$BUGSoutput$mean$beta.c)
gammaExp_i = ratJagsExp_i$BUGSoutput$mean$gamma
xBar = mean(d$x)
```

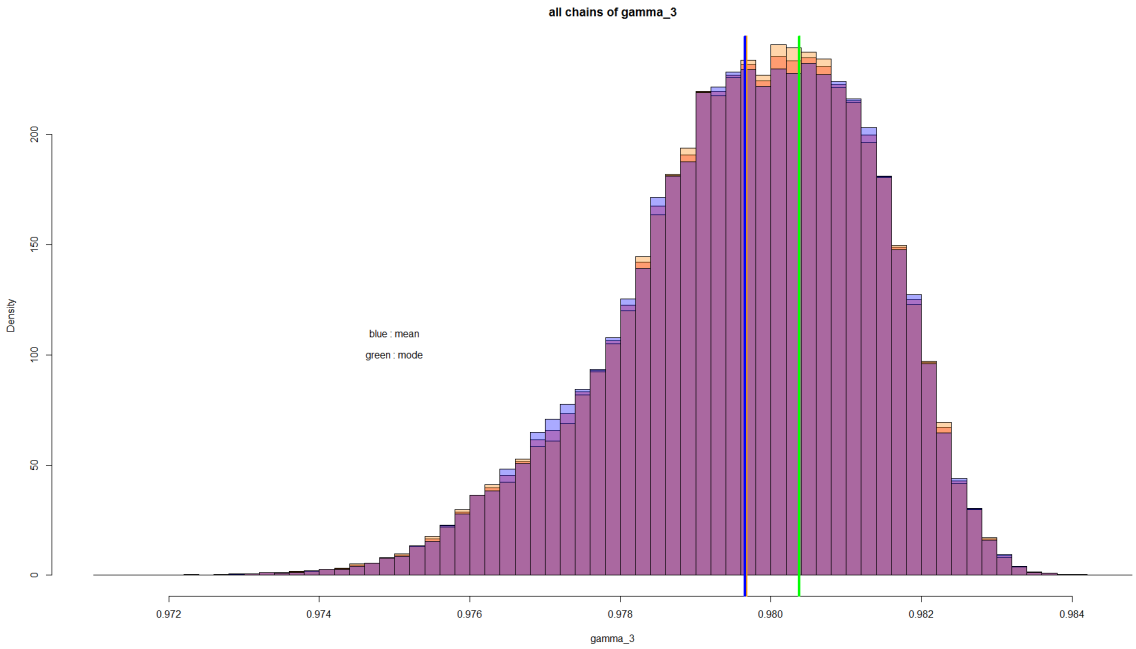
To see how the new model behaves, we always use rat_3, the rat that it seemed to have an exponential behaviour. The first thing we will do, it will be looking at the histograms of alpha[3], beta[3] and gamma[3] chains. This way we will be comparing modes and expected values.

```
i = 3

cols = 1:30
#ratJagsExp_i$BUGSoutput$sims.matrix[1,cols]
alphaChain1 = ratJagsExp_i$BUGSoutput$sims.matrix[1:nsim,cols]
alphaChain2 = ratJagsExp_i$BUGSoutput$sims.matrix[(nsim+1):(nsim*2),cols]
alphaChain = ratJagsExp_i$BUGSoutput$sims.matrix[,cols]
alphaVectExp_i_mode = calc_MoreMode(alphaChain)

threeHistMY(alphaChain1[,i],alphaChain2[,i],alphaChain[,i],mean(alphaChain1[,i]),mean(
    red↪ alphaChain2[,i]),alphaVectExp_i[i],T,alphaVectExp_i_mode[i],paste('all chains
    red↪ of alpha_',i,sep=""),paste('alpha_',i,sep=""))

cols = 32:61
#ratJagsExp_i$BUGSoutput$sims.matrix[1,cols]
betaChain1 = ratJagsExp_i$BUGSoutput$sims.matrix[1:nsim,cols]
betaChain2 = ratJagsExp_i$BUGSoutput$sims.matrix[(nsim+1):(nsim*2),cols]
betaChain = ratJagsExp_i$BUGSoutput$sims.matrix[,cols]
betaVectExp_i_mode = calc_MoreMode(betaChain)

threeHistMY(betaChain1[,i],betaChain2[,i],betaChain[,i],mean(betaChain1[,i]),mean(
    red↪ betaChain2[,i]),betaVectExp_i[i],T,betaVectExp_i_mode[i],paste('all chains of
    red↪ beta_',i,sep=""),paste('beta_',i,sep=""))

cols = 64:93
#ratJagsExp_i$BUGSoutput$sims.matrix[1,cols]
gammaChain1 = ratJagsExp_i$BUGSoutput$sims.matrix[1:nsim,cols]
gammaChain2 = ratJagsExp_i$BUGSoutput$sims.matrix[(nsim+1):(nsim*2),cols]
gammaChain = ratJagsExp_i$BUGSoutput$sims.matrix[,cols]
gammaExp_i_mode = calc_MoreMode(gammaChain)
```

```
threeHistMY(gammaChain1[,i],gammaChain2[,i],gammaChain[,i],mean(gammaChain1[,i]),mean(
    red↪ gammaChain2[,i]),gammaExp_i[i],T,gammaExp_i_mode[i],paste('all chains of
    red↪ gamma_',i,sep=""),paste('gamma_',i,sep=""))
```

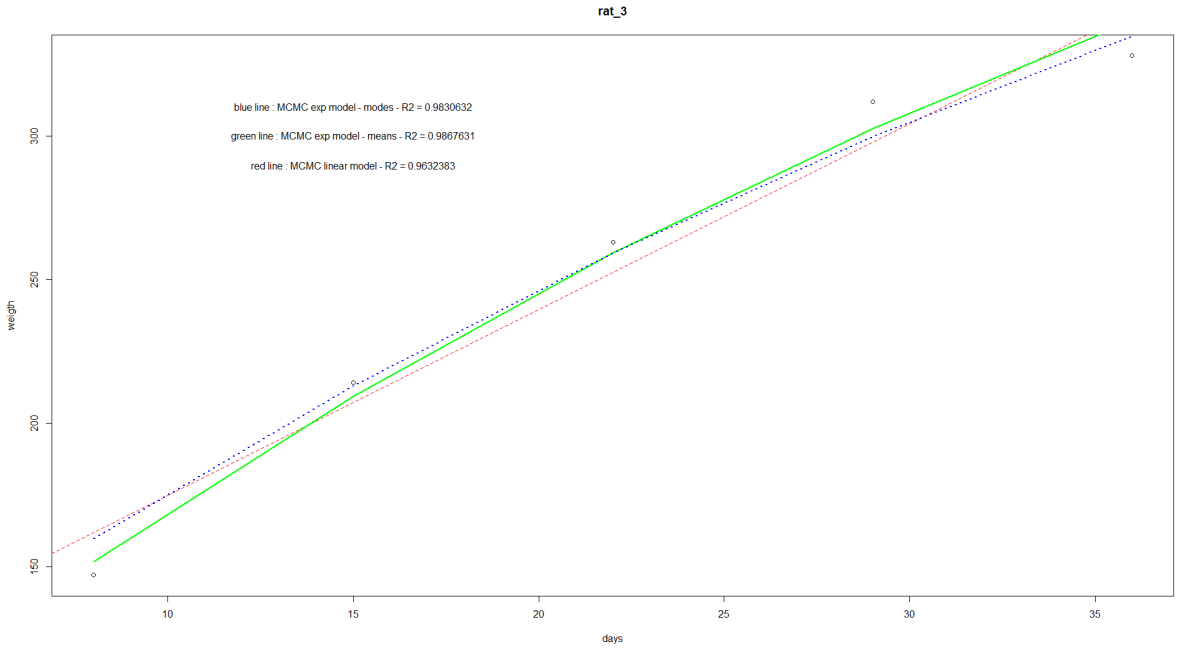






As we can see each plot has always 3 histograms, 2 different chains for the same parameter and an overall one. Like before those 3 histograms are similar, but now the mode and the expected value are not equal. We can compute the curve for rat_3 both with the expected values and mode of the overall chain.

```
i = 3

plot(d$x,d$Y[i,],m=R2allNames[i], ylab='weigth', xlab='days')
abline(a = alphaVect[i]-betaVect[i]*xBar, b=betaVect[i], col='red', lty='dashed',lw=1)
lines(d$x,alphaVectExp_i[i]-betaVectExp_i[i]*gammaExp_i[i]^(d$x-xBar),col="green",lw=2)
lines(d$x,alphaVectExp_i_mode[i]-betaVectExp_i_mode[i]*gammaExp_i_mode[i]^(d$x-xBar),col
    red ↪ ="blue",lty='dotted',lw=2)
```



The green that uses expected values looks more precise. To be sure we can use the coefficient of determination R^2, which the more is close to 1 the less are the difference between the data Y and the approximation given by the curve.

```
R2all_Exp_i = calc_R2all_exp(d$Y, alphaVectExp_i, betaVectExp_i, gammaExp_i, d$x, xBar)
R2all_Exp_i_mode = calc_R2all_exp(d$Y, alphaVectExp_i_mode, betaVectExp_i_mode,
    red↪ gammaExp_i_mode, d$x, xBar)
```

```
> R2all[3]
     rat3
0.9632383
> R2all_Exp_i[3]
     rat3
0.9867631
> R2all_Exp_i_mode[3]
     rat3
0.9830632
```

The results show a better performance of the exponential model in approximating the rat_3 as both R^2 for the modes and for the expected values are greater than the R^2 from the past linear model. Let's also compute the average R^2 for all rats.

```
> mean(R2all)
[1] 0.9938301
> mean(R2all_Exp_i)
[1] 0.9965007
> mean(R2all_Exp_i_mode)
[1] 0.9859929
```

The difference is now not as great. We will come back on the perfomance of the model later.

4 Birats: a bivariate Normal hierarchical model with linear expected value

$$Y_{ij} \sim N(\mu_{ij}, \tau_c)$$

$$\mu_{ij} = \beta_{1i} + \beta_{2i}\, x_j$$
$$\bar{x} = \frac{1}{5}\sum_j x_j = 22$$

$$\beta_i = (\beta_{1i}, \beta_{2i}) \sim MVN(\mu_\beta, \Omega_\beta)$$

$$\tau_c \sim G(0.001, 0.001)$$

The jags code for the model:

```
model {

    for (i in 1:N) {

      for (j in 1:T) {

        Y[i,j] ~ dnorm(mu[i,j],tau.c);  #
```

```
        mu[i,j] <- beta[i,1] + beta[i,2] * x[j];   }

    beta[i,] ~ dmnorm(mu.beta[],Omega.beta[,]);  # bivariate Normal }

  tau.c   ~ dgamma(1.0E-3,1.0E-3);
  sigma <- 1.0/sqrt(tau.c);
```

The R code:

```
db <- read.jagsdata("birats-data.R")
initsFileb <- read.jagsdata("birats-inits.R")
initsb=function(){
  initsFileb }
attributes(initsFileb)$names
paramsb = c("tau.c", "tau.beta", "Omega.beta", "beta", "mu.beta")
ratJagsb=jags(data=db,inits=initsb,parameters.to.save=paramsb,model.file="birats4.bug",
                n.chains=nc, n.thin = 1, n.iter=ni, n.burnin =nb)
a = ratJagsb$BUGSoutput$mean$beta[,1]
b = ratJagsb$BUGSoutput$mean$beta[,2]
```

The results show an R^2 not much different from the first linear model, always finding the best linear expected value for each rat. The fact that the angular coefficients β_{2i} and the intercepts β_{1i} are drawn from a multivariate normal distribution let us have a formula where we do not need to normalize with $\bar{x}$.

$$\mu_{ij} = \beta_{1i} + \beta_{2i}\, x_j$$

```
> mean(calc_R2all_biv(d$Y,a,b,d$x,xBar))
[1] 0.9933959
```

5 Comparing the models

What I understood by running the different models, was that the performance really changed by varying the number of iterations and the percentage of burn-in. A correct measure to analyse the performance of the model is to use the measure DIC. Such measure is the deviance information criterion and it uses the deviance measure. The deviance D gives, given a parameter value θ, some kind of error using the log-likelihood function:

$$D(\theta) = -2\, logL(\theta)$$

Once the posterior means $\bar{\theta}$ are computed, you can measure the their deviance as:

$$D(\bar{\theta}) = -2\, logL(\bar{\theta})$$

Taking all steps i in the different MCMCs gives a vector of parameters θ_i with which a $D(\theta_i)$ can be computed. By takeing all $D(\theta_i)$ is is possible to compute an average called posterior mean of the deviance as follows:

$$\bar{D} = \frac{1}{t}\sum_{i=1}^{t} D(\theta_i)$$

It is now possible to compute the effective numbers of parameters p_D in the model as follows:

$$p_D = \bar{D} - D(\bar{\theta})$$

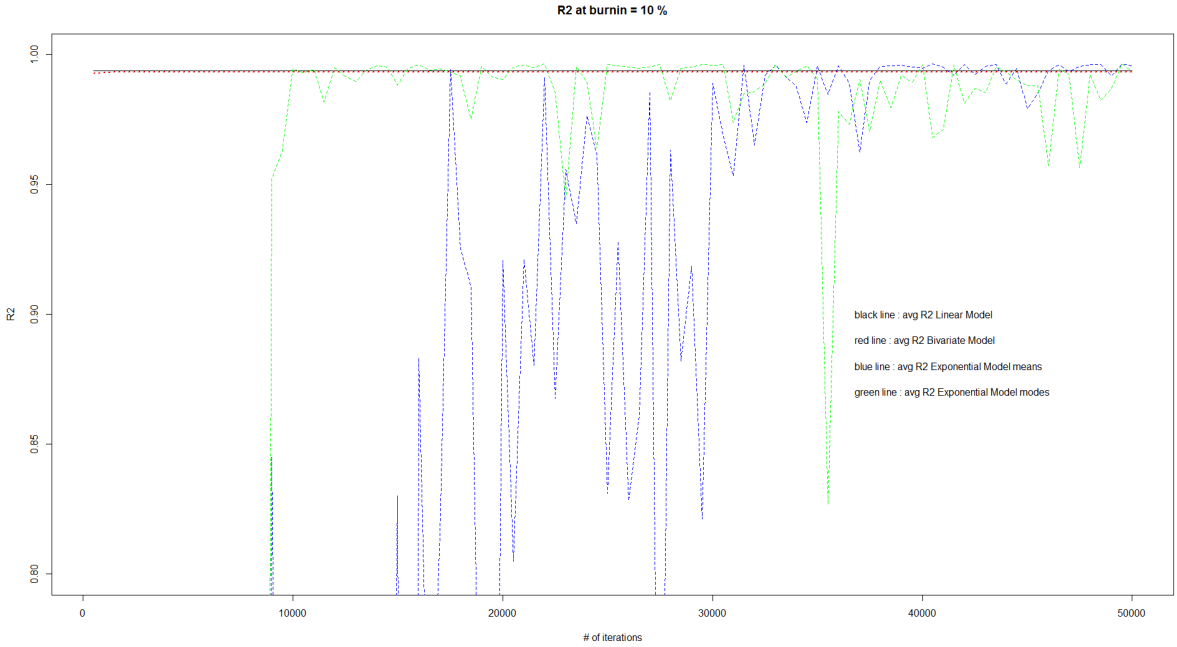
Then the DIC is computed as follows:

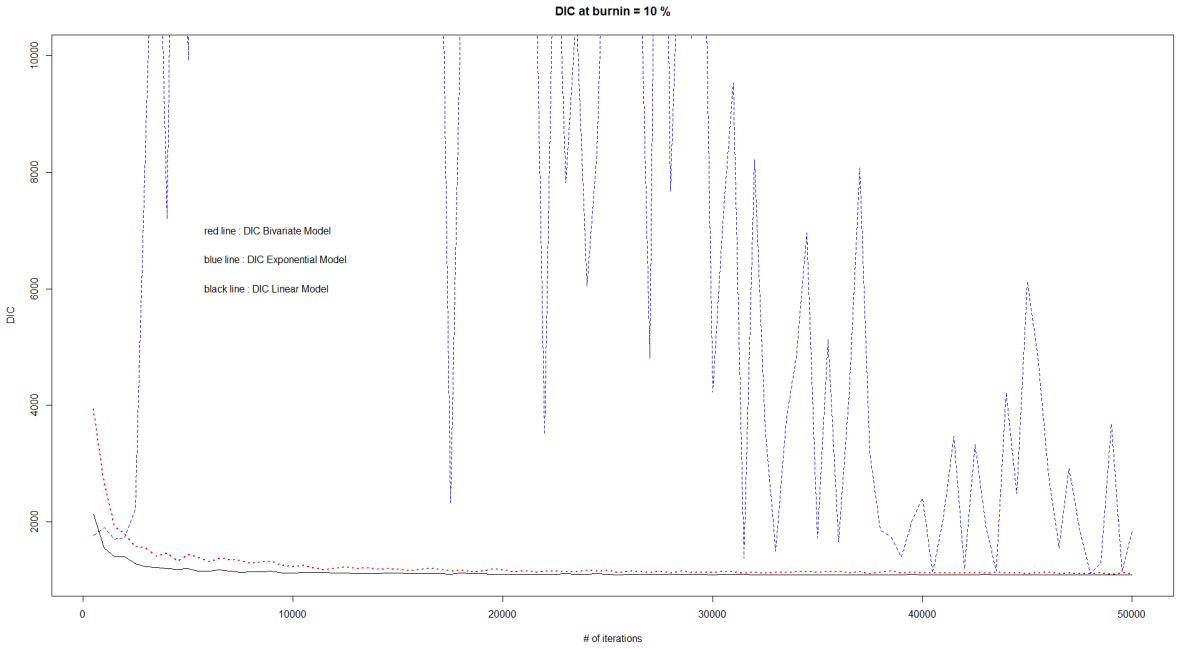
$$DIC = D(\bar{\theta}) + 2\,p_D$$

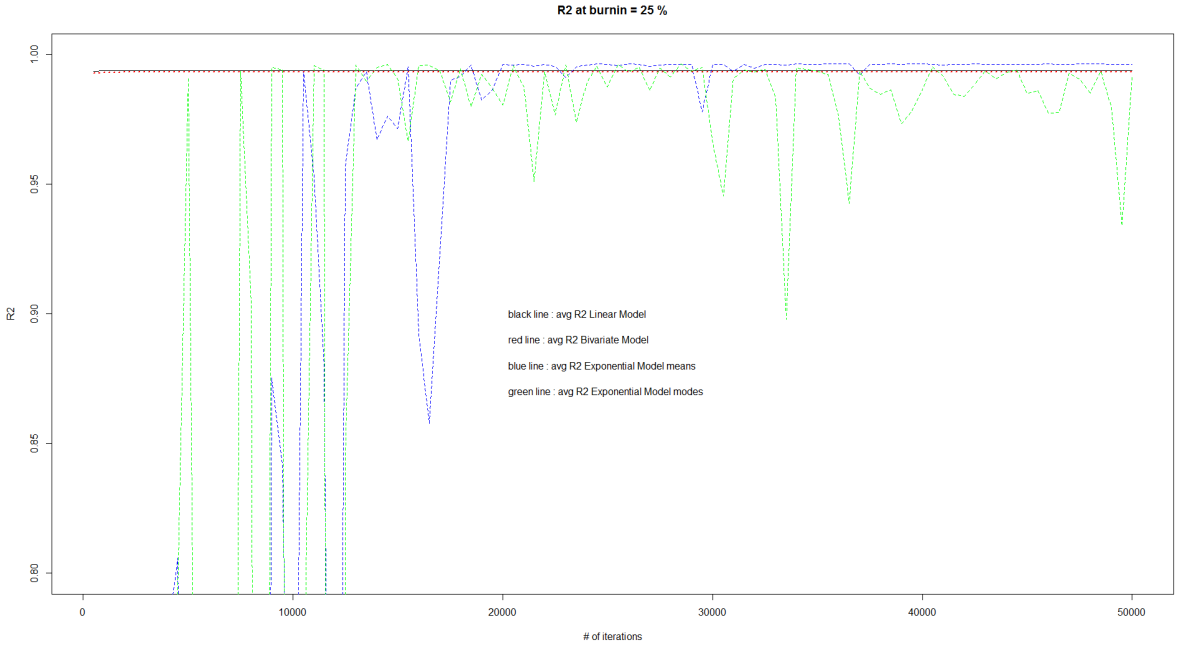
and we can use it to measure the performance of models in terms of uncertainty of the results. Given a model M_1 and a model M_2, M_1 is better than M_2 if:

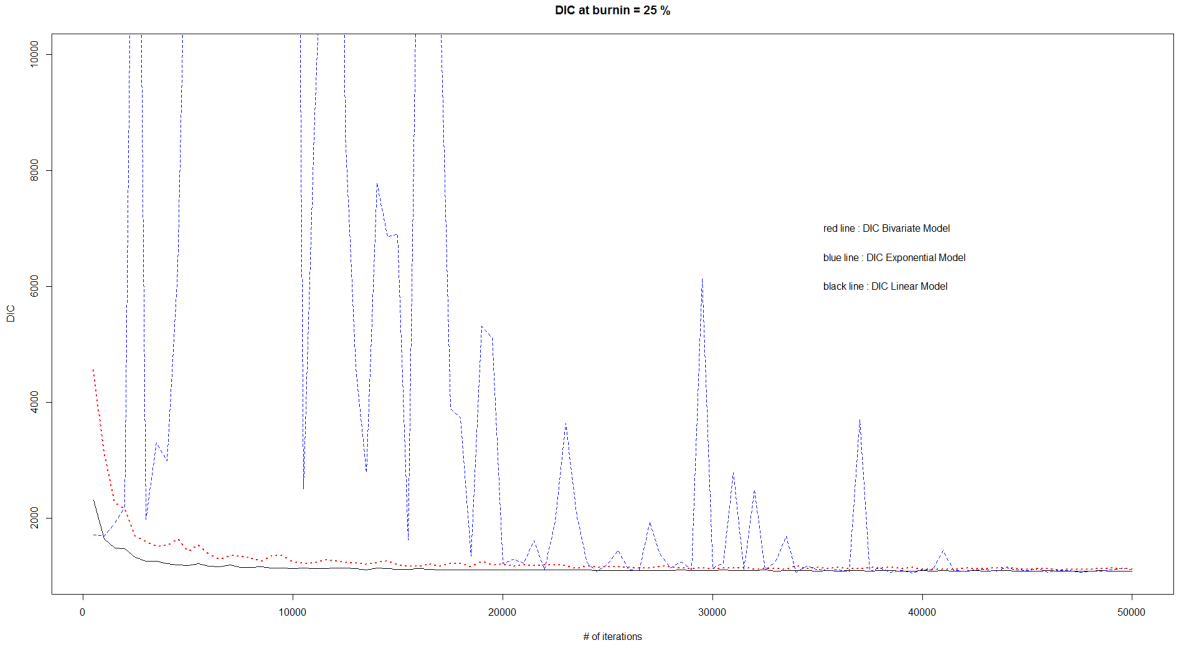
$$DIC_1 < DIC_2$$

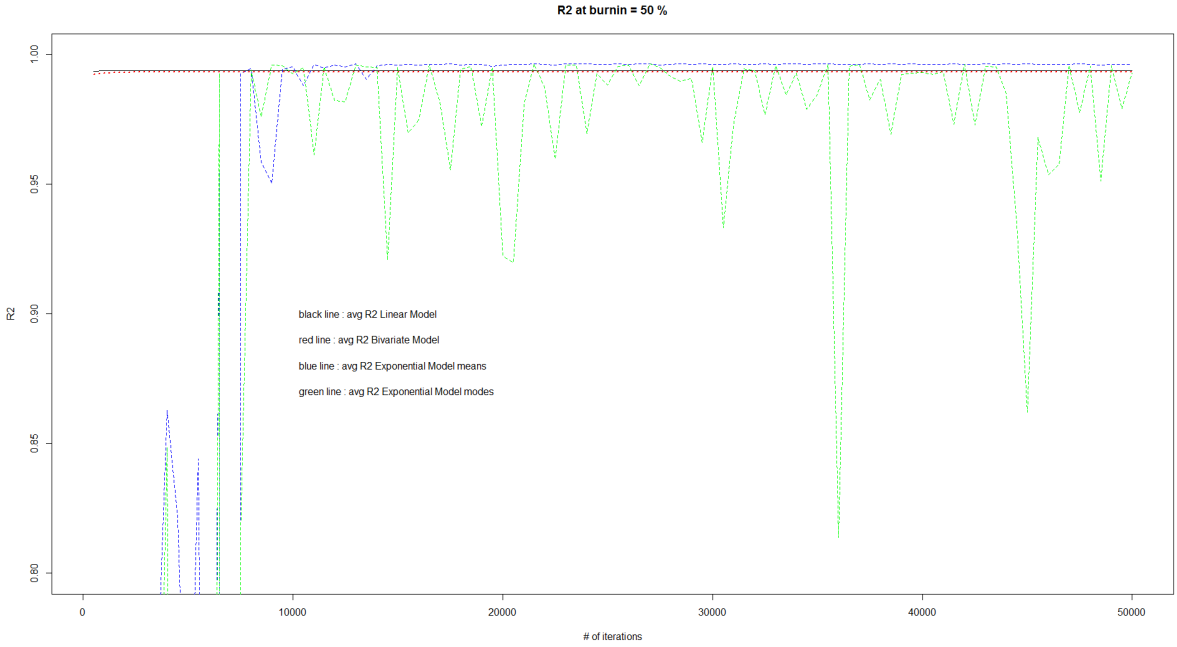
Those plots, that required a long time to be computed, show how DIC and the average R^2 vary as the number of iteration increases. This as been done for 3 different percentages of burn-in: 10%, 25% and 50%.

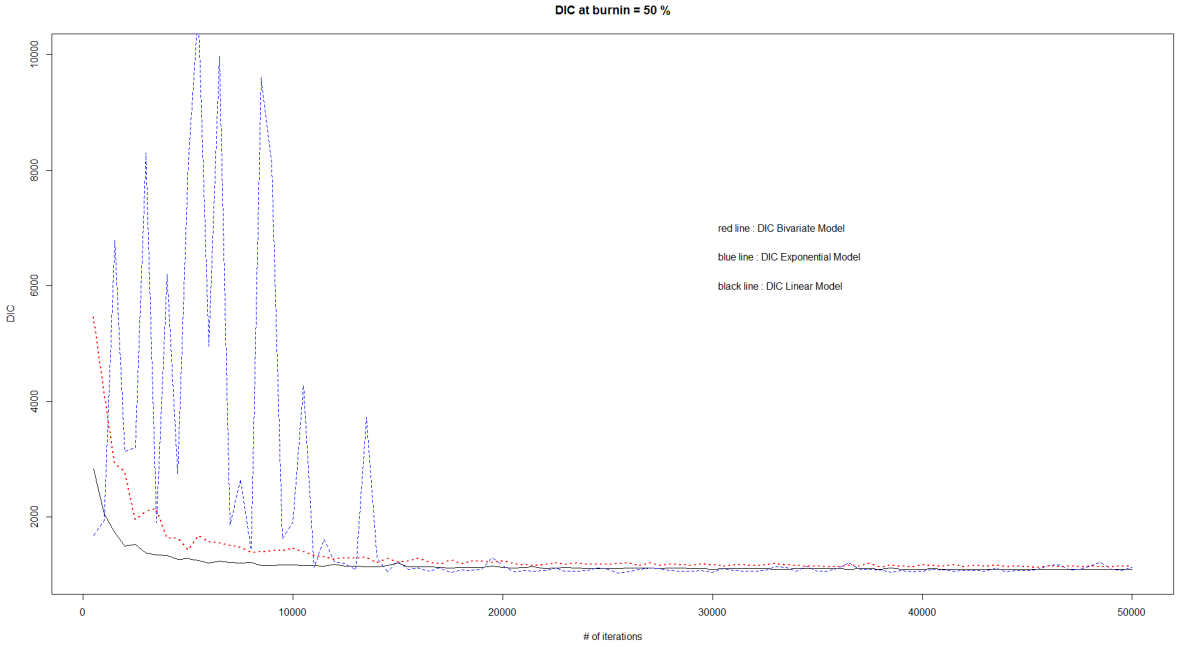





R2 at burnin = 25 %
R2
1.00
0.95
0.90
0.85
0.80
0
10000
20000
30000
40000
50000
# of iterations
black line : avg R2 Linear Model
red line : avg R2 Bivariate Model
blue line : avg R2 Exponential Model means
green line : avg R2 Exponential Model modes


DIC at burnin = 25 %
DIC
10000
8000
6000
4000
2000
0
10000
20000
30000
40000
50000
# of iterations
red line : DIC Bivariate Model
blue line : DIC Exponential Model
black line : DIC Linear Model

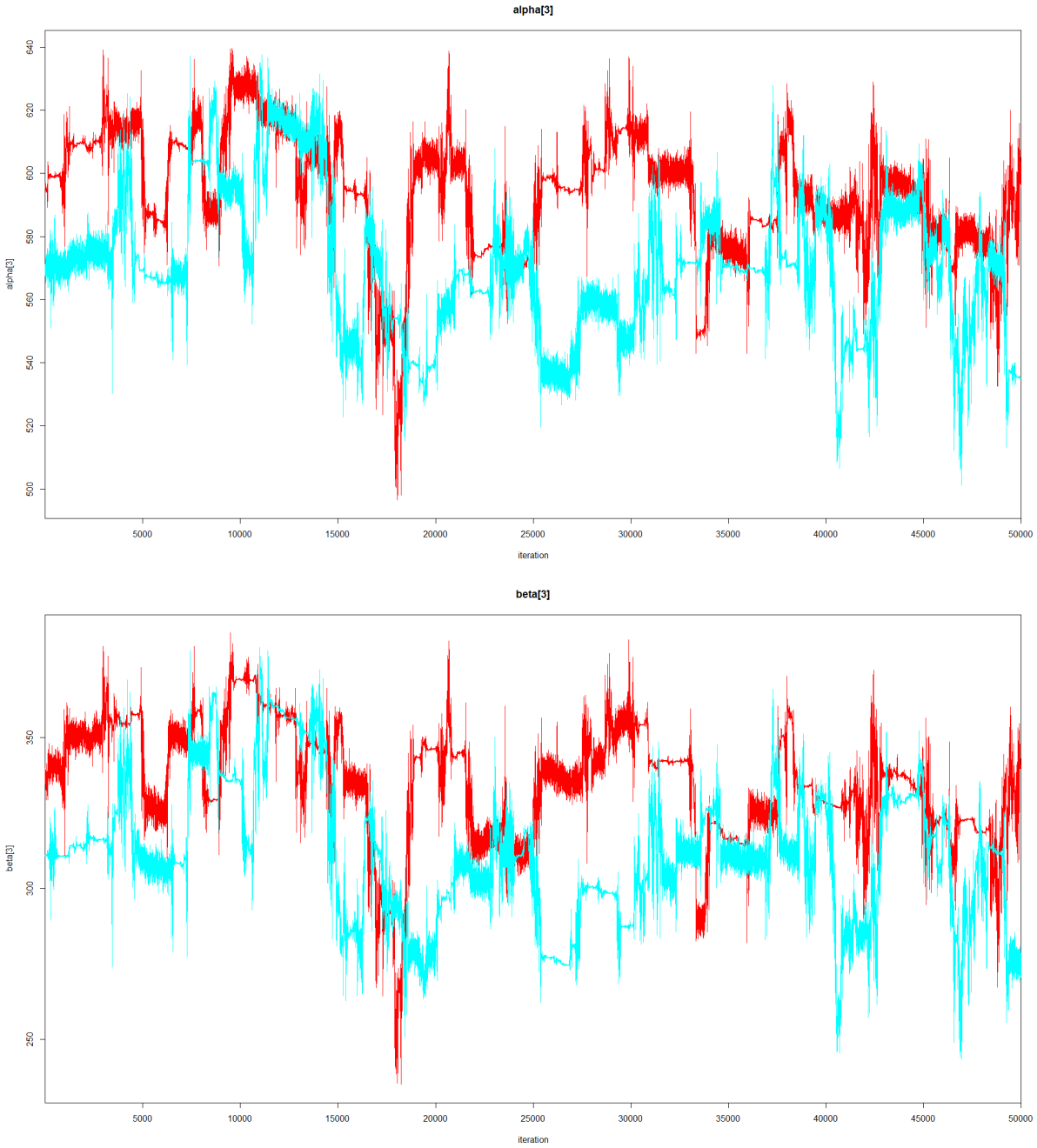

R2 at burnin = 50 %
R2
1.00
0.95
0.90
0.85
0.80
0
10000
20000
30000
40000
50000
# of iterations
black line : avg R2 Linear Model
red line : avg R2 Bivariate Model
blue line : avg R2 Exponential Model means
green line : avg R2 Exponential Model modes

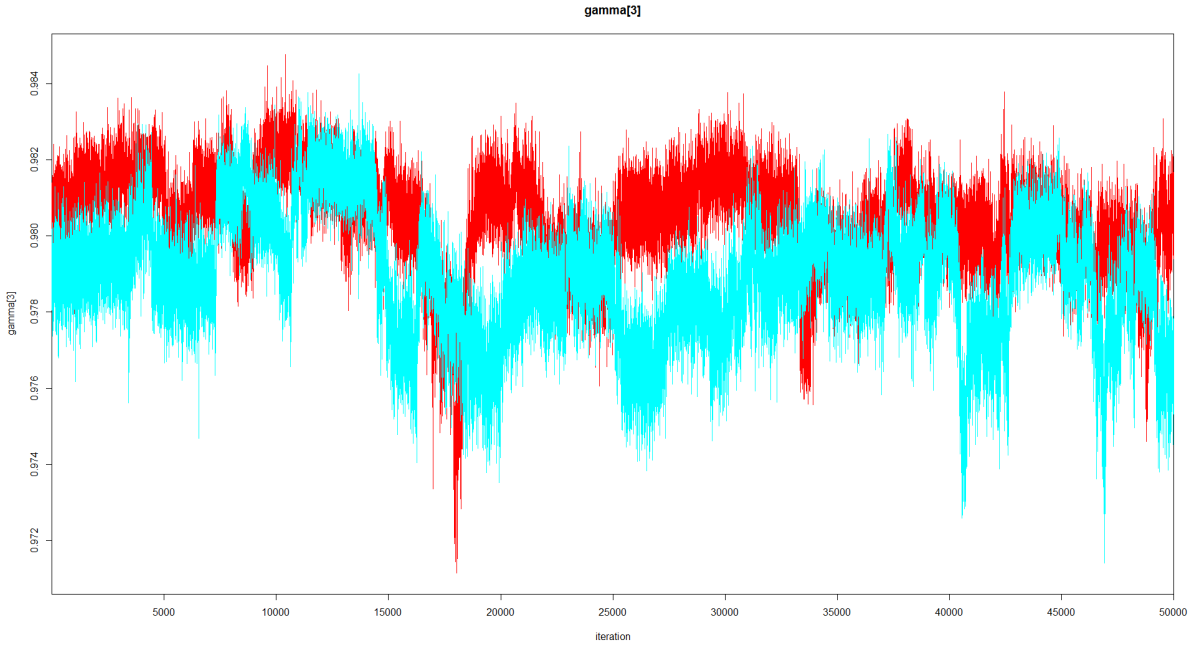

DIC at burnin = 50 %
DIC
10000
8000
6000
4000
2000
0
10000
20000
30000
40000
50000
# of iterations
red line : DIC Bivariate Model
blue line : DIC Exponential Model
black line : DIC Linear Model

6 Other plots

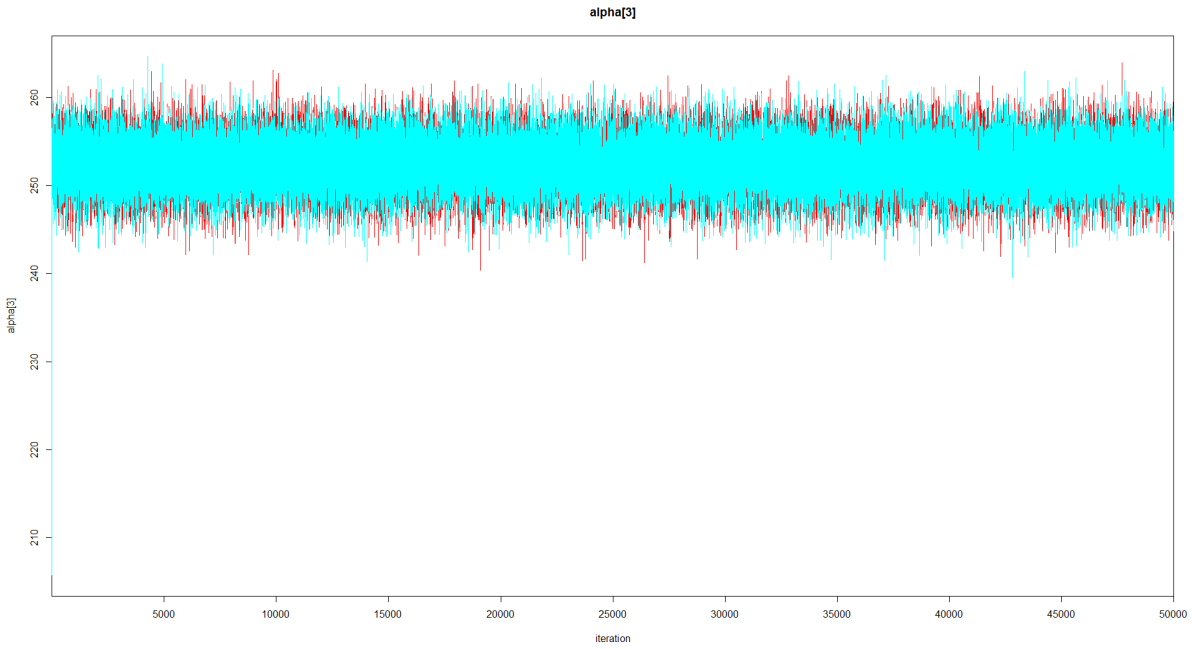
6.1 Trace-plots

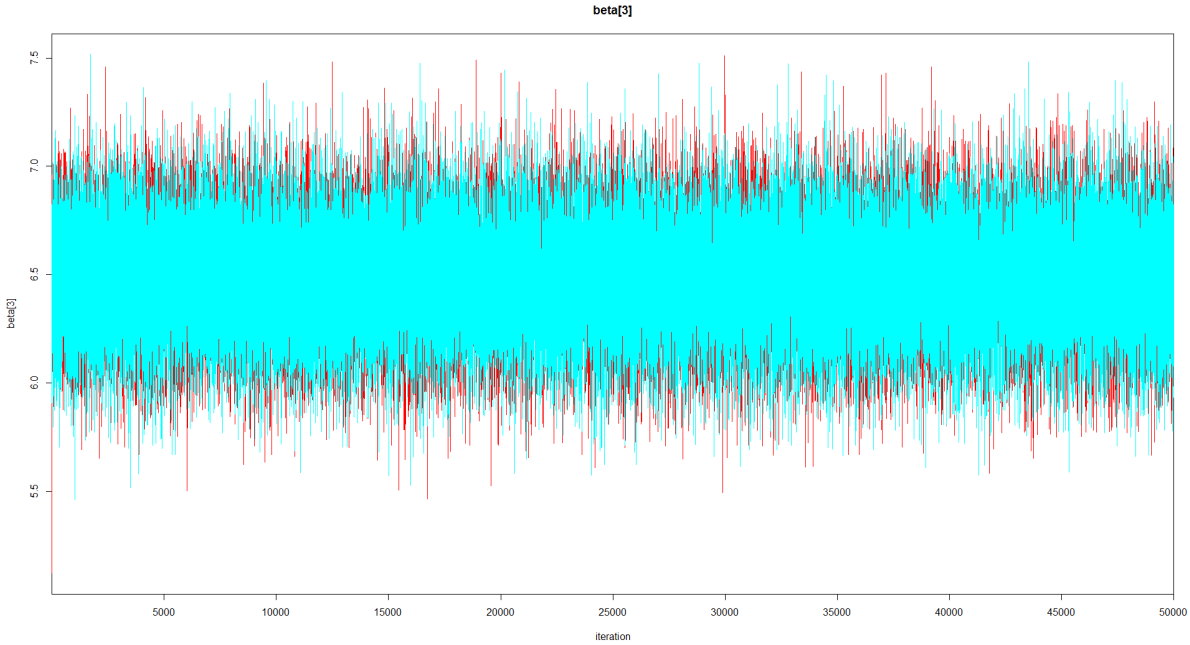
The traceplots are of 2 different colors because they contain 2 different chains each, but of course of the same parameter and with the same starting value. Now we will see the 3 trace-plots for the exponential model for α, β and γ.





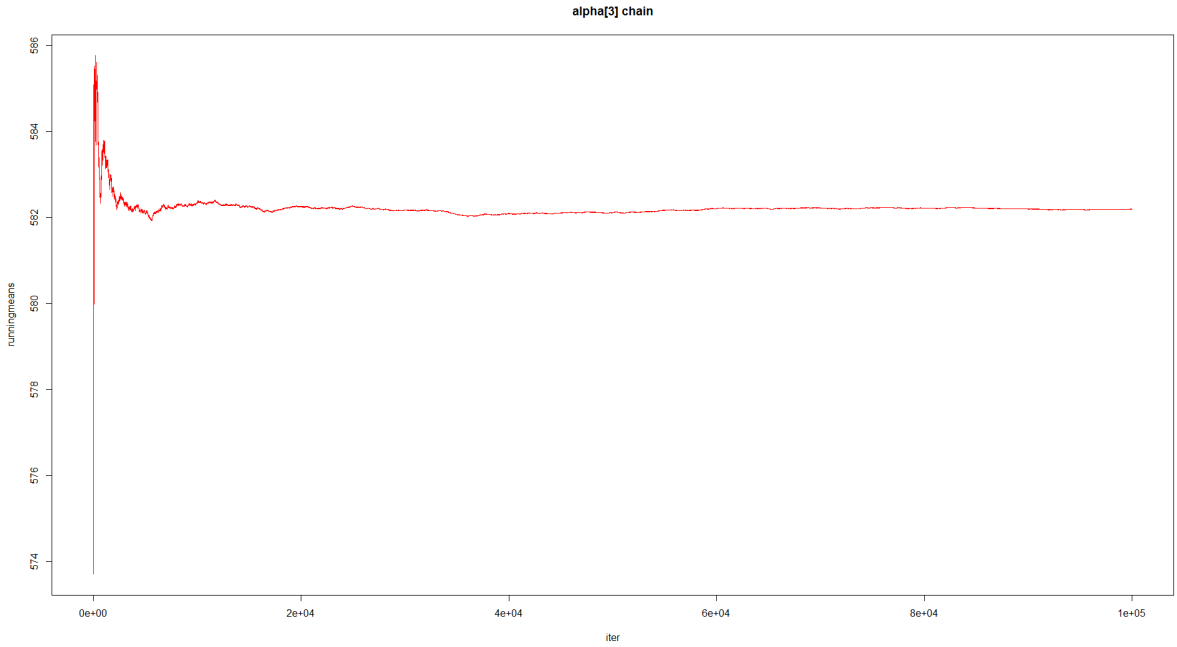
The trace-plots for α and β instead for the initial linear model are much different.

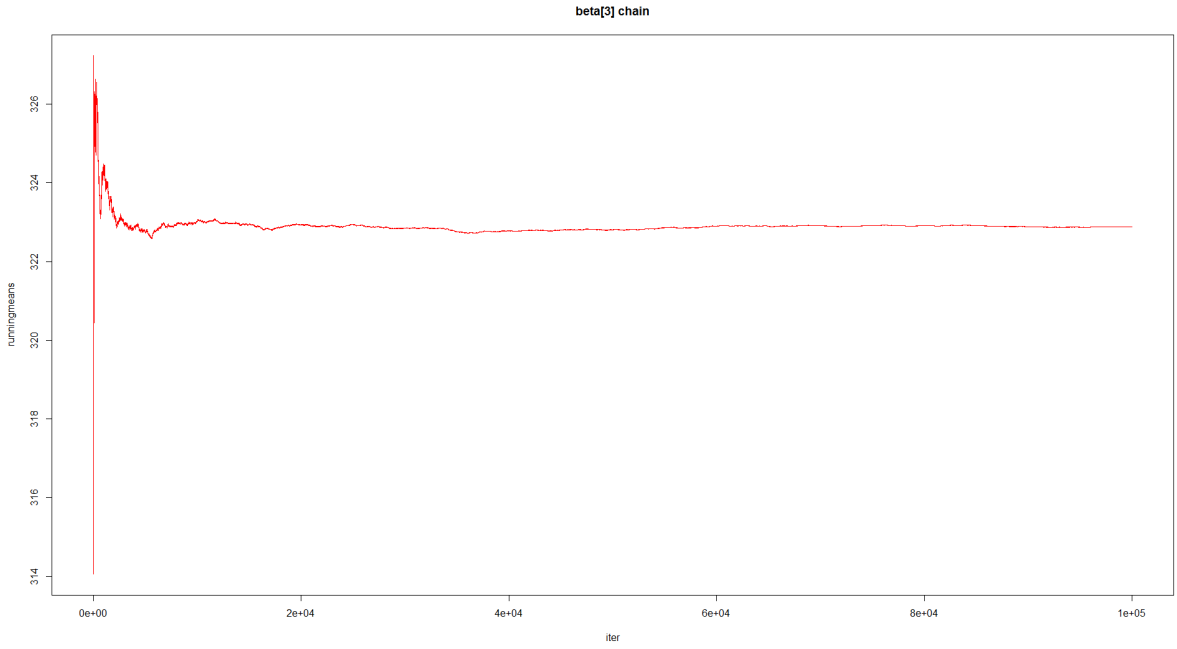


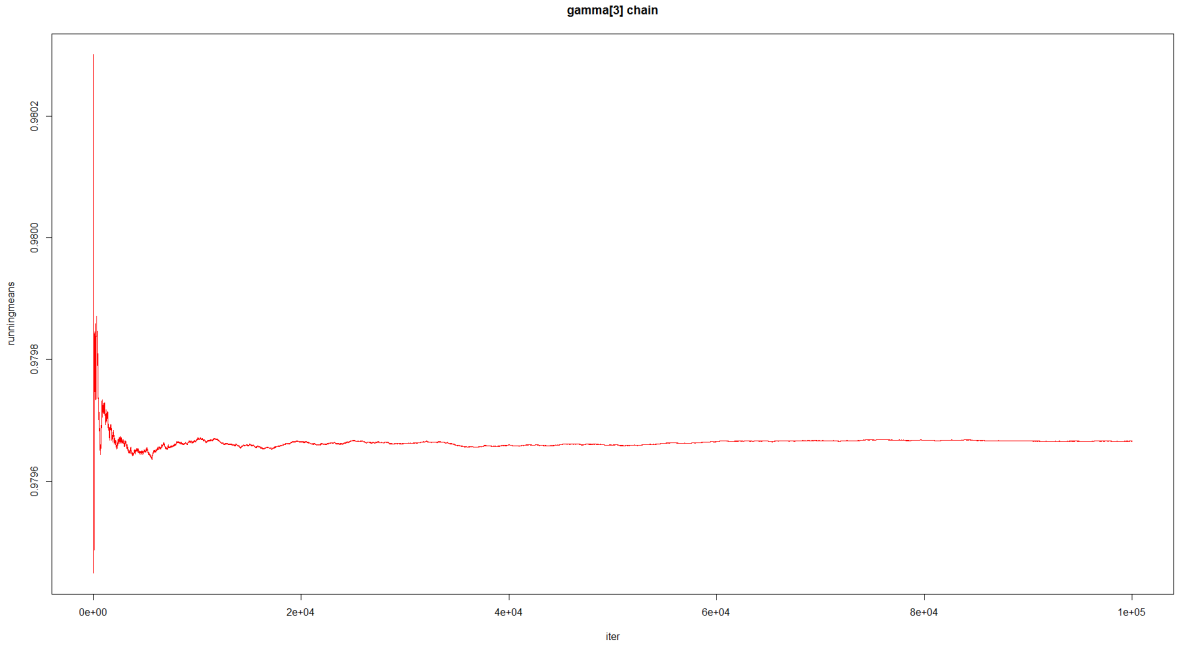

beta[3]
beta[3]
7.5
7.0
6.5
6.0
5.5
5000
10000
15000
20000
25000
30000
35000
40000
45000
50000
iteration

6.2 Running means

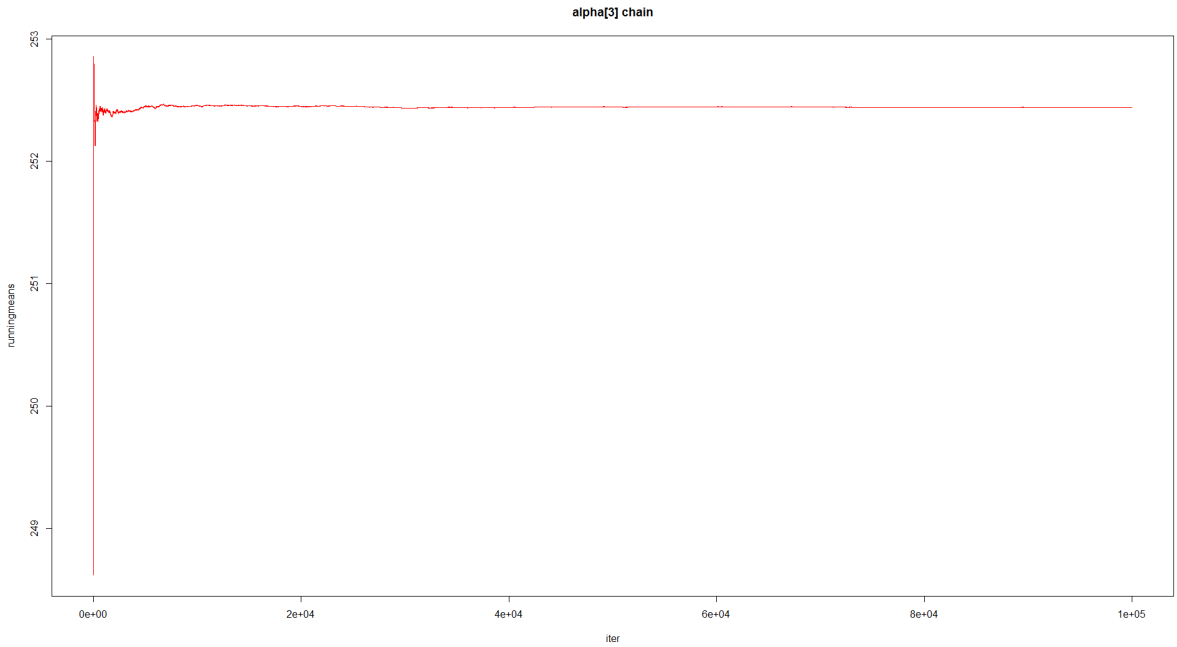
The running means are computed on the overall simulation containing the 2 chains of each parameter. Below there the 3 plots for the posterior means for the exponential model for α,β and γ computed at each step of the chain.

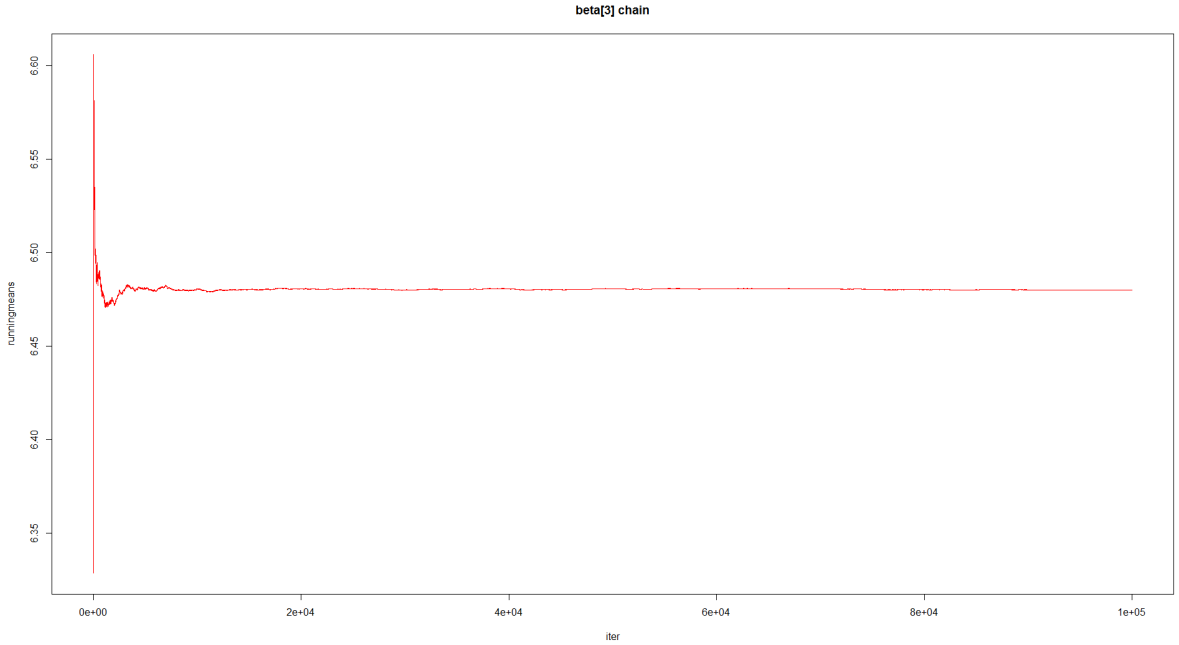






For α and β instead for the initial linear model are the following.




beta[3] chain
runningmeans
6.60
6.55
6.50
6.45
6.40
6.35
0e+00
2e+04
4e+04
6e+04
8e+04
1e+05
iter

7 Used functions

```
histMY = function(chain,val,stringa1,stringa2){
  hist(chain,freq = F, breaks=75, col="orange",xlab = stringa1, main= stringa2)
  abline(v =val,col='red', lw=4) }

calc_moda = function(vattore){
  z <- density(vattore)
  return(z$x[z$y==max(z$y)])}

calc_MoreMode = function(vetofvett){
  foriter=dim(vetofvett)[2]
  modeVect = rep(0, foriter)
  for (i in 1:foriter) {
    modeVect[i]=calc_moda(vetofvett[,i]) }
  return(modeVect)}

threeHistMY = function(chain1,chain2,chainAll,val1,val2,val3,booL,mode,stringa1,stringa2
    red↪ ){
  p1 <- hist(chain1,freq = F, breaks=75)
  p2 <- hist(chain2,freq = F, breaks=75)
  p3 <- hist(chainAll,freq = F, breaks=75)
  plot(p3,col=rgb(1,0,0,1/3), main=stringa1,xlab=stringa2, freq = F)
  plot(p2,col=rgb(1,0.5,0,1/3),add=T,freq = F)
  plot(p1,col=rgb(0,0,1,1/3),add=T,freq = F)
  abline(v =val3,col='red',lw=4)
  abline(v =val2,col='orange',lw=4)
  abline(v =val1,col='blue',lw=4)
  if (booL) {
    abline(v =mode,col='green',lw=4) }}

calc_A = function(uai,bi,xB) {
  return(mean(uai)-bi*xB) }

calc_B = function(xi,uaib,xbu,nu) {
  Sxy = (sum(xi*uaib)-xbu*sum(uaib))
  Sxx = (sum(xi^2)-nu*xbu^2)
  return(Sxy/Sxx) }

calc_R2all_lin = function(uailol,a,b,xlol,xBarlol){
  R2tutti<-rep(0, 30)
  R2tuttiNames<-rep(0, 30)
  for (i in 1:30) {
    SSreg = sum((uailol[i,]-(a[i]+b[i]*(xlol-xBarlol)))^2)
    SStot = sum((uailol[i,]-mean(uailol[i,]))^2)
    R2tuttiNames[i] = paste('rat', i, sep = "")
    R2tutti[i] = 1 - SSreg/SStot }
  names(R2tutti) = R2tuttiNames
  return(R2tutti)}

calc_R2all_biv = function(uailol,a,b,xlol,xBarlol){
  R2tutti<-rep(0, 30)
  R2tuttiNames<-rep(0, 30)
  for (i in 1:30) {
    SSreg = sum((uailol[i,]-(a[i]+b[i]*xlol))^2)
    SStot = sum((uailol[i,]-mean(uailol[i,]))^2)
    R2tuttiNames[i] = paste('rat', i, sep = "")
    R2tutti[i] = 1 - SSreg/SStot }
  names(R2tutti) = R2tuttiNames
```

```
  return(R2tutti)}

calc_R2all_exp = function(uailol,a,b,g,xlol,xBarlol){
  R2tutti<-rep(0, 30)
  R2tuttiNames<-rep(0, 30)
  for (i in 1:30) {
    SSreg = sum((uailol[i,]-(a[i]-b[i]*g[i]^(xlol-xBarlol)))^2)
    SStot = sum((uailol[i,]-mean(uailol[i,]))^2)
    R2tuttiNames[i] = paste('rat', i, sep = "")
    R2tutti[i] = 1 - SSreg/SStot }
  names(R2tutti) = R2tuttiNames
  return(R2tutti)}

gamma_k <- function(k,vet,Icap) {
  t = length(vet)
  somma = 0
  for (ki in 1:(t-k)) {
    somma = somma + (vet[ki] - Icap)*(vet[ki+k] - Icap) }
  return(somma/(t-k))}


varCap <- function(vet,Icap) {
  somma = 0
  tVar = length(vet)
  print('Summing up all gamma_k, from k = 1 to:')
  print(tVar)
  print('Starting now..')
  print(paste('iter. ', 1, sep = ""))
  for (klol in 1:(tVar-1)) {
    if (klol%%500==0) {
      print(paste('iter. ', klol, sep = "")) }
    somma = somma + gamma_k(klol,vet,Icap) }
  num = gamma_k(0,vet,Icap) + 2*somma
  #print(num)
  return(num/tVar) }
```