Homework #1

Stat4DS2+DS
https://elearning2.uniroma1.it/course/view.php?id=4951

Contents

Your Last+First Name ___MELE_*UMBERTO_Jr* Your Matricola *1388371* ————

Question 1

Part-a

1a) Illustrate the characteristics of the statistical model for dealing with the *Dugong*'s data. Lenghts (Y_i) and Ages (x_i) of 27 Dugongs have been recorded and the following (non linear) regression model is considered:

$$\begin{aligned} Y_i &\sim N(\mu_i, \tau^2) \\ \mu_i = f(x_i) &= \alpha - \beta\gamma^{x_i} \end{aligned}$$

Model parameters are $\alpha \in (1, \infty)$, $\beta \in (1, \infty)$, $\gamma \in (0, 1)$, $\tau^2 \in (0, \infty)$. Let us consider the following prior distributions:

$$\begin{aligned} \alpha &\sim N(0, \sigma_\alpha^2) \\ \beta &\sim N(0, \sigma_\beta^2) \\ \gamma &\sim Unif(0, 1) \\ \tau^2 &\sim IG(a, b))(InverseGamma) \end{aligned}$$

-Answer

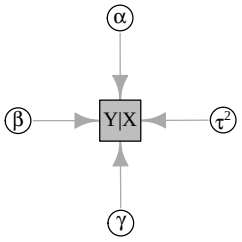
We have a regression problem, where we assume that $\mathbb{E}(Y|X = x) = \alpha - \beta \cdot \gamma^x$ and the noise ϵ is distribuited like a normal with variance τ^2.

Since we can absume that the parameters of the regression are indipendent we can rappresent this problem by a DAG , a graphical rapresentation.

Where variables $Y|X$ is the distribution that dipends on others parameters; while the other parameters of the problem are handling using some prior distribution that have the pourpose of give a kind of regularization to my loss function.

Important for our model is to write the join distribution in an appropriate way, so:

$$J(Y, X, \alpha, \beta, \gamma, \tau^2) = P(Y|X, \alpha, \beta, \gamma, \tau^2) \cdot P_\alpha(\alpha) \cdot P_\beta(\beta) \cdot P_\gamma(\gamma) \cdot P_{\tau^2}(\tau^2)$$



Part-b

1b) Derive the corresponding likelihood function

-Answer

We have a set of observed data of 27 Dugong's Ages and Lengths, so from them we want to build a non-linear regression function such that the error is distribuited like a Normal with variance τ^2 and $\mathbb{E}(Y|X) = \mu = \alpha - \beta \cdot \gamma^x$.

So computing the Maximum Likelihood Estimator we have almost the same effect, how we will see later, of the minimization of the least square error.

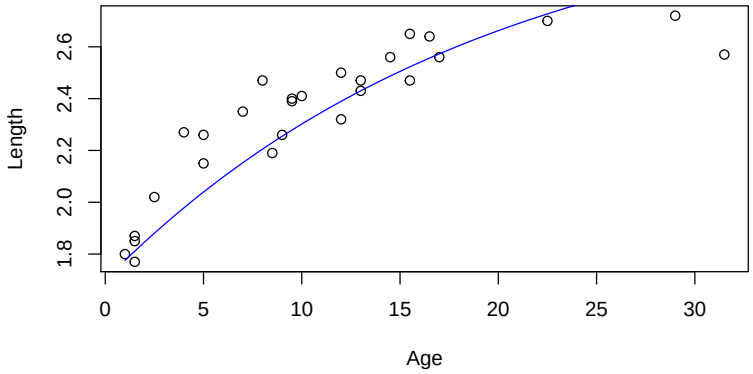
So, computing the likelihood:

$$L(\alpha, \beta, \gamma, \tau^2|(Y|X)) = \prod_{i=1}^{27} P((Y = y_i|X = x_i)|\alpha, \beta, \gamma, \tau^2) = \prod_{i=1}^{27} \frac{1}{\sqrt{2\tau^2\pi}} e^{-\frac{(y_i-\mu_i)^2}{2\tau^2}} = \prod_{i=1}^{27} \frac{1}{\sqrt{2\pi\tau^2}} e^{-\frac{(y_i-\alpha+\beta\gamma^{x_i})^2}{2\tau^2}}$$

and the negative log-likelihood function:

$$-l(\alpha, \beta, \gamma, \tau^2|(Y|X)) = 27 \cdot ln(\sqrt{2\pi\tau^2}) + \frac{1}{2\tau^2} \sum_{i=1}^{27} (y_i - \alpha + \beta\gamma^{x_i})^2 \propto \sum_{i=1}^{27} (y_i - \alpha + \beta\gamma^{x_i})^2$$

and that is the least square error problem with some inference on the noise.



Part-c

1c) Write down the expression of the joint prior distribution of the parameters at stake and illustrate some suitable choice for the hyperparameters.

-Answer

And we can suppose that our priors distributions are a type of regularization for our estimate paramaters.

So, considering the indipendence of the parameters the prior distribution can be written by a product of the distributions:

$$\pi(\alpha, \beta, \gamma, \tau^2) = \pi_\alpha(\alpha) \cdot \pi_\beta(\beta) \cdot \pi_\gamma(\gamma) \cdot \pi_{\tau^2}(\tau^2)$$

$$\mathbb{1}_{[1,\infty]}(\alpha)\frac{1}{\sigma_\alpha\sqrt{2\pi}}e^{-\frac{\alpha^2}{2\sigma_\alpha}} \cdot \mathbb{1}_{[1,\infty]}(\beta)\frac{1}{\sigma_\beta\sqrt{2\pi}}e^{-\frac{\beta^2}{2\sigma_\beta}} \cdot \mathbb{1}_{[0,1]}(\gamma)\frac{1}{1-0} \cdot \mathbb{1}_{[0,\infty]}(\tau^2)\frac{b^a}{\Gamma(a)}\tau^{2(-\alpha-1)}e^{-\frac{\beta}{\tau^2}}$$

So looking for each prior:

- α is the maximum value that the Lenght can reach when the age is infinity and giving to it a normal distribution with $\mathbb{E}(\alpha) = 0$ and $\mathbb{V}(\alpha) = \sigma_\alpha$, it means for small values of the variance penalize more distance from 0, so to push this value close to 0.(I think should be interesting also play with the mean of this value such that it's possible to push up or down the regression function)
- β is the coefficient of the pow so playing with its means have a bigger or smaller growth, but it's important also to define the intercept at the origin since when $x = 0$: $y = \alpha - \beta \cdot 1$ And we can think at it like we did with *alpha*.
- γ define the speed of how the length converge to the maximum value α. So an appropriate prior is to give it equal probability for each value.
- τ^2 rappresent's the variance of our error, so we could push it to be small or less small using the hyperparameters of an inverse-gamma distribution.

Part-d

1d) Compute numerically the maximum likelihood estimate for the vector of parameters of interest $(\alpha, \beta, \gamma, \tau)$ and compare it with the Maximum-a-Posterori estimate.

-Answer

```
dat <- read.table('Dugong.txt', header = T)
Likelihood_3 <- function(vector){
  Lik = 0
  alpha=vector[1]
  beta = vector[2]
  gamma = vector[3]
  tau.square = vector[4]
  if(((alpha>1)*(beta>1)*(gamma>0 && gamma<1)
      *(tau.square>0))==0){
    Lik = 1e+10
    return(Lik)
  }
  for( i in 1:nrow(dat)){
    mu = (alpha-beta* (gamma^(dat$Age[i])))
    if(mu <0){
      Lik = 1e+10
      return(Lik)
    }
    l = log(tau.square^(-0.5)) - ((dat$Length[i] - mu)^2 / (2*tau.square))
    Lik = (sum(Lik, l))
  }
  return(-Lik)
}

MLE <- optim(c(1.1, 1.1, 0.2, 3),Likelihood_3)


cat('MLE alpha =',MLE$par[1],'\n',
    'MLE beta =',MLE$par[2],'\n',
    'MLE gamma =',MLE$par[3],'\n',
    'MLE tau.square =',MLE$par[4],'\n',
    '- log(Likelihood) optim =',Likelihood_3(MLE$par),'\n',
    'initial value - log(Likelihood) = ',Likelihood_3(c(1.1, 1.1, 0.1, 3)))
```

```
## MLE alpha = 3.360694
##  MLE beta = 1.472192
##  MLE gamma = 0.965209
##  MLE tau.square = 0.01523837
##  - log(Likelihood) optim = -43.62098
##  initial value - log(Likelihood) =  22.07852
```

```
i=1
MLE <- optim(c(1.1, 1.1, 0.2, 3),Likelihood_3)
res <- matrix(c(i,MLE$par,Likelihood_3(MLE$par)), ncol = 6)

for(a in 1:3){
  for(b in 1:3){
    for(g in 1:3){
      for(t in 1:3){
        i=i+1
        cat(i,' ',Likelihood_3(c(1.1+(a-1)*1, 1.1+(b-1)*1, 0.2+(g-1)*0.1, 3+(t-1)*1)),'\n')
        cat(c(1.1+(a-1)*1, 1.1+(b-1)*1, 0.2+(g-1)*0.1, 3+(t-1)*1),'\n')
        MLE <- optim(c(1.1+(a-1)*1, 1.1+(b-1)*1, 0.2+(g-1)*0.1, 3+(t-1)*1),Likelihood_3)
        res <- rbind(res,c(i, MLE$par, Likelihood_3(MLE$par)))
      }
    }
  }
}
```

```
opt = unlist(res[res[,6]==min(res[,6]),][2:6])
cat('MLE search alpha =',opt[1],'\n',
    'MLE search beta =',opt[2],'\n',
    'MLE search gamma =',opt[3],'\n',
    'MLE search tau.square =',opt[4],'\n',
    '- log(Likelihood) optim =',Likelihood_3(opt[1:4]),'\n')
```

```
## MLE search alpha = 2.677312
##  MLE search beta = 1.000025
##  MLE search gamma = 0.874115
##  MLE search tau.square = 0.00698923
##  - log(Likelihood) optim = -53.54471
```

```
Prior <- function(vector){
  alpha=vector[1]
  beta = vector[2]
  gamma = vector[3]
  tau.square = vector[4]
  alpha.p <- dnorm(alpha,0,1e+3)*(alpha>=1)
  beta.p <- dnorm(beta,0,1e+3)*(beta>=1)
  gamma.p <- dunif(gamma,0,1)
  tau.square.p <- dinvgamma(tau.square,1,1)*(tau.square>=0)
  join <- log(alpha.p + 1e-10) + log(beta.p + 1e-10) +
          log(gamma.p + 1e-10) + log(tau.square.p + 1e-10)
  return(-join)
}

Posterior <- function(vector){Prior(vector)+Likelihood_3(vector)}

MPO <- optim(c(1.5, 1.5, 0.02 , 3),Posterior)
MPO$par
```

```
## [1] 2.84342847 1.04450786 0.92247431 0.06504345
```

```
cat('Maximum a Posterior alpha.hat =',MPO$par[1],'\n',
    'Maximum a Posterior beta.hat =',MPO$par[2],'\n',
    'Maximum a Posterior gamma =',MPO$par[3],'\n',
    'Maximum a Posterior tau.square =',MPO$par[4],'\n',
    ' -log(Posterior) local optim =',Posterior(MPO$par),'\n',
    ' -log(Posterior) initial value =',Posterior(c(1.1, 1.1, 0.2, 3)))
```

```
## Maximum a Posterior alpha.hat = 2.843428
##  Maximum a Posterior beta.hat = 1.044508
##  Maximum a Posterior gamma = 0.9224743
##  Maximum a Posterior tau.square = 0.06504345
##   -log(Posterior) local optim = -9.543952
##   -log(Posterior) initial value = 40.35078
```

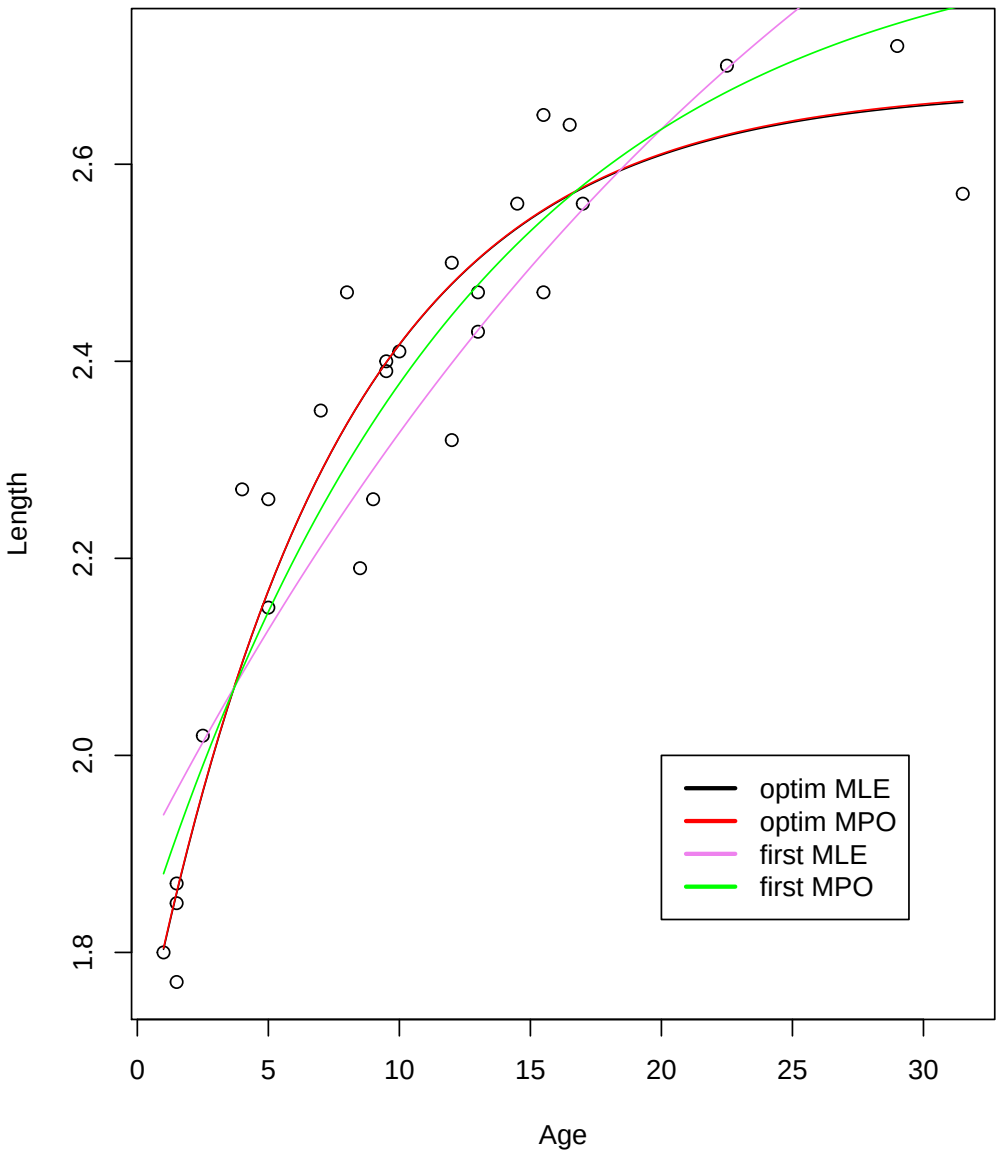
```
i=1
MPO <- optim(c(1.1, 1.1, 0.2, 3),Posterior)
res_mpo <- matrix(c(i,MPO$par,Posterior(MPO$par)), ncol = 6)

for(a in 1:3){
  for(b in 1:3){
    for(g in 1:3){
      for(t in 1:3){
        i=i+1
        cat(i,' ',Posterior(c(1.1+(a-1)*1, 1.1+(b-1)*1, 0.2+(g-1)*0.1, 3+(t-1)*1)),'\n')
        cat(c(1.1+(a-1)*1, 1.1+(b-1)*1, 0.2+(g-1)*0.1, 3+(t-1)*1),'\n')
        MPO <- optim(c(1.1+(a-1)*1, 1.1+(b-1)*1, 0.2+(g-1)*0.1, 3+(t-1)*1),Posterior)
        res_mpo <- rbind(res_mpo,c(i, MPO$par, Posterior(MPO$par)))
      }
    }
  }
}
```

```
opt_mpo = unlist(res_mpo[res_mpo[,6]==min(res_mpo[,6]),][2:6])
cat('MPO search alpha =',opt[1],'\n',
```

```
    'MPO search beta =',opt[2],'\n',
    'MPO search gamma =',opt[3],'\n',
    'MPO search tau.square =',opt[4],'\n',
    '- log(Posterior) optim =',(opt[1:4]),'\n')
```

```
## MPO search alpha = 2.677312
##  MPO search beta = 1.000025
##  MPO search gamma = 0.874115
##  MPO search tau.square = 0.00698923
##  - log(Posterior) optim = 2.677312 1.000025 0.874115 0.00698923
```

```
mu.func <- function(x, vector= vector){
  alpha=vector[1]
  beta = vector[2]
  gamma = vector[3]
  tau.square = vector[4]
  mu = (alpha-beta* (gamma^(x)))
  return(mu)
}

plot(dat$Age,dat$Length, col='black',xlab='Age',ylab='Length')
curve(mu.func(x, vector = opt[1:4]), add = TRUE, col='black')
curve(mu.func(x, vector = opt_mpo[1:4]), col = 'red', add= TRUE)
curve(mu.func(x, vector = MLE$par), add = TRUE, col='violet')
curve(mu.func(x, vector = MPO$par), col = 'green', add= TRUE)
legend(20,2, c("optim MLE","optim MPO","first MLE","first MPO"), lty = c(1,1,1,1),
       lwd = c(2.5,2.5,2.5,2.5), col = c('black',"red","violet","green"))
```



And we can clearly see that both likelihood than posterior gives us the same result after some better optimization.

Question 2

2) Consider the Acceptance-Rejection algorithm in the most general form and denote with $\theta = Y^A$ the random variable obtained with the algorithm

Part-a

2a) Determine the analytic expression of the acceptance probability

- Answer

We want to derive the probabilty of $E = 1$, and we now that E is a r.v. distribuited like a *Bernoulli* with probabilty $p = \frac{f(y)}{kg(y)}$.

Where $g_y(y)$ is our *instrumental density distribution*, while $f_x(X)$ is our *target density*, such that $X \subseteq Y$, while k is choose to make the coinstrant $0 \leq \frac{f(y)}{kg(y)} \leq 1$ so that it can be considered a probability.

$$P(E = 1) = \int J(E = 1, Y = y)\, dy = \int P(E = 1|Y = y) \cdot P(Y = y)\, dy = \int \frac{f(y)}{kg(y)} \cdot g(y)\, dy$$

$$P(E = 1) = \frac{1}{k} \int f(y) dy = \frac{1}{k}$$

Part-b

2b) Prove that θ has the desired target distribution

-Answer

Since $\theta = Y^A$ we can write his probability:

$$P(\theta) = P(Y = y|E = 1) = \frac{P(E = 1|Y = y) \cdot P(Y = y)}{P(E = 1)} = \frac{\frac{f(y)}{kg(y)} \cdot g(y)}{\frac{1}{k}} = \begin{cases} f(y) & if \quad y \in X \\ 0 & otherwise \end{cases}$$

Where the last equality is true since $f(y) = 0$ if $y \in Y - X$.

Part-c

2c) Show how in Bayesian inference you could use simulations from the prior (auxiliary density) to get a random draw from the posterior (target distribution) without knowing the proportionality constant

-Answer

Suppose we have a function $f(x)$ from which we wish to simulate, but this function is not normalized, and his integral is difficult to compute analitically: $\int f(x)\,dx = K$, and we want to use as instrumental distribution $\tilde{q}(y)$ with $x \in X, \quad y \in Y, \quad X \subseteq Y$.

(*Where* $K = \int \frac{f(\theta)}{\tilde{q}(\theta)} \cdot \tilde{q}(\theta)\,d\theta$ *could be computed using MC.*)

We can simulate from it, thinking in a Bayesian Inference setup.

Using:

$$\begin{cases} \pi(\theta|E) = \frac{f(\theta)}{K} \\ \pi(\theta) = \tilde{q}(\theta) \\ L_{E|\theta}(\theta) = Ber(\frac{f(\theta)}{M \cdot \tilde{q}(\theta)}) \end{cases}$$

Where $\frac{f(\theta)}{M \cdot q(\theta)} \leq 1$ and $L() \propto \frac{\pi(\theta|E)}{\pi(\theta)} \leq M \cdot K$

So a problem could be to see if the Likelihood function is bounded or not... and this is not always true!

Then if the Likelihood is bounded, we want to find an optimal value for M since taking big values for it mean have a low probability of acceptance.

So dipending on the function $f(x)$ that we want to simulate the problem is to choose an appropriate prior distribution such that M is the minimum value possible, and for this reason exist methods like "Adaptive Rejection Sampling" that using convexity properties of the target distribution, can reduce a lot the probabilty of acceptance.

$$\pi(\theta|E) = \frac{L(E|\theta) \cdot \pi(\theta)}{\int L(E|\theta) \cdot \pi(\theta)\,d\theta}$$

So the algorithm is:

1) Simulate from the auxiliar distribution $\pi(\theta)$.
2) Then we accept if $E = 1|Y = y \sim Bernoulli(p = \frac{1}{M} \cdot \frac{f(\theta)}{\tilde{q}(\theta)})$

Where:

$$P(E = 1) = P(E = 1|Y = y) \cdot P(Y = y) \int \frac{1}{M} \cdot \frac{f(\theta)}{\tilde{q}(\theta)} \cdot \pi(\theta)\,dy = \frac{1}{M} \cdot K$$

Example of how use Monte Carlo to compute K:

Let's use as auxiliar distribution a $Beta(1,1)$ to simuate r.v. from a function $f(x) = x^3(1-x)^6 \quad whit \quad x \in [0,1]$

where:

$$\int_0^1 x^3(1-x)^6 dx = B(4,7) = \frac{\Gamma(4) \cdot \Gamma(7)}{\Gamma(4+7)} = \frac{3! \cdot 6!}{10!} = \frac{6}{5040} = 0,00119$$

In this case is possible to compute the integral easy, but in other cases could be not so easy, but with montecarlo we approsimate it:

```r
dtarget<- function(x) x^3*(1-x)^6
a = 1
b = 1
dauxiliary <- function(x) dbeta(x,a,b)
rauxiliary <- function(n) rbeta(n,a,b)

samp = runif(1e+5)
k= mean((dtarget(samp)/dauxiliary(samp)))
k
```

```
## [1] 0.001196045
```

Part-d

2d) Illustrate analytically possible difficulties of this approach with a simple conjugate model

-Answer

The main problem of this approach is to find a good estimator of M that bound the Likelihood function, but not too big because for great values of M we have a low probability of acceptance.

The natural choose for this number is of course the maximum value of $\frac{f(y)}{\tilde{q}(y)}$, but this number could be difficult to find analytically.

$$sup \quad \frac{f(y)}{\tilde{q}(y)}$$

For the sake of clarity let's play with some conjugate model; let's try to produce some random variable from a beta distribution with parameters $\mathbb{B}(4,7)$, sampling from a beta $\mathbb{B}(1,1)$ that is a uniform distribution.

So, we have to find:

$$sup \quad \frac{f(y)}{\tilde{q}(y)} = sup \quad \frac{\frac{x^{4-1}(1-x)^{7-1}}{\mathbb{B}(4,7)}}{1}$$

and:

$$\frac{d}{dy}\frac{f(y)}{\tilde{q}(y)} = 3x^2(1-x)^6 - 6x^3(1-x)^5 = 0$$

that has the maximum in $\frac{3}{9}$, but not evry time it's possible to find this maximum if the function it's not bounded.

Another issues is that the probabilty of Acceptance decrease if we have too many observations, because K go down faster than how M decrease.

Remember that $P(E=1) = \frac{K}{M}$

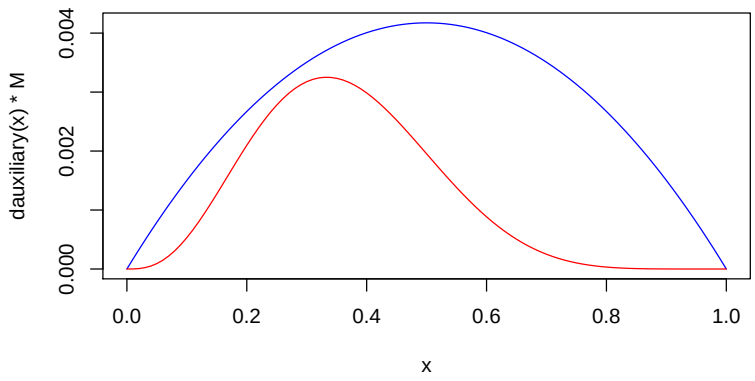
Part-e

2e) Verify your conclusions implementing the Acceptance-Rejection approach with your conjugate model (verify empirically that θ has the desired target distribution $\pi(\theta|x_1, .., x_n)$)

-Answer

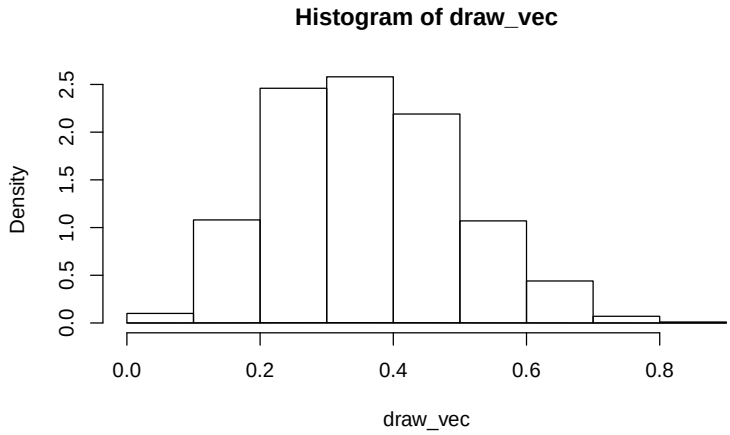
Let's use as auxiliar distribution a $Beta(5, 8)$ to simuate r.v. from a function $f(x) = x^3(1-x)^6 \quad whit \quad x \in [0, 1]$

```
dtarget<- function(x) x^3*(1-x)^6
a = 2
b = 2
dauxiliary <- function(x) dbeta(x,a,b)
rauxiliary <- function(n) rbeta(n,a,b)

samp = runif(1e+5)
M = max((dtarget(samp)/dauxiliary(samp)))
k= mean((dtarget(samp)/dauxiliary(samp)))
M = M*(1.1)

AR=function(dtarget,dauxiliary,rauxiliary,M){
  count=0
  E=0
  while(E==0){
    candidate = rauxiliary(1)
    acc_prob= (dtarget(candidate)/(M*dauxiliary(candidate)))
    E = sample(c(1,0),prob=c(acc_prob, 1-acc_prob),size=1)
    count=count+1
  }
  return(list(draw=candidate,computational_effort=count))
}

curve(dauxiliary(x)*M, col ='blue')
curve(dtarget, col= 'red',add=T)
```

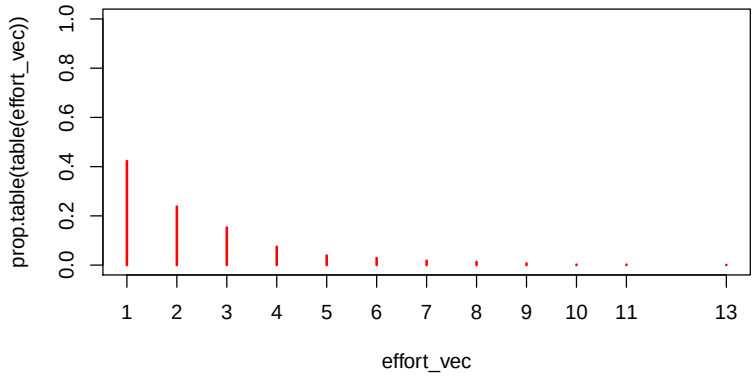



```
mcsize=1000
draw_vec=rep(NA,mcsize)
```

```r
effort_vec=rep(NA,mcsize)

for(i in 1:mcsize){

  DD=AR(dtarget,dauxiliary,rauxiliary,M)
  draw_vec[i] = DD$draw
  effort_vec[i] = DD$computational_effort

}

hist(draw_vec,freq=FALSE)
curve(dtarget(x),add=TRUE)
```




```r
plot(prop.table(table(effort_vec)),ylim=c(0,1),pch=16,col="red")
```



Question 3

3) Simulate from a standard Normal distribution using pseudo-random deviates from a standard Cauchy and the A-R algorithm. Write the expression the corresponding acceptance probabilityof and evaluate it numerically by MC approximation.

-Answer

To use pseudo-random deviates to sample from a standad Cauchy, we should use the fact that the Cumulative Density Function is invertible compute the inverse of the Cumulative Density Function to make

$P(E=1) = \int P(e=1|Y=y) \cdot P(Y=y)\, dy = \int \frac{f(y)}{M \cdot g(y)} g(y)\, dy = \frac{1}{M} \int f(y)\, dy$

where M is the minum M such that $\frac{f(y)}{M \cdot g(y)} \leq 1 \ \forall y \in Y$.

So numerically we should find:

$$sup\ \frac{f(y)}{g(y)} = sup\ \frac{\frac{1}{\sqrt{2\pi}} \cdot e^{-\frac{x^2}{2}}}{\frac{1}{\pi \cdot (1+x^2)}} = sup\ \frac{\pi \cdot (1+x^2)}{\sqrt{2\pi}} \cdot e^{-\frac{x^2}{2}}$$

so:

$$\frac{d}{dy} \cdot \frac{f(y)}{g(y)} = 2x \frac{\pi}{\sqrt{2\pi}} \cdot e^{-\frac{x^2}{2}} - x \frac{\pi \cdot (1+x^2)}{\sqrt{2\pi}} \cdot e^{-\frac{x^2}{2}} = e^{-\frac{x^2}{2}} \cdot \frac{\pi}{\sqrt{2\pi}} \cdot (2x - x \cdot (1+x^2))$$

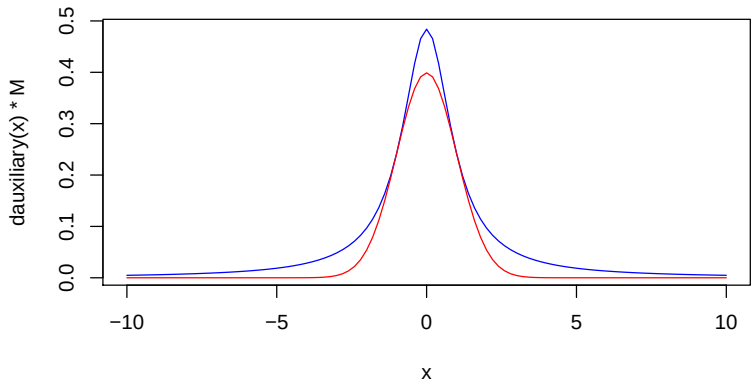
that is equal to zero and mazimum in $x = \pm 1$, and so $M = \sqrt{\frac{2\pi}{e}}$ and of course:

$$P(E=1) = \frac{1}{M} = \sqrt{\frac{e}{2\pi}} = 0.66$$

So numerically let's see:

```
dtarget<- function(x) dnorm(x)

dauxiliary <- function(x) dcauchy(x)
rauxiliary <- function(n) rcauchy(n)

samp = rauxiliary(1e+5)
M = max((dtarget(samp)/dauxiliary(samp)))

M = M*(1.0001)
AR=function(dtarget,dauxiliary,rauxiliary,M){
  count=0
  E=0
  while(E==0){
    candidate = rauxiliary(1)
    acc_prob= (dtarget(candidate)/(M*dauxiliary(candidate)))
    E = sample(c(1,0),prob=c(acc_prob, 1-acc_prob),size=1)
    count=count+1
  }
  return(list(draw=candidate,computational_effort=count))
}
```

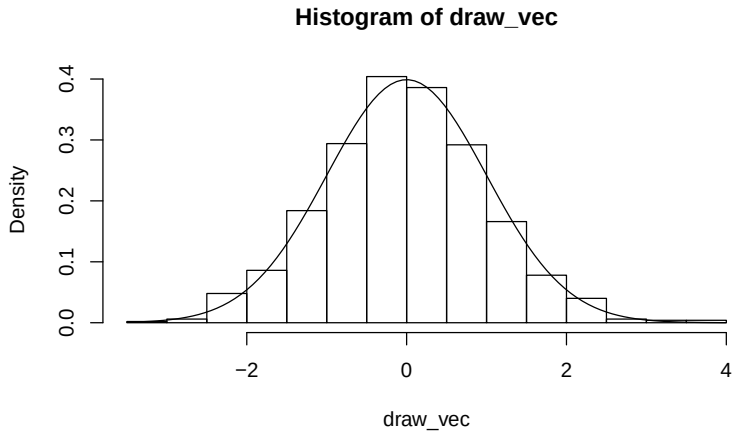
```
curve(dauxiliary(x)*M, col ='blue', from = -10, to = 10)
curve(dtarget, col= 'red',add=T)
```



where M is:

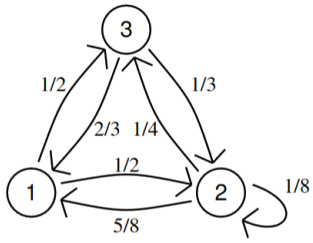
```
##          M      P(E=1)
##  1.5204989  0.6576789
```

```
mcsize=1000
draw_vec=rep(NA,mcsize)
effort_vec=rep(NA,mcsize)

for(i in 1:mcsize){

  DD=AR(dtarget,dauxiliary,rauxiliary,M)
  draw_vec[i] = DD$draw
  effort_vec[i] = DD$computational_effort

}

hist(draw_vec,freq=FALSE)
curve(dtarget(x),add=TRUE)
```



Question 4

4) Let us consider a Markov chain $(X_t)_{t\geq 0}$ defined on the state space $\mathcal{S} = \{1, 2, 3\}$ with the following transition



Part-a and -b

4a) Starting at time $t = 0$ in the state $X_0 = 1$ simulate the Markov chain with distribution assigned as above for $t = 1000$ consecutive times

b) compute the empirical relative frequency of the three states in your simulation

-Answer

```
states <- c(1,2,3)
transition_matrix <- matrix(data= c(0,1/2,1/2,
                                    5/8,1/8,1/4,
                                    2/3,1/3,0), byrow = TRUE ,nrow=3)




t0 <- 1

nsample<-1000
chain<-rep(NA,nsample+1) # vector that will hold
# all the simulated values
chain[1]<-t0              # starting value x1 assigned to chain[1]
for(t in 1:nsample){
  chain[t+1]<-sample(states,size=1,prob=transition_matrix[chain[t],])
}

prop.table(table(chain))
```

```
## chain
##         1         2         3
## 0.3976024 0.3166833 0.2857143
```

Part-c

4c) repeat the simulation for 500 times and record only the final state at time $t = 1000$ for each of the 500 simulated chains. Compute the relative frequency of the 500 final states. What distribution are you approximating in this way?
Try to formalize the difference between this point and the previous point.

-Answer

```
t0 <- 1

ntimes <- 500
nsample<-1000
t1000 <-rep(NA,ntimes)

for(i in 1:ntimes){
  chain<-rep(NA,nsample+1) # vector that will hold
  chain[1]<-t0             # starting value x1 assigned to chain[1]
  for(t in 1:nsample){
    chain[t+1]<-sample(states,size=1,prob=transition_matrix[chain[t],])
  }
  t1000[i]<- chain[nsample+1]
}

prop.table(table(t1000))
```

```
## t1000
##     1     2     3
## 0.408 0.322 0.270
```

In this last case we're trying to approximate the distribution of the invariant distribution using only the kernel matrix $K^{1000}(x_0 = 1, \Omega)$.

While in the preview simulation we were approximating the same invariant distribution, but using a random walk given by the simulation that start from x_0, that is given by the chain originated by the transition kernel s.t. :

$$K(x_{t-1}, x_t) \quad \forall\, t = 1, 2..., 1000$$

This first case have the issues of take into account the initial point and the correlation on the preview state in time, so the variance of his estimate his bigger in respect of the second estimate, but instead the first one consume less computation than the second.

Part-d

4d) compute the theoretical stationary distribution π and explain how you have obtained it

-Answer

In theory the stationary distribution it can be computed looking for the eigen vector related to the eigen value equal to 1. But in R the function that gives us this this vector doesn't work so well:

```
eigen(transition_matrix)
```

```
## $values
## [1]  1.0000000 -0.6509781 -0.2240219
##
## $vectors
##           [,1]       [,2]       [,3]
## [1,] 0.5773503  0.7330694  0.1410902
## [2,] 0.5773503 -0.4174362 -0.7309268
## [3,] 0.5773503 -0.5369882  0.6677122
```

```
eigen(transition_matrix)$vectors[1,]%*%transition_matrix
```

```
##           [,1]      [,2]      [,3]
## [1,] 0.5522285 0.4273389 0.4719425
```

```
eigen(transition_matrix)$vectors[1,]
```

```
## [1] 0.5773503 0.7330694 0.1410902
```

So it's possible to computed it using an high power of the transition matrix:

(*I used markovchain library*)

```
(Mc_states^1000)[1,]
```

```
##         1         2         3
## 0.3917526 0.3298969 0.2783505
```

Part-e

4e) is it well approximated by the simulated empirical relative frequencies computed in (b) and (c)?

-Answer

The simulation computed in b) it's not good such the one computed in c), because like I said before the variance of the estimate in b) is grater than the one in c).

But despite everything are not bad both.

Part-f

4f) what happens if we start at $t = 0$ from state $X_0 = 2$ instead of $X_0 = 1$?

-Answer

```
states <- c(1,2,3)
transition_matrix <- matrix(data= c(0,1/2,1/2,
                                    5/8,1/8,1/4,
                                    2/3,1/3,0), byrow = TRUE ,nrow=3)




t0 <- 2

nsample<-1000
chain1<-rep(NA,nsample+1) # vector that will hold
# all the simulated values
chain1[1]<-t0             # starting value x1 assigned to chain[1]
for(t in 1:nsample){
  chain1[t+1]<-sample(states,size=1,prob=transition_matrix[chain1[t],])
}

prop.table(table(chain))
```

```
## chain
##         1         2         3
## 0.3866134 0.3266733 0.2867133
```

```
ntimes <- 500
nsample<-1000
t1000 <-rep(NA,ntimes)

for(i in 1:ntimes){
  chain2<-rep(NA,nsample+1) # vector that will hold
  chain2[1]<-t0             # starting value x1 assigned to chain[1]
  for(t in 1:nsample){
    chain2[t+1]<-sample(states,size=1,prob=transition_matrix[chain2[t],])
  }
  t1000[i]<- chain2[nsample+1]
}

prop.table(table(t1000))
```

```
## t1000
##     1     2     3
## 0.414 0.292 0.294
```

How we said before this approxiamtion dipends on the initial state, so our estimate stationary distribution change from the previouse ones.

Question-5

5) Consider again the Bayesian model for Dugong's data:

Part-a and -b

5a) Derive the functional form (up to proportionality constants) of all *full-conditionals* b) Which distribution can you recognize within standard parametric families so that direct simulation from full conditional can be easily implemented ?

-Answer

To compute the full-conditionals PDF let's compute before the join posterior distributions:

$$\pi(\alpha,\beta,\gamma,\tau^2|(Y|X)) \propto (\tau^2)^{-\frac{n}{2}} exp\Big\{ - \frac{n\alpha^2 + \sum_i y_i^2 + \sum_i \beta^2\gamma^{2x_i} - 2\alpha(\sum_i(y_i + \beta\gamma^{x_i})) + 2\sum_i(y_i\beta\gamma^{x_i})}{2\tau^2}\Big\}\cdot$$

$$\cdot exp\Big\{ - \frac{\alpha^2}{2\sigma_\alpha^2} - \frac{\beta^2}{2\sigma_\beta^2} - \frac{b}{\tau^2}\Big\} \cdot \tau^{2(-a-1)} \qquad \forall\alpha \in (1;\infty), \beta \in (1;\infty), \gamma \in (0,1), \tau^2 \in (0;\infty)$$

So to try to implement the Gibbs sampling, we have to derive analitically the distribution probability functions of each full-conditionals posteriors.

So for α:

$$\pi(\alpha|\beta,\gamma,\tau^2,Y|X) \propto exp\Big\{ - \frac{n}{2\tau^2}\cdot\alpha^2 + \frac{\sum_i(y_i + \beta\cdot\gamma^{x_i})}{\tau^2}\alpha - \frac{1}{2\sigma_\alpha^2}\alpha^2\Big\}\mathbb{1}_{(0,\infty)}(\alpha)$$

$$\alpha' \sim N_{\alpha'}\left(a = \frac{\tau^2 + n\sigma_\alpha^2}{\tau^2\cdot\sigma_\alpha^2}; b = \frac{\sum_i(y_i + \beta\cdot\gamma^{x_i})}{\tau^2}\right) \qquad \forall\alpha' \in (1,\infty)$$

So for β:

$$\pi(\beta|\alpha,\gamma,\tau^2,Y|X) \propto exp\Big\{ - \frac{\sum_i\gamma^{2x_i}}{2\tau^2}\cdot\beta^2 + \frac{(\sum_i(\alpha - y_i)\gamma^{x_i})}{\tau^2}\beta - \frac{1}{2\sigma_\beta^2}\beta^2\Big\}\mathbb{1}_{(0,\infty)}$$

$$\beta' \sim N_{\beta'}\left(a = \frac{\sigma_\beta^2\sum_i\gamma^{2x_i} + \tau^2}{\tau^2\cdot\sigma_\beta^2}; b = \frac{(\sum_i(\alpha - y_i)\gamma^{x_i})}{\tau^2}\right) \qquad \forall\beta' \in (1;\infty)$$

for γ:

$$\pi(\gamma|\alpha,\beta,\tau^2,Y|X) \propto exp\Big\{ - \frac{\sum_i[\beta\gamma^{x_i}(\beta\gamma^{x_i} - 2\alpha + 2y_i)]}{2\tau^2}\Big\} \qquad \gamma \in (0;1)$$

for τ^2:

$$\pi(\tau^2|\gamma,\alpha,\beta,Y|X) \propto (\tau^2)^{-\frac{n}{2}} \cdot exp\Big\{ - \frac{\sum_i (y_i - \alpha + \beta\gamma^{x_i})^2}{2\tau^2} \Big\} \cdot exp\Big\{ - \frac{b}{\tau^2} \Big\} \cdot \tau^{2(-a-1)}$$

$$\propto exp\Big\{ - \frac{\frac{1}{2}\sum_i (y_i - \alpha + \beta\gamma^{x_i})^2 + b}{\tau^2} \Big\} \cdot \tau^{2(-a-\frac{n}{2}-1)}$$

$$\tau^2_, \sim IG(\alpha = a + \frac{n}{2} \; ; \; \beta = \frac{1}{2}\sum_i (y_i - \alpha + \beta\gamma^{x_i})^2 + b) \qquad \forall \tau^2 \in (0;\infty)$$

So we have two normal distributions, one inverse gamma distribution and something that it's a distribution of probability because it's bounded and strictly positive, but it's not a distribution from standards families.

Part-c

5c) Using a suitable Metropolis-within-Gibbs algorithm simulate a Markov chain ($T = 10000$) to approximate the posterior distribution for the above model

-Answer

```
library(truncnorm)

sigma_alpha <- 1000

full.conditional.alpha <- function(start){
  al = (1/(sigma_alpha^2)) + (27/start['tau.square'])
  be = ((start['beta']*(sum(start['gamma']^dat$Age)))   +
         sum(dat$Length))/ start['tau.square']
  sim_alpha <- 0
  while(sim_alpha<=1){
    sim_alpha <- rtruncnorm(1,mean = be/al , sd = sqrt(1/al), a =1)
  }
  return(sim_alpha)
}

sigma_beta <- 1000

full.conditional.beta <- function(start){
  al = ((sum(start['gamma']^(2*dat$Age))/start['tau.square']) + (1/(sigma_beta^2)))
  be = (start['alpha']*sum(start['gamma']^dat$Age)  -
         sum(dat$Length * (start['gamma']^dat$Age))) / start['tau.square']
  sim_beta <- 0
  while(sim_beta<=1){
    sim_beta <- rtruncnorm(1,mean = be/al , sd = sqrt(1/al), a=1)
  }
  return(sim_beta)
}


gamma.function.ugly.distribution <- function(gam, start){
  alp.n = start['alpha']
  bet.n = start['beta']
  tau.2.n = start['tau.square']
  mu.n = alp.n - bet.n*(gam^dat$Age)
  ret <- exp(- (1/tau.2.n)*0.5*sum((dat$Length - mu.n)^2))
  return(ret)
}


full.conditional.gamma.1 <- function(start){
  gamma.new <- runif(1)
  check_MH <- runif(1)
  gamm <- start['gamma']
  prob <- min(((gamma.function.ugly.distribution(gamma.new, start))/(gamma.function.ugly.distribution(g
  check_MH <- rbinom(1,1,prob = prob)
  if(check_MH){
    gamm <- gamma.new
```

```
  }
  return(gamm)
}

al <- 3
be <- 0.5

full.conditional.tau.square <- function(start){
  alpha.n <- start['alpha']
  beta.n <- start['beta']
  gamma.n <- start['gamma']
  a = al + 27/2
  b = ((1/2)*sum((dat$Length - alpha.n + beta.n*(gamma.n^dat$Age))^2)) + be
  tau.square.sim <- rinvgamma(1,a,b)
  return(tau.square.sim)
}


start<- c('alpha'=1.1, 'beta'=1.01, 'gamma'=0.87,'tau.square'=0.007)

n=10000
simulation1 <- matrix(nrow = n, ncol= 4)
for(i in 1:n){
  start['alpha']= full.conditional.alpha(start)
  start['beta']= full.conditional.beta(start)
  start['gamma'] = full.conditional.gamma.1(start)
  start['tau.square'] = full.conditional.tau.square(start)
  simulation1[i,]=start
}

m <- mcmc(simulation1)
AcceptanceRate(m)
```

```
AcceptanceRate(simulation1)
```

```
## [1] 1.00000000 1.00000000 0.06340634 1.00000000
```

Part d, e & f

5d) Show the 4 univariate trace-plots of the simulations of each parameter

e) Evaluate graphically the behaviour of the empirical averages $\hat{I}_t$ with growing $t = 1, ..., T$

f) Provide estimates for each parameter together with the approximation error and explain how you have evaluated such error

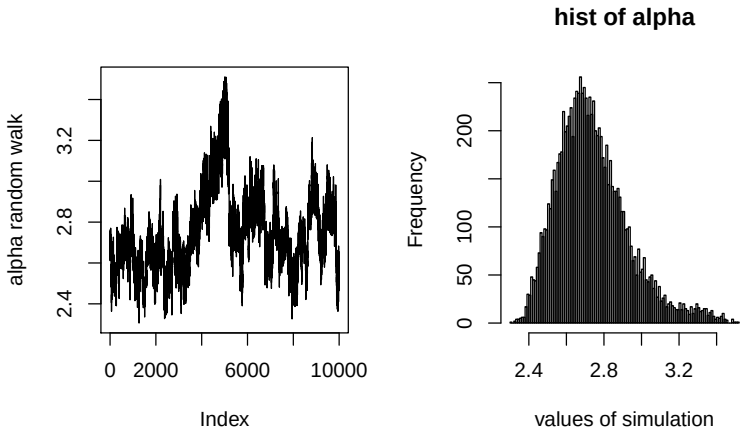
-Answer

for α, the simulation gives me this result:

```
summary(simulation1[,1])
```

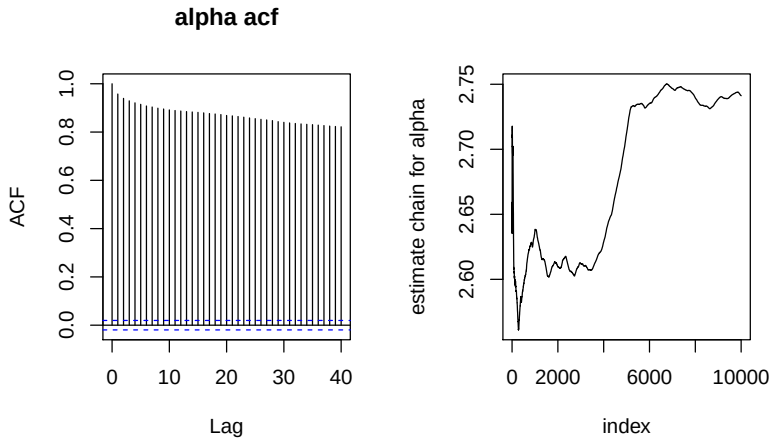
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.306   2.606   2.714   2.741   2.845   3.511
```

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

plot(simulation1[,1], type = 'l', ylab='alpha random walk')
hist(simulation1[,1], breaks = 150, main = 'hist of alpha', xlab = 'values of simulation' )
```



And the graphically behaviour of the empirical averages, is:

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

acf(simulation1[,1], main ='alpha acf')
plot(cumsum(simulation1[,1])/seq_along(simulation1[,1]), type='l',
     xlab ='index' ,ylab='estimate chain for alpha')
```



where:

$$\hat{I} = \frac{1}{t}\sum^{t}\theta_t \quad \forall\, \theta_t \sim \pi(\theta)$$

$$\mathbb{E}(\theta) = \int \theta\pi(\theta)\, d\theta \;\approx \frac{1}{t}\sum^{t}\theta_t \quad \forall\, \theta_t \sim \pi(\theta)$$

and the variance of the estimation can be computed using 'Bacth Means':

$$\mathbb{V}(\hat{I}) = B\frac{1}{\frac{t}{B}-1}\sum_{b=1}^{B}(\hat{I}_{(b)} - \hat{I}_t)^2$$

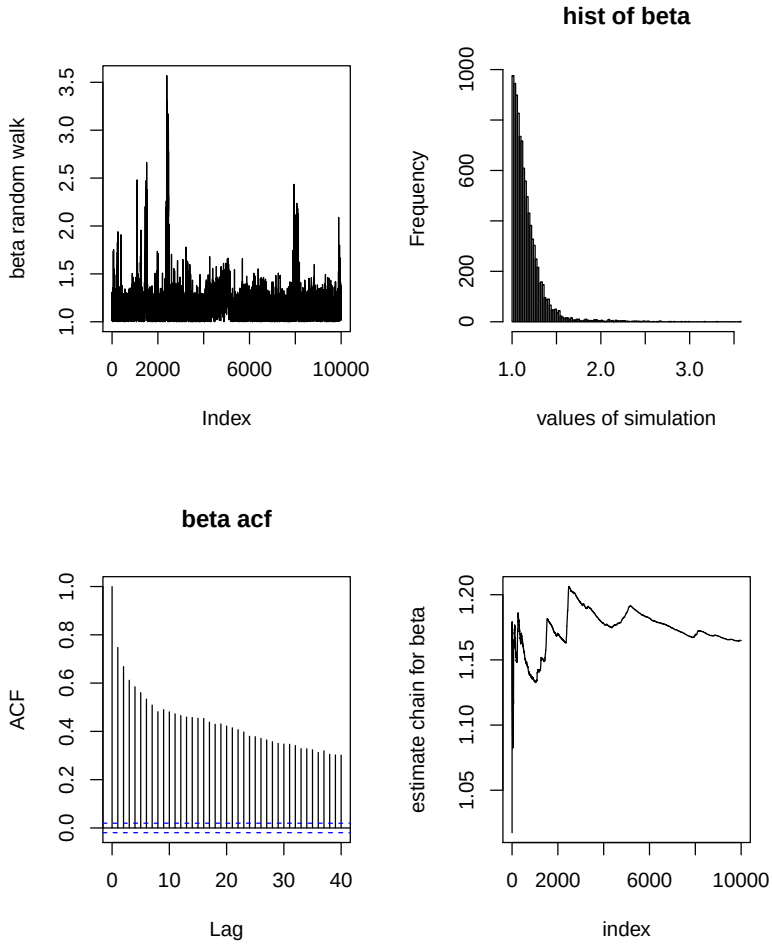
And i used 'mcmcse' packet to compute the standard error (computed using batch means with size equal to $\sqrt{10000} = 100$):

```
c("mean estimator for alpha" = mean(simulation1[,1]),"standard error"= mcmcse::mcse(simulation1[,1])$se
```

```
## mean estimator for alpha           standard error
##               2.74117549               0.01779908
```

for β, I have this results:

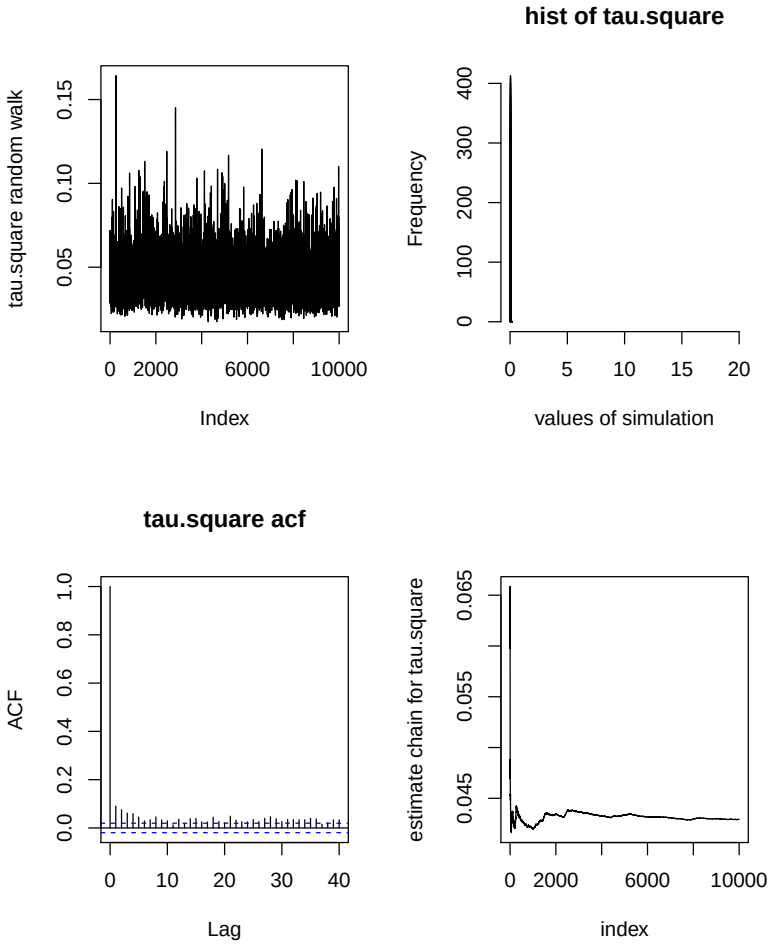
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.053   1.118   1.165   1.216   3.570
```




```
## mean estimator for beta          standard error
##              1.16491041              0.01059794
```

for τ^2, I have this results:

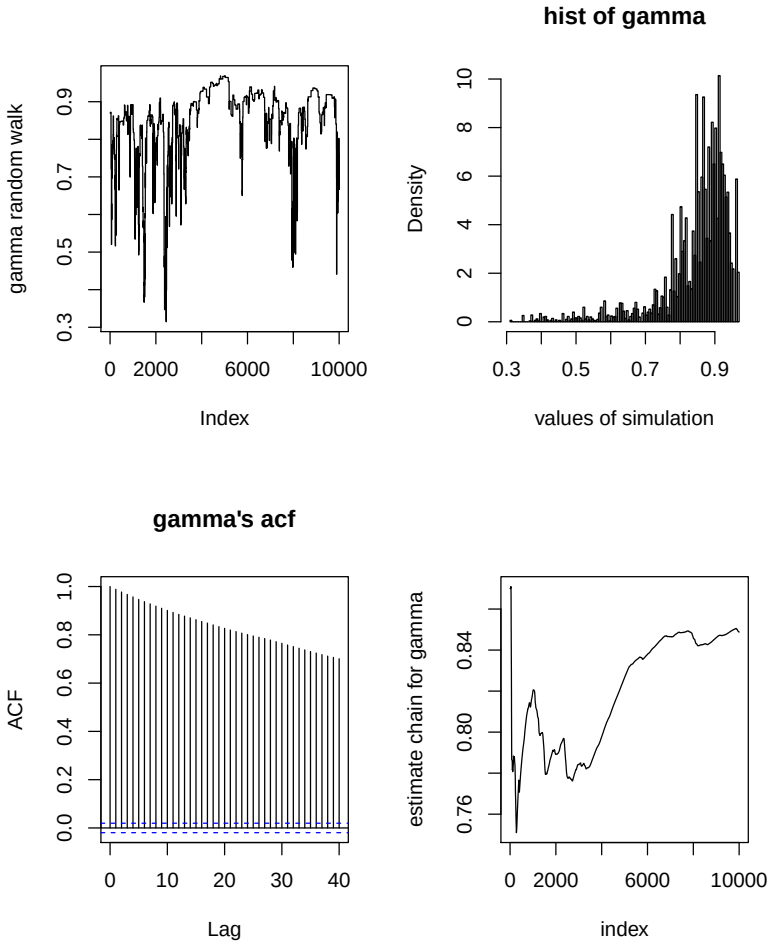
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.01744 0.03456 0.04096 0.04294 0.04915 0.16430
```




```
## mean estimator for tau.square                standard error
##                    0.0429384104                  0.0002432464
```

for γ, I have this results:

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.3146  0.8152  0.8721  0.8488  0.9125  0.9694
```




```
## mean estimator for gamma           standard error
##              0.848790588              0.008719695
```

Part -g

5g) Which parameter has the largest posterior uncertainty? How did you measure it?

-Answer

for the posterior uncertainty, i used the formula of the 'relative error':

$$\frac{\sqrt{\mathbb{V}(\theta)}}{\mathbb{E}(\theta)}$$

```
c("relative error for alpha"=mcmcse::mcse(simulation1[,1])$se/mean(simulation1[,1]) ,
  "relative error for beta"=mcmcse::mcse(simulation1[,2])$se/mean(simulation1[,2])
  ,"relative error for gamma"=mcmcse::mcse(simulation1[,3])$se/mean(simulation1[,3]),
  "relative error for tau.square"=mcmcse::mcse(simulation1[,4])$se/mean(simulation1[,4]))
```

```
##      relative error for alpha       relative error for beta
##                   0.006493230                   0.009097644
##      relative error for gamma relative error for tau.square
##                   0.010273082                   0.005665007
```

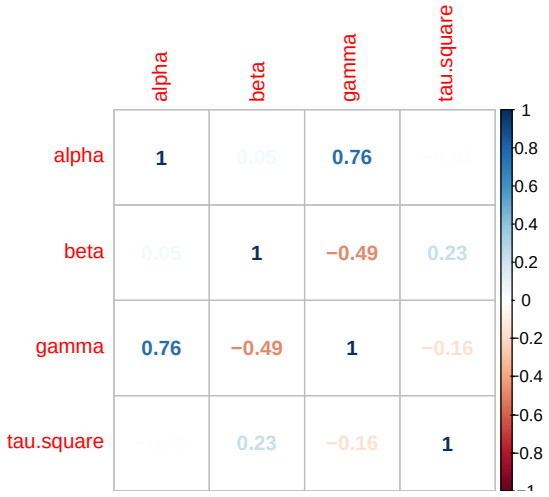
So the parameter with the largest uncertaintly is gamma.

Part -h

5h) Which couple of parameters has the largest correlation (in absolute value)?

-Answer

```
colnames(simulation1) <- c("alpha","beta", 'gamma','tau.square')
corrplot(cor(simulation1),method = "number")
```

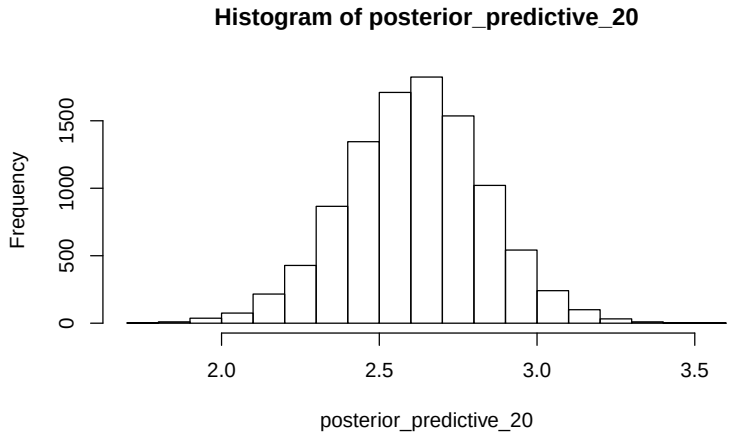


Part -i and -j

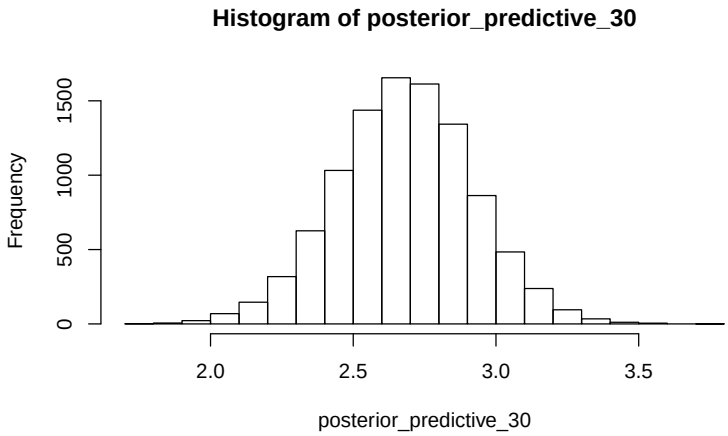
5i) Use the Markov chain to approximate the posterior predictive distribution of the length of a dugong with age of 20 years.

j) Provide the prediction of another dugong with age 30

```
likel <- function(vector,xi){
  alpha.hat=vector[1]
  beta.hat =vector[2]
  gamma.hat =vector[3]
  tau.square.hat = vector[4]
  mu.hat = (alpha.hat - beta.hat*(gamma.hat^(xi)))
  resul  <- rnorm(n = 1, mean = mu.hat, sd = sqrt(tau.square.hat))
  return(resul)

  }
posterior_predictive_20 <- matrix(nrow = dim(simulation1)[1])
for(ind in 1:(dim(simulation1)[1])){
  posterior_predictive_20[ind] <- likel(simulation1[ind,],20)
}

hist(posterior_predictive_20)
```




```
posterior_predictive_30 <- matrix(nrow = dim(simulation1)[1])
for(ind in 1:(dim(simulation1)[1])){
  posterior_predictive_30[ind] <- likel(simulation1[ind,],30)
}

hist(posterior_predictive_30)
```



Part -k

k) Which prediction is less precise?

-Answer

Using the 'relative error':

```
## relative error for prediction of a dugong with 20 years
##                                              0.002131093
## relative error for prediction of a dugong with 30 years
##                                              0.003733091
```

```
## This homework will be graded and it will be part of your final evaluation
##
##
## Last update by LT: Sun May 28 23:51:36 2017
```