HW2 - Tardella - part II: exercises 4 and 5

Your Last+First Name: Rossini Valerio, Your Matricola: 1613638

Part 4

Marchov chain

A Markov chain on a discrete space $\mathcal{S}$ is a stochastic process indexed by a discrete time index t $\{X_t; t = 0, 1, ...\}$ such that $\forall i, j, ..., r, s \in \mathcal{S}$:

$$Pr\{X_{t+1} = r | X_0 = i, X_1 = j, ..., X_t = s\} = Pr\{X_{t+1} = r | X_t = s\}$$

About the basic ingredients for setting up a Markov chain on a general state space: let $t \in \{0, 1, 2, ...\}$ be the index of the process ans $\mathcal{S} \subset \mathbb{R}^k$ the general space of spaces.

1. $\mu \rightarrow$ initial distribution at time t=0
2. transitional kernel $K_t(x, A) = Pr\{X_{t+1} \in A | X_t = x\}$ for each $t = 1, 2, ...$

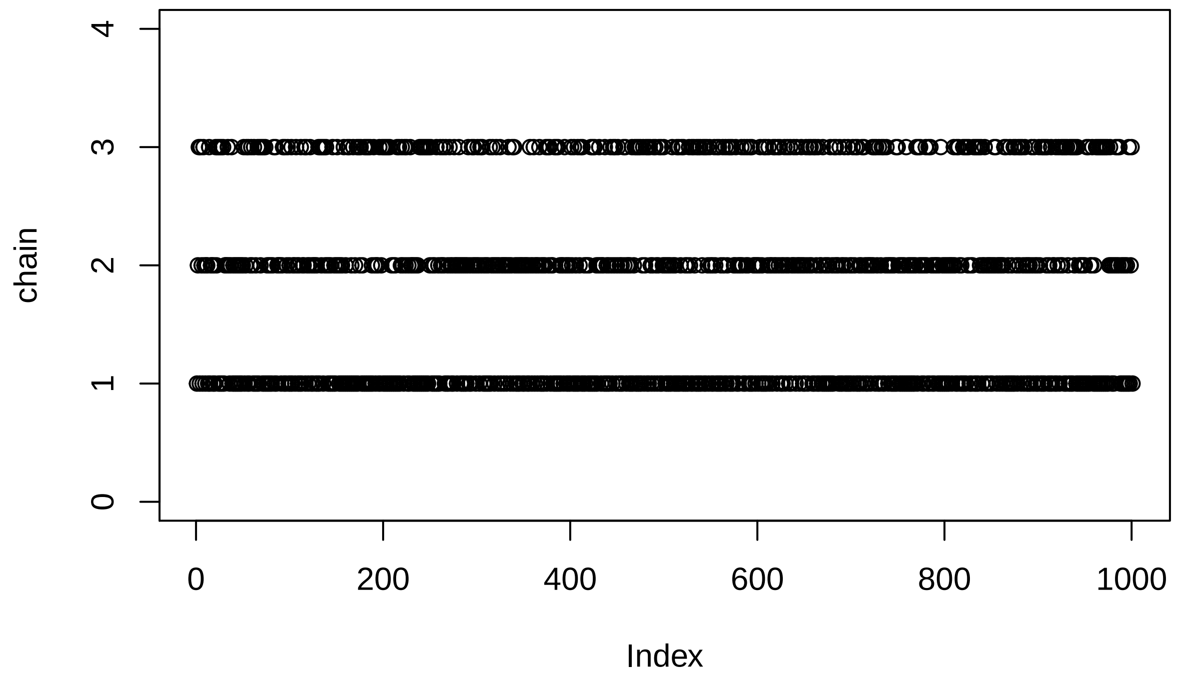
4a answer)

Starting at time $t = 0$ in the state $X_0 = 1$ we can simulate the Markov chain with distribution assigned as above for $t = 1000$ consecutive times. Here below the r code:

```
# Discrete Markov Chain simulation
set.seed(123)
mpt=matrix(c(0, 1/2, 1/2, 5/8, 1/8,
             1/4, 2/3, 1/3, 0),nrow=3,byrow=T)
mpt
```

```
##           [,1]      [,2] [,3]
## [1,] 0.0000000 0.5000000 0.50
## [2,] 0.6250000 0.1250000 0.25
## [3,] 0.6666667 0.3333333 0.00
```

```
S=c(1,2,3)
x0=1

nsample=1000
# vector that will hold all the simulates values starting
# value x1 assigned to chain[1]
chain=rep(NA,nsample+1)
chain[1]=x0
for(t in 1:nsample){
  chain[t+1]=sample(S,size=1,prob=mpt[chain[t],])
}

# the plot shows how many each state occurs in the Markov chain
# generated with distribution assigned in the transition matrix
# and starting point X_0 = 1
plot(chain,ylim=c(0,4))
```



```
table(chain)
```

```
## chain
##   1   2   3
## 386 348 267
```

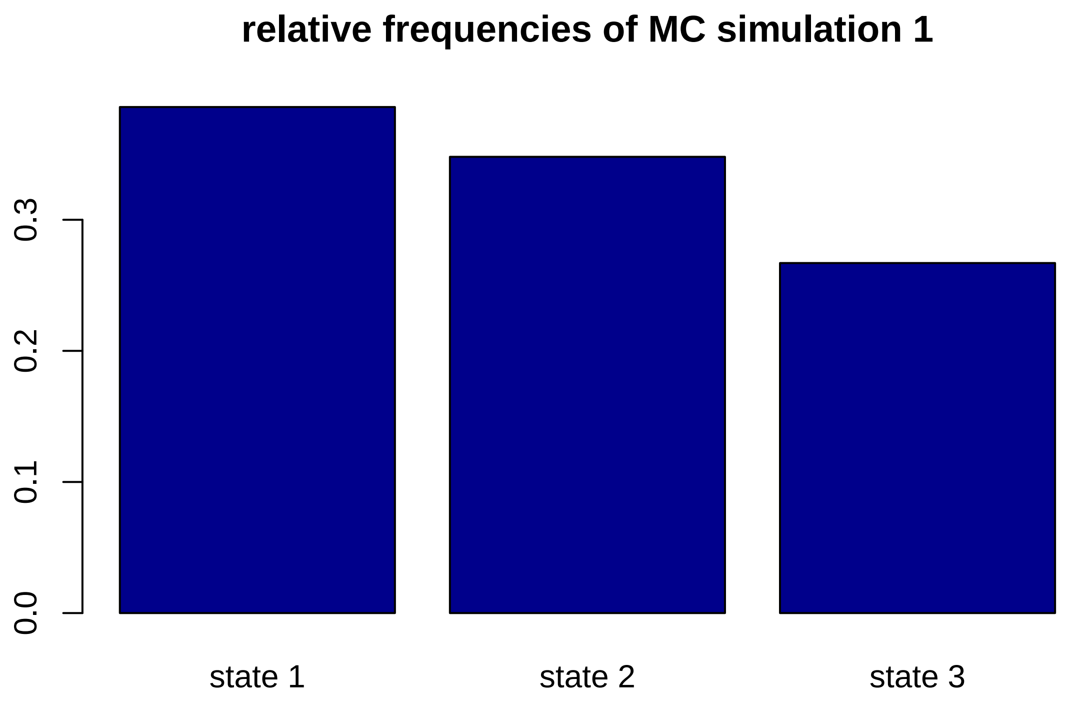
4b answer)

We can compute easily the empirical relative frequency of the three states in our simulation

```
MC_simulation_1=table(chain)/nsample
MC_simulation_1
```

```
## chain
##     1     2     3
## 0.386 0.348 0.267
```

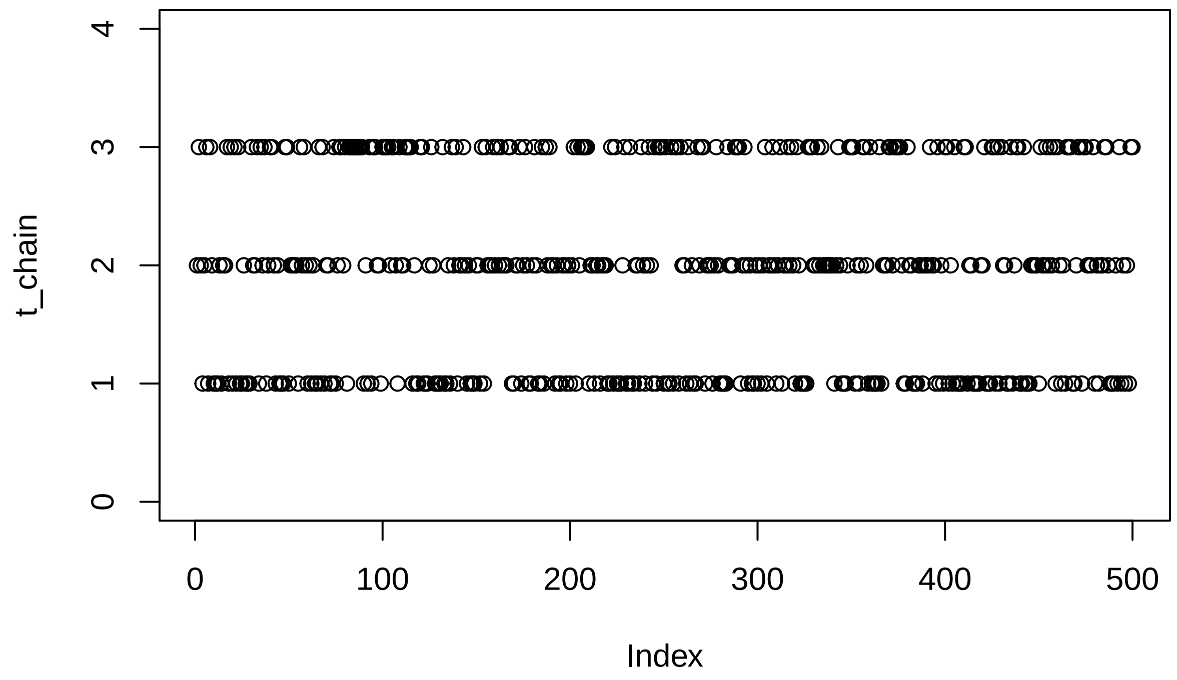
```
barplot(MC_simulation_1, main="relative frequencies of MC simulation 1",
        names.arg=c("state 1", "state 2", "state 3"),col="darkblue")
```



4c answer)

Now we repeat the simulation for 500 times and record only the final state at time $t = 1000$ for each of the 500 simulated chains

```
n=500
t_chain=rep(NA, n)
for (i in 1:n){
  nsample=1000
  chain=rep(NA,nsample+1)
  chain[1]=x0
  for(t in 1:nsample){
    chain[t+1]=sample(S,size=1,prob=mpt[chain[t],])
  }
  t_chain[i]=tail(chain,1)
}

# the plot shows how many each state occurs in the Markov chain
# generated with distribution assigned in the transition matrix
# and starting point X_0 = 1
plot(t_chain,ylim=c(0,4))
```



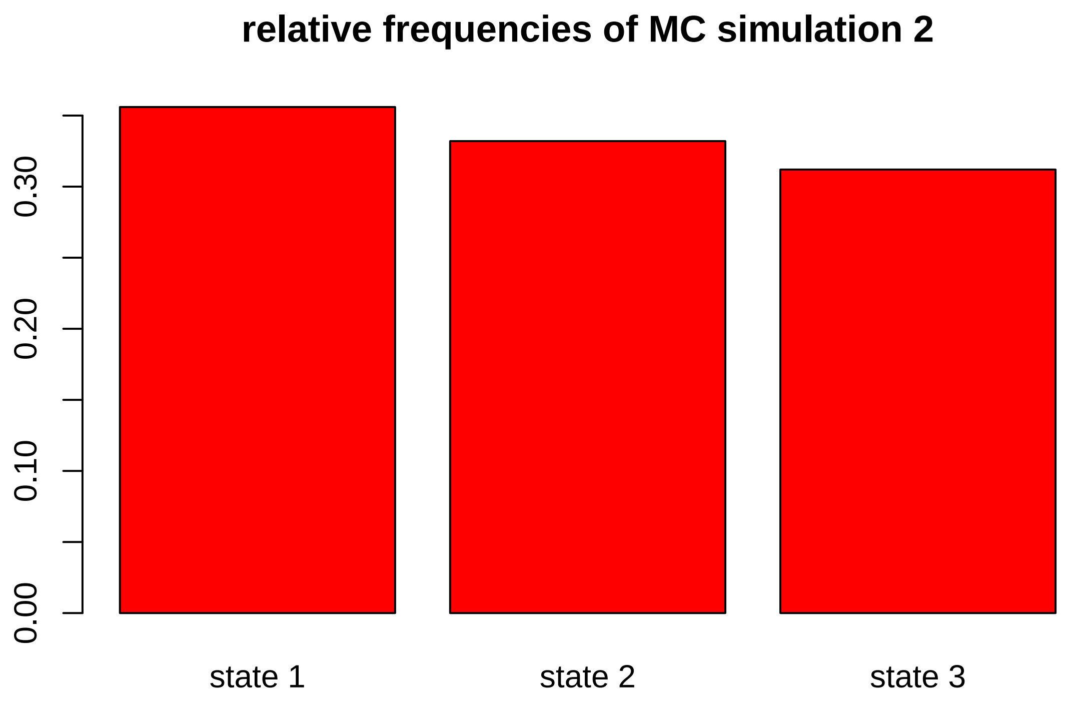
```r
table(t_chain)
```

```
## t_chain
##   1   2   3
## 178 166 156
```

```r
MC_simulation_2=table(t_chain)/n
MC_simulation_2
```

```
## t_chain
##     1     2     3
## 0.356 0.332 0.312
```

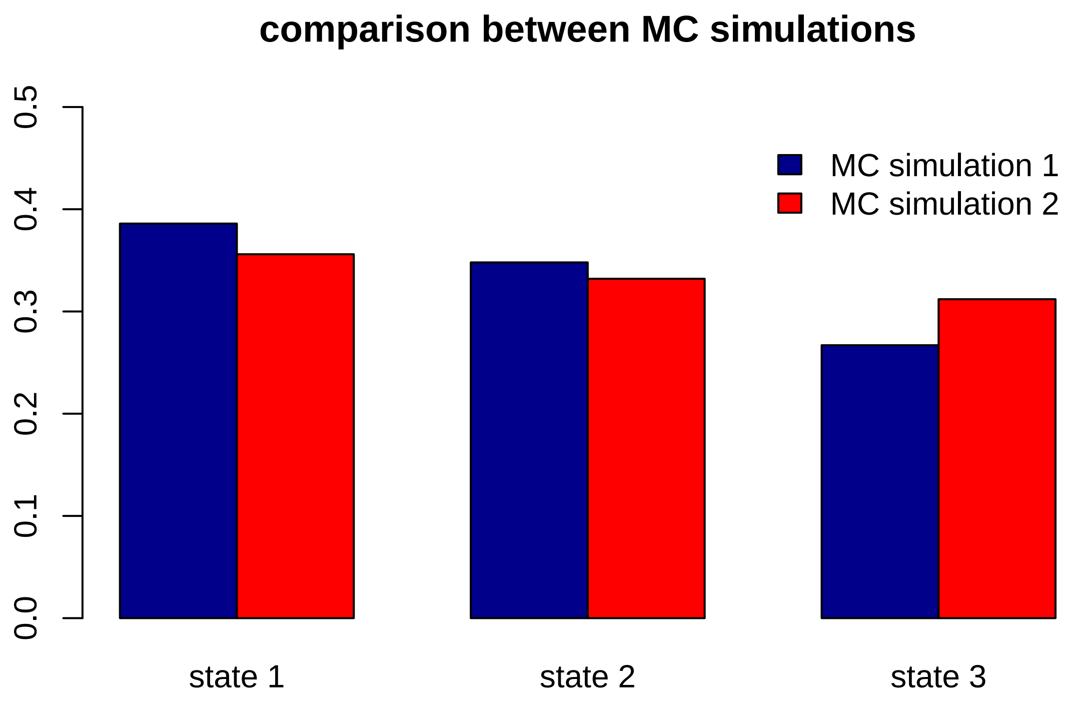
```r
barplot(MC_simulation_2, main="relative frequencies of MC simulation 2",
        names.arg=c("state 1", "state 2", "state 3"), col="red")
```



```
rbind(MC_simulation_1,MC_simulation_2)
```

```
##                     1     2     3
## MC_simulation_1 0.386 0.348 0.267
## MC_simulation_2 0.356 0.332 0.312
```

```
counts=rbind(MC_simulation_1, MC_simulation_2)
barplot(counts, main="comparison between MC simulations", col=c("darkblue","red"),
        names.arg=c("state 1", "state 2", "state 3"),
        legend = c("MC simulation 1","MC simulation 2"),
        beside=TRUE, ylim=c(0,0.5), args.legend=c(bty="n"))
```



When we repeat the simulation for 500 times and record only the final state at time $t = 1000$ for each of the 500 simulated chains, we are not talking properly about of a Markov Chain, because a sample X_t is not depending from the state X_{t-1}. In theory the correct distribution from which we are sampling is:

$$P_{x_0}(X_t \in A) = K^t(x_0, A) = \int_S K(y, A)K^{t-1}(x_0, dy)$$

But it is important to remember that for an ergodic Markov chain (see below the definition of ergodic) the probability of the three states converge to the stationary distribution π. So taking the distribution of the 1000th states of multiple chains could still be a good approximation of the stationary distribution.

If we want to try to formalize the difference between this point and the previous point, we can say that in the previous point we considered the whole Markov chain, from iteration 0 to iteration 1000 and now, in this point, we consider just the last iteration.

definition of ergodic: a state i is said to be ergodic if it is aperiodic and positive recurrent. In other words, a state i is ergodic if it is recurrent, has a period of 1, and has finite mean recurrence time. If all states in an irreducible Markov chain are ergodic, then the chain is said to be ergodic.

4d answer)

We can find the stationary distribution by multiplying the transition matrix by itself over and over again until it finally converges or solving the system or using the eigenvalues.

The stationary distribution $\pi = (\pi_1, \pi_2, \pi_3)^T$ must satisfy the equations

$$\begin{cases} \pi_1 p_{11} + \pi_2 p_{21} + \pi_3 p_{31} = \pi_1 \\ \pi_1 p_{12} + \pi_2 p_{22} + \pi_2 p_{32} = \pi_2 \\ \pi_1 p_{13} + \pi_2 p_{23} + \pi_2 p_{33} = \pi_3 \end{cases}$$

which can be re-written in matrix notation as follows

$$P^T \pi = \pi$$

Hence π must be one of the possible solutions of the following characteristic system of equations

$$(P^T - \lambda I)\pi = 0$$

corresponding to $\lambda = 1$ or, equivalently, π must be in the eigenspace corresponding to the eigenvalue $\lambda = 1$. However, there are infinite possible such solutions. The only one we are interested in is the solution π such that $\pi_1 + \pi_2 + \pi_3 = 1$.

```
# first method
pi=solve(matrix(c(-1, 5/8, 2/3, 1/2, 1/8-1, 1/3, 1, 1, 1), nrow=3, byrow = T), c(0,0,1))
pi
```

```
## [1] 0.3917526 0.3298969 0.2783505
```

```
# second method
mpt_copy = mpt
for(i in 1:100)
  mpt_copy = mpt_copy %*% mpt
mpt_copy
```

```
##           [,1]      [,2]      [,3]
## [1,] 0.3917526 0.3298969 0.2783505
## [2,] 0.3917526 0.3298969 0.2783505
## [3,] 0.3917526 0.3298969 0.2783505
```

```
# confirm that pi*transition matrix = pi
pi%*%mpt_copy
```

```
##           [,1]      [,2]      [,3]
## [1,] 0.3917526 0.3298969 0.2783505
```

```
# third method
eigen(t(mpt))
```

```
## $values
## [1]  1.0000000 -0.6509781 -0.2240219
##
## $vectors
##           [,1]       [,2]        [,3]
## [1,] 0.6720665  0.8083220  0.06959404
## [2,] 0.5659508 -0.3043531 -0.73933055
## [3,] 0.4775210 -0.5039689  0.66973652
```

```
eigen(t(mpt))$vector[,1]
```

```
## [1] 0.6720665 0.5659508 0.4775210
```

```
pi=eigen(t(mpt))$vector[,1]/sum(eigen(t(mpt))$vector[,1])
pi
```

```
## [1] 0.3917526 0.3298969 0.2783505
```

```
# confirm that pi*transition matrix = pi
pi%*%mpt
```

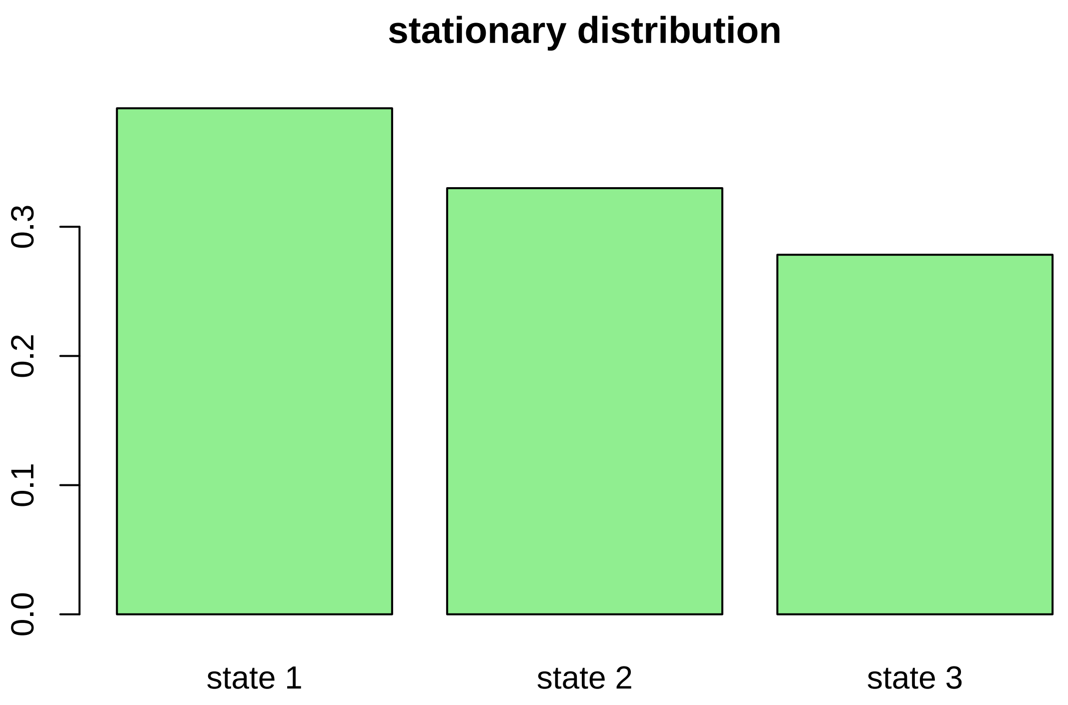
```
##           [,1]      [,2]      [,3]
## [1,] 0.3917526 0.3298969 0.2783505
```

4e answer)

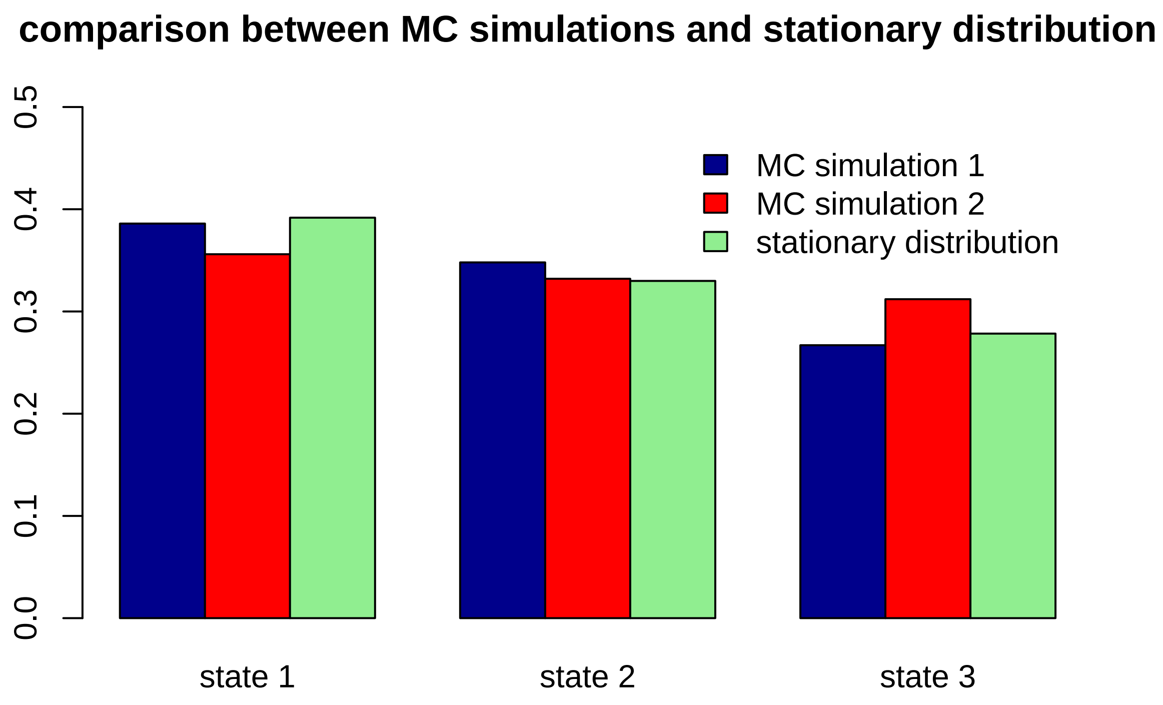
The π distribution is well approximated by simulated empirical relative frequencies both in the Markov chain in point (b) and in point (c).

We can see that the difference between π and the empirical relative frequencies are almost insignificant

```
barplot(pi, main="stationary distribution",
        names.arg=c("state 1", "state 2", "state 3"), col="lightgreen")
```



```
counts=rbind(MC_simulation_1, MC_simulation_2, pi)
barplot(counts, main="comparison between MC simulations and stationary distribution",
        names.arg=c("state 1", "state 2", "state 3"), ylim=c(0,0.5),
        legend = c("MC simulation 1","MC simulation 2","stationary distribution"),
        beside=TRUE, args.legend=c(bty="n"), col=c("darkblue","red","lightgreen"))
```



```
first_difference=pi-MC_simulation_1
first_difference
```

```
## chain
##           1            2            3
##  0.005752577 -0.018103093  0.011350515
```

```
second_difference=pi-MC_simulation_2
second_difference
```

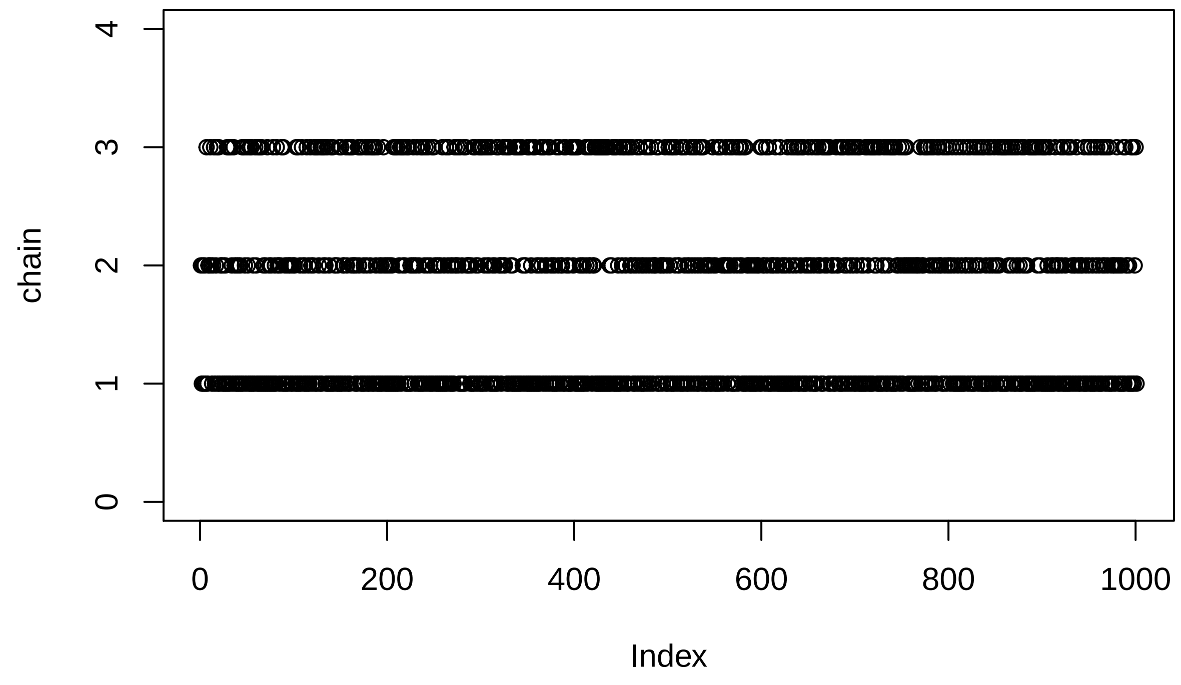
```
## t_chain
##           1            2            3
##  0.035752577 -0.002103093 -0.033649485
```

4f answer)

The ergodic properties of MC will ensure that will be no difference starting from any other state, so we don't care if we start at $t = 0$ from state $X_0 = 2$ instead of $X_0 = 1$

```
x0=2

nsample=1000
chain=rep(NA,nsample+1)
chain[1]=x0
for(t in 1:nsample){
  chain[t+1]=sample(S,size=1,prob=mpt[chain[t],])
}
```

```r
# the plot shows how many each state occurs in the Markov chain
# generated with distribution assigned in the transition matrix
# and starting point X_0 = 1
plot(chain,ylim=c(0,4))
```



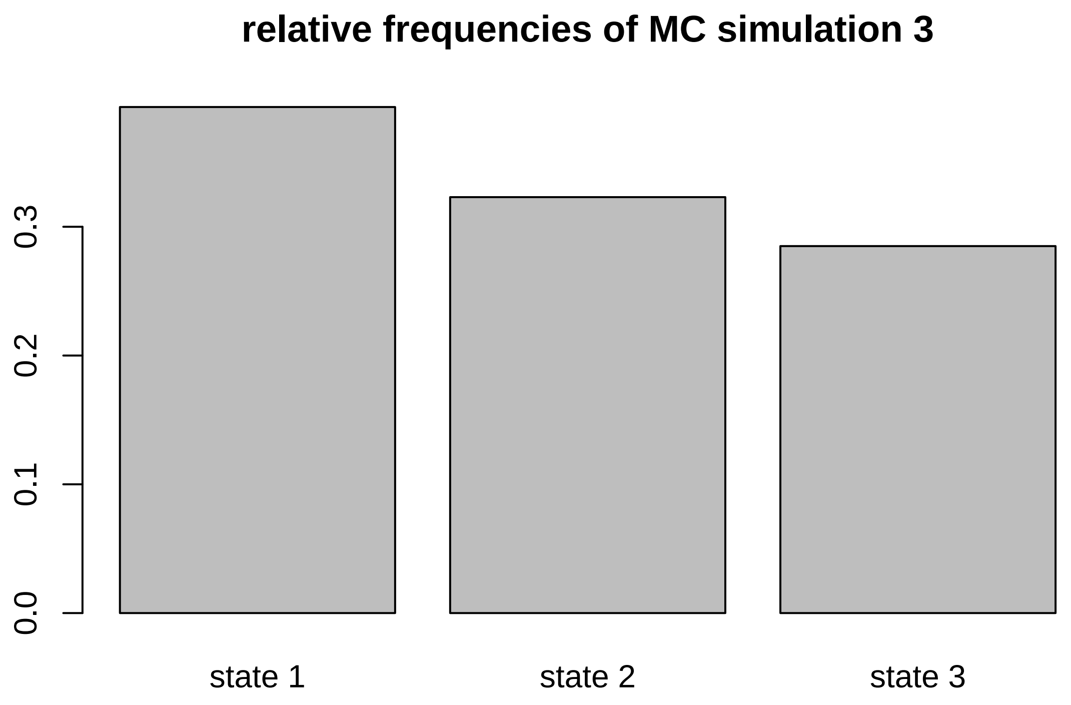
```r
table(chain)
```

```
## chain
##   1   2   3
## 393 323 285
```

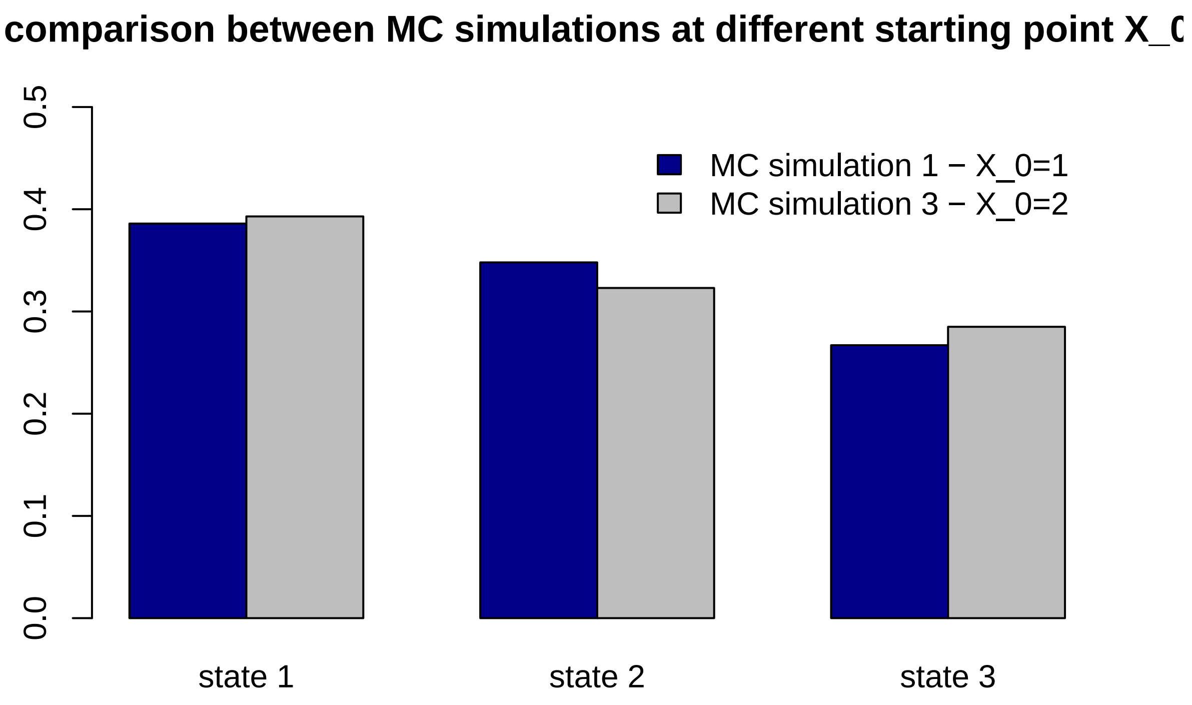
```r
MC_simulation_3=table(chain)/nsample
MC_simulation_3
```

```
## chain
##     1     2     3
## 0.393 0.323 0.285
```

```r
barplot(MC_simulation_3, main="relative frequencies of MC simulation 3",
        names.arg=c("state 1", "state 2", "state 3"),col="grey")
```



```
counts=rbind(MC_simulation_1, MC_simulation_3)
barplot(counts, main="comparison between MC simulations at different starting point X_0",
        col=c("darkblue","grey"), names.arg=c("state 1", "state 2", "state 3"),
        legend = c("MC simulation 1 - X_0=1","MC simulation 3 - X_0=2"),
        beside=TRUE, ylim=c(0,0.5), args.legend=c(bty="n"))
```



Part 5

Dugongs - part II

5a answer)

Full conditional for α:

$$\pi(\alpha|\beta,\gamma,\tau^2,x,y)=\frac{1}{(2\pi\tau^2)^{\frac{n}{2}}}\cdot exp\left\{-\frac{1}{2\tau^2}\sum_{i=1}^{n}(y_i-\mu_i)^2\right\}\frac{1}{\sqrt{2\pi\sigma_\alpha^2}}exp\left\{-\frac{\alpha^2}{2\sigma_\alpha^2}\right\}\mathbb{I}_{(1,\infty)}(\alpha)$$

$$=\frac{1}{(2\pi\tau^2)^{\frac{n}{2}}}\cdot exp\left\{-\frac{1}{2\tau^2}\sum_{i=1}^{n}(y_i-\alpha+\beta\gamma^{x_i})^2\right\}\frac{1}{\sqrt{2\pi\sigma_\alpha^2}}exp\left\{-\frac{\alpha^2}{2\sigma_\alpha^2}\right\}\mathbb{I}_{(1,\infty)}(\alpha)$$

$$\propto exp\left\{-\frac{1}{2\tau^2}\sum_{i=1}^{n}(y_i-\alpha+\beta\gamma^{x_i})^2\right\}exp\left\{-\frac{\alpha^2}{2\sigma_\alpha^2}\right\}\mathbb{I}_{(1,\infty)}(\alpha)\propto exp\left\{-\frac{\alpha^2}{2\sigma_\alpha^2}-\frac{\sum_{i=1}^{n}(\alpha^2-2\alpha y_i-2\alpha\beta\gamma^{x_i})}{2\tau^2}\right\}\mathbb{I}_{(1,\infty)}(\alpha)$$

$$\propto exp\left\{-\frac{\alpha^2(\tau^2+n\sigma_\alpha^2)-2\alpha\sigma_\alpha^2\sum_{i=1}^{n}(y_i+\beta\gamma^{x_i})}{2\sigma_\alpha^2\tau^2}\right\}\mathbb{I}_{(1,\infty)}(\alpha)\propto exp\left\{-\frac{\alpha^2}{2}\cdot\frac{\tau^2+n\sigma_\alpha^2}{\sigma_\alpha^2\tau^2}+\alpha\cdot\frac{\sigma_\alpha^2\sum_{i=1}^{n}(y_i+\beta\gamma^{x_i})}{\sigma_\alpha^2\tau^2}\right\}\mathbb{I}_{(1,\infty)}(\alpha)$$

$$\implies\pi(\alpha|\beta,\gamma,\tau^2,x,y)\sim N_{(1,\infty)}\left(\frac{\sigma_\alpha^2\sum_{i=1}^{n}(y_i+\beta\gamma^{x_i})}{\tau^2+n\sigma_\alpha^2},\frac{\sigma_\alpha^2\tau^2}{\tau^2+n\sigma_\alpha^2}\right)$$

Full conditional for β:

$$\pi(\beta|\alpha,\gamma,\tau^2,x,y) = \frac{1}{(2\pi\tau^2)^{\frac{n}{2}}} \cdot exp\left\{-\frac{1}{2\tau^2}\sum_{i=1}^{n}(y_i-\mu_i)^2\right\} \frac{1}{\sqrt{2\pi\sigma_\beta^2}} exp\left\{-\frac{\beta^2}{2\sigma_\beta^2}\right\} \mathbb{I}_{(1,\infty)}(\beta)$$

$$= \frac{1}{(2\pi\tau^2)^{\frac{n}{2}}} \cdot exp\left\{-\frac{1}{2\tau^2}\sum_{i=1}^{n}(y_i-\alpha+\beta\gamma^{x_i})^2\right\} \frac{1}{\sqrt{2\pi\sigma_\beta^2}} exp\left\{-\frac{\beta^2}{2\sigma_\beta^2}\right\} \mathbb{I}_{(1,\infty)}(\beta)$$

$$\propto exp\left\{-\frac{1}{2\tau^2}\sum_{i=1}^{n}(y_i-\alpha+\beta\gamma^{x_i})^2\right\} exp\left\{-\frac{\beta^2}{2\sigma_\beta^2}\right\} \mathbb{I}_{(1,\infty)}(\beta) \propto exp\left\{-\frac{\beta^2}{2\sigma_\beta^2} - \frac{\sum_{i=1}^{n}(\beta^2\gamma^{2x_i}-2\beta y_i\gamma^{x_i}-2\alpha\beta\gamma^{x_i})}{2\tau^2}\right\} \mathbb{I}_{(1,\infty)}(\beta)$$

$$\propto exp\left\{-\frac{\beta^2(\tau^2+\sigma_\beta^2\sum_{i=1}^{n}\gamma^{2x_i}) - 2\beta\sigma_\beta^2\sum_{i=1}^{n}(\alpha\gamma^{x_i}-y_i\gamma^{x_i})}{2\sigma_\beta^2\tau^2}\right\} \mathbb{I}_{(1,\infty)}(\beta)$$

$$\propto exp\left\{-\frac{\beta^2(\tau^2+\sigma_\beta^2\sum_{i=1}^{n}\gamma^{2x_i}) - 2\beta\sigma_\beta^2\sum_{i=1}^{n}\gamma^{x_i}(\alpha-y_i)}{2\sigma_\beta^2\tau^2}\right\} \mathbb{I}_{(1,\infty)}(\beta)$$

$$\propto exp\left\{-\frac{\beta^2}{2} \cdot \frac{\tau^2+\sigma_\beta^2\sum_{i=1}^{n}\gamma^{2x_i}}{\sigma_\beta^2\tau^2} + \beta \cdot \frac{\sigma_\beta^2\sum_{i=1}^{n}\gamma^{x_i}(\alpha-y_i)}{\sigma_\beta^2\tau^2}\right\} \mathbb{I}_{(1,\infty)}(\beta)$$

$$\implies \pi(\beta|\alpha,\gamma,\tau^2,x,y) \sim N_{(1,\infty)}\left(\frac{\sigma_\beta^2\sum_{i=1}^{n}\gamma^{x_i}(\alpha-y_i)}{\tau^2+\sigma_\beta^2\sum_{i=1}^{n}\gamma^{2x_i}}, \frac{\sigma_\beta^2\tau^2}{\tau^2+\sigma_\beta^2\sum_{i=1}^{n}\gamma^{2x_i}}\right)$$

Full conditional for γ:

$$\pi(\gamma|\alpha,\beta,\tau^2,x,y) = \frac{1}{(2\pi\tau^2)^{\frac{n}{2}}} \cdot exp\left\{-\frac{1}{2\tau^2}\sum_{i=1}^{n}(y_i-\mu_i)^2\right\} \mathbb{I}_{(0,1)}(\gamma)$$

$$= \frac{1}{(2\pi\tau^2)^{\frac{n}{2}}} \cdot exp\left\{-\frac{1}{2\tau^2}\sum_{i=1}^{n}(y_i-\alpha+\beta\gamma^{x_i})^2\right\} \mathbb{I}_{[0,1]}(\gamma) \propto exp\left\{-\frac{1}{2\tau^2}\sum_{i=1}^{n}(y_i-\alpha+\beta\gamma^{x_i})^2\right\} \mathbb{I}_{(0,1)}(\gamma)$$

Full conditional for τ^2:

$$\pi(\tau^2|\alpha,\beta,\gamma,x,y) = \frac{1}{\tau^{2\left(\frac{n}{2}\right)}\tau^{2(a+1)}} exp\left\{\frac{-\frac{1}{2}\sum_{i=1}^{n}(y_i-\alpha+\beta\gamma^{x_i})^2 - b}{\tau^2}\right\} \mathbb{I}_{(0,\infty)}(\tau^2)$$

$$\frac{1}{\tau^{2\left(\frac{n}{2}+a+1\right)}} exp\left\{-\frac{b+\frac{1}{2}\sum_{i=1}^{n}(y_i-\alpha+\beta\gamma^{x_i})^2}{\tau^2}\right\} \mathbb{I}_{[0,1]}(\tau^2)$$

$$\implies \pi(\tau^2|\alpha,\beta,\gamma,x,y) \sim IG_{(0,\infty)}\left(a+\frac{n}{2}, b+\frac{1}{2}\sum_{i=1}^{n}(y_i-\alpha+\beta\gamma^{x_i})^2\right)$$

5b answer)

The parameters α, β, τ^2 have their own full conditional within standard parametric families (Normal for α and β, and InverseGamma for τ^2), so we can easily simulate from them. Instead for the parameter γ we don't have a standard parametric family, so we implement the Metropolis-Hastings algorithm.

5c answer)

About the values of $\sigma^2_\alpha, \sigma^2_\beta, a, b$ we cose the same values of the previous exercise 1, i.e. $\sigma^2_\alpha = 10000, \sigma^2_\beta = 10000, a = 0.001, b = 0.001$

```
df=read.csv("dugongs.txt",sep="\t")

# let's see the variables into the dataframe
names(df)
```

```
## [1] "Age"    "Length"
```

```
# number of dugongs present in the dataframe
n = dim(df)[1]
n
```

```
## [1] 27
```

```
# the dependent variable represents the length of the dugongs, while the dependent
# variable represents the age of the dugongs
x = df$Age
y = df$Length

## x = age
# x = c( 1.0,  1.5,  1.5,  1.5, 2.5,   4.0,  5.0,  5.0,  7.0,
#        8.0,  8.5,  9.0,  9.5, 9.5,  10.0, 12.0, 12.0, 13.0,
#        13.0, 14.5, 15.5, 15.5, 16.5, 17.0, 22.5, 29.0, 31.5)

## y = lenght
# y = c(1.80, 1.85, 1.87, 1.77, 2.02, 2.27, 2.15, 2.26, 2.47,
#       2.19, 2.26, 2.40, 2.39, 2.41, 2.50, 2.32, 2.32, 2.43,
#       2.47, 2.56, 2.65, 2.47, 2.64, 2.56, 2.70, 2.72, 2.57)


# load tha packages
library(MCMCpack)
```

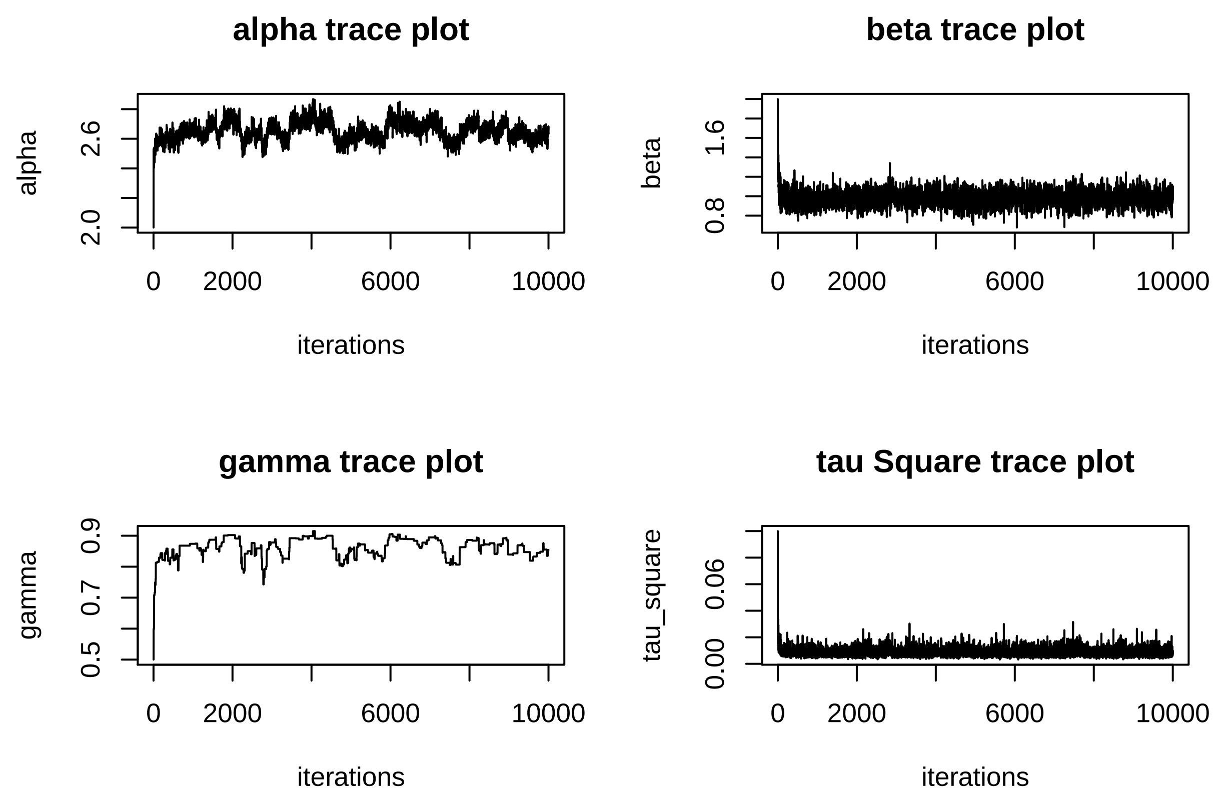
```
## Warning: package 'MCMCpack' was built under R version 3.3.3

## Loading required package: coda

## Warning: package 'coda' was built under R version 3.3.3

## Loading required package: MASS

## ##
## ## Markov Chain Monte Carlo Package (MCMCpack)

## ## Copyright (C) 2003-2017 Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park

## ##
## ## Support provided by the U.S. National Science Foundation
## ## (Grants SES-0350646 and SES-0350613)
## ##
```

```
iterations = 10000
alpha = rep(NA, iterations+1)
beta = rep(NA, iterations+1)
gamma = rep(NA, iterations+1)
tau_square = rep(NA, iterations+1)

# initial states for each parameter
alpha[1] = 2
beta[1] = 2
gamma[1] = 0.5
tau_square[1] = 0.1

# full conditional gamma
full_cond_gamma = function(gamma, alpha, beta, tau_square) {
  return(exp(-1/(2*tau_square)*sum((y-alpha+beta*gamma^x)^2)))
}

for(t in 1:iterations) {
  # update alpha Markov chain
  alpha[t+1] = rnorm(1,(10000*sum(y+beta[t]*gamma[t]^x))/(n*10000+tau_square[t]),
                     sqrt((tau_square[t]*10000)/(n*10000 + tau_square[t])))

  # update beta Markov chain
  beta[t+1] = rnorm(1,(10000*sum((alpha[t+1]-y)*gamma[t]^x)) /
                      (10000*sum(gamma[t]^(2*x))+tau_square[t]),
                    sqrt((tau_square[t]*10000)/(10000*sum(gamma[t]^(2*x))+tau_square[t])))

  # MH to update gamma Markov chain
  proposal = runif(1,0,1)
  prob = full_cond_gamma(proposal, alpha[t+1], beta[t+1], tau_square[t]) /
    full_cond_gamma(gamma[t], alpha[t+1], beta[t+1], tau_square[t])
  gamma[t+1] = ifelse(runif(1,0,1)<=prob, proposal, gamma[t])

  # update tau square markov chain
  tau_square[t+1] = rinvgamma(1, n/2 + 0.001,
                              0.001+1/2*sum((y-alpha[t+1]+beta[t+1]*(gamma[t+1]^x))^2))

}
MG = cbind(alpha,beta, gamma, tau_square)
head(MG)
```

```
##         alpha     beta     gamma tau_square
## [1,] 2.000000 2.000000 0.5000000 0.10000000
## [2,] 2.449305 1.902389 0.5000000 0.01996861
## [3,] 2.493373 1.747236 0.5987425 0.01850918
## [4,] 2.531535 1.448931 0.5987425 0.02002107
## [5,] 2.463318 1.388475 0.5987425 0.01944214
## [6,] 2.506067 1.284952 0.5987425 0.03371453
```

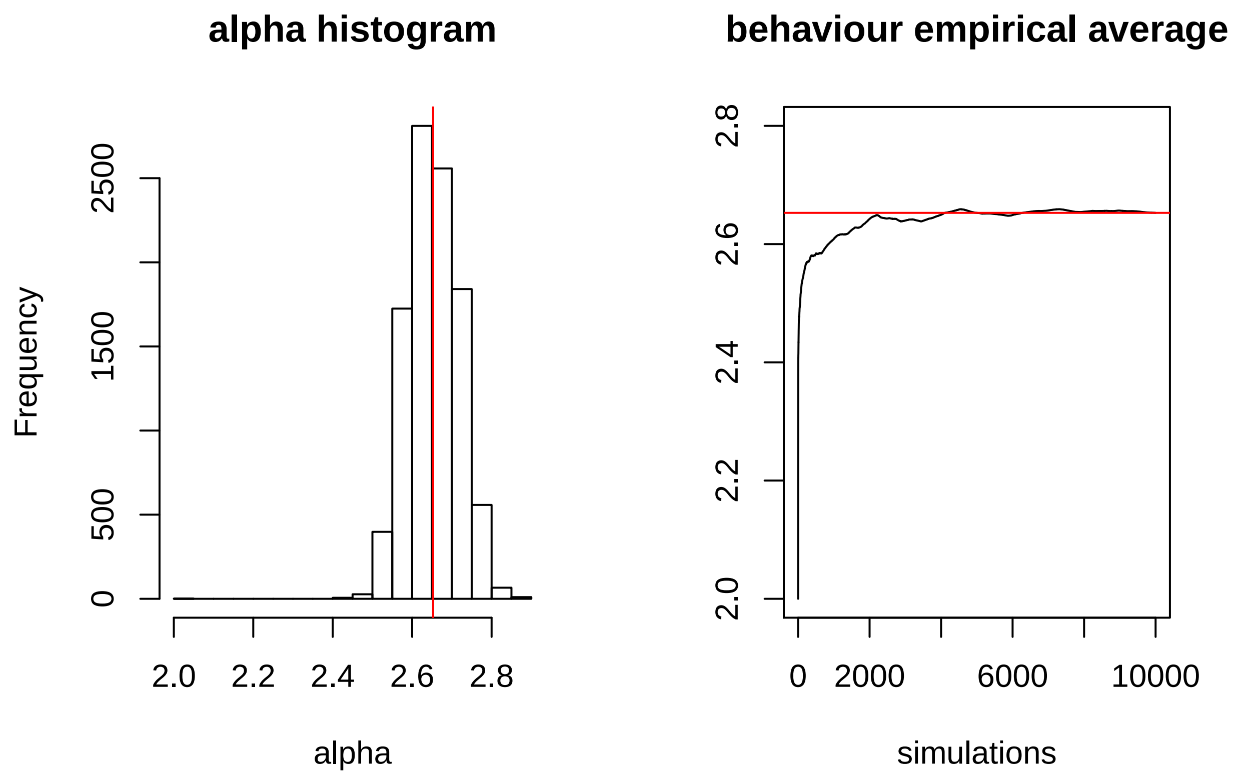
5d answer)

```r
# Show the 4 univariate trace-plots of the simulations of each parameter

par(mfrow=c(2,2))
plot(alpha, xlab = "iterations", main="alpha trace plot",type="l")
plot(beta, xlab = "iterations", main="beta trace plot",type="l")
plot(gamma, xlab = "iterations", main="gamma trace plot",type="l")
plot(tau_square, xlab = "iterations", main="tau Square trace plot",type="l")
```

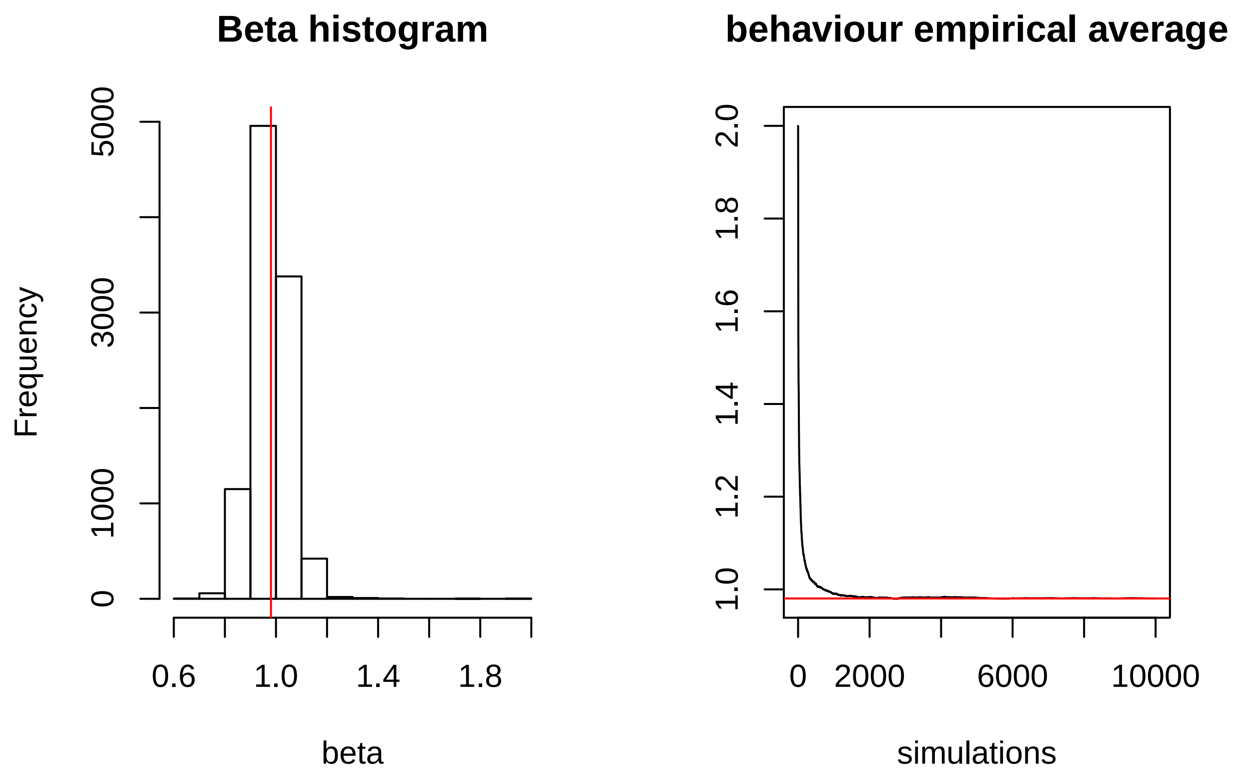


5e answer)

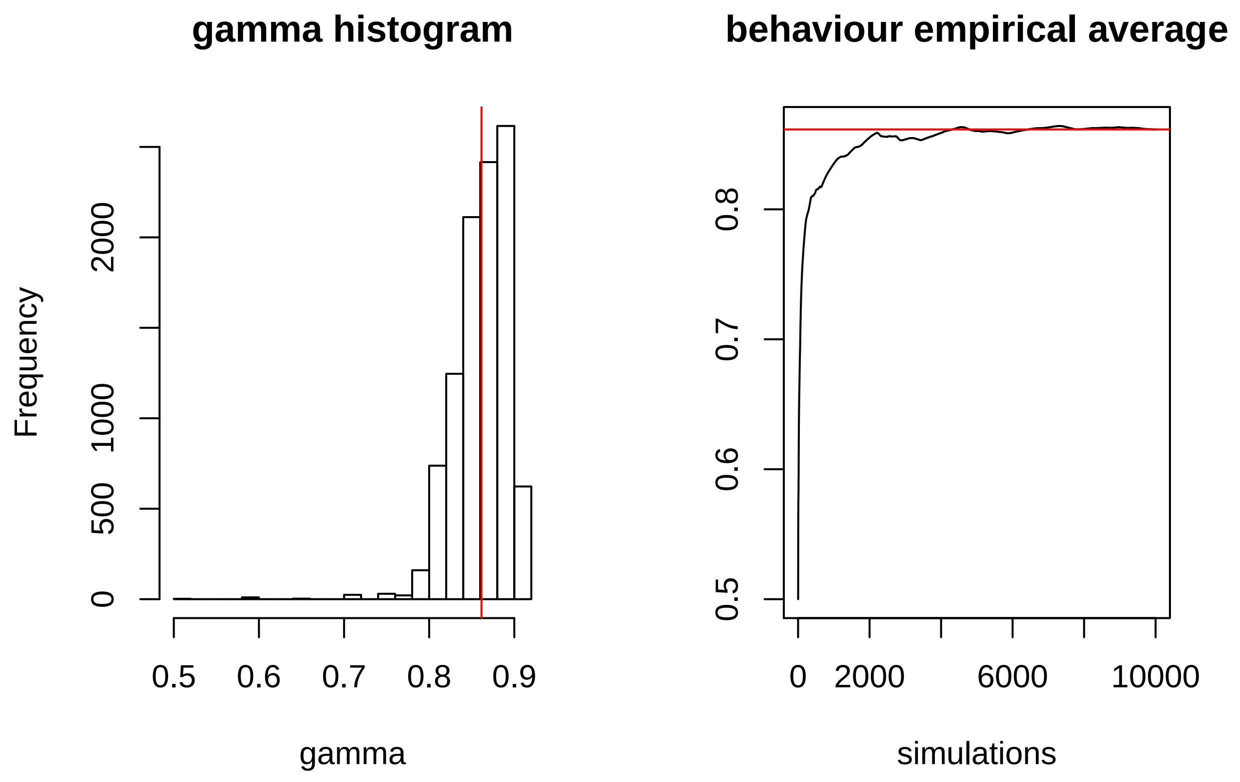
```r
par(mfrow=c(1,2))

#alpha
hist(alpha, main= "alpha histogram", xlab = "alpha")
abline(v=mean(alpha), col="red")
plot(cumsum(alpha)/(1:length(alpha)), type="l",ylim = c(2, 2.8), ylab="",
     main="behaviour empirical average", xlab="simulations")
abline(h=mean(alpha), col="red")
```



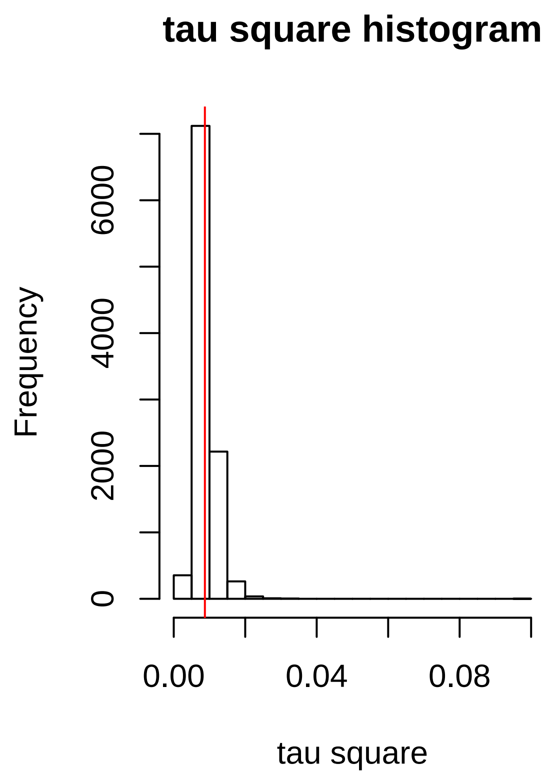
```
#beta
hist(beta, main= "Beta histogram", xlab = "beta")
abline(v=mean(beta), col="red")
plot(cumsum(beta)/(1:length(beta)), type="l", ylab="",
     main="behaviour empirical average", xlab="simulations")
abline(h=mean(beta), col="red")
```

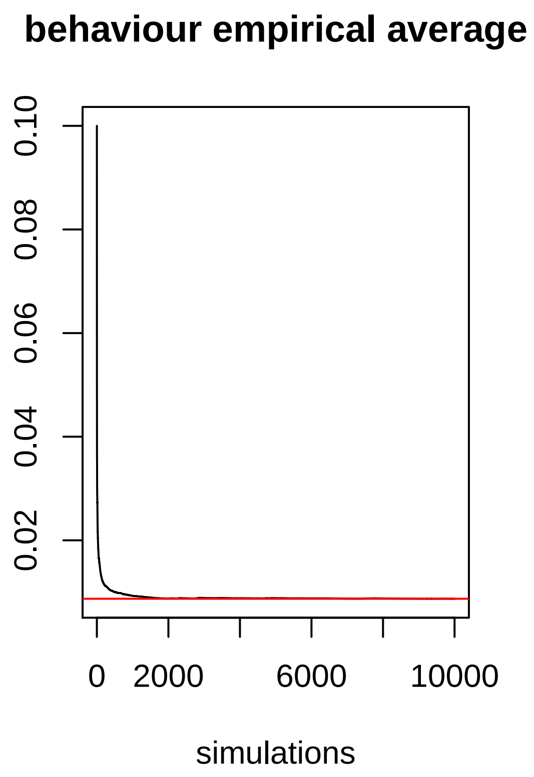


```r
#gamma
hist(gamma, main= "gamma histogram", xlab = "gamma")
abline(v=mean(gamma), col="red")
plot(cumsum(gamma)/(1:length(gamma)), type="l", ylab="",
     main="behaviour empirical average", xlab="simulations")
abline(h=mean(gamma), col="red")
```



```r
#tau square
hist(tau_square, main= "tau square histogram", xlab = "tau square")
abline(v=mean(tau_square), col="red")
plot(cumsum(tau_square)/(1:length(tau_square)), type="l", ylab="",
     main="behaviour empirical average", xlab="simulations")
abline(h=mean(tau_square), col="red")
```





Let's note that the empirical average, with growing t, converges to the mean of the whole Markov chain

5f answer)

The approximation error can be computed in this way:

$$\mathbb{E}\left[(\hat{I_n} - I)^2\right] = \mathbb{V}\left[\hat{I_n}\right] = \frac{1}{n}\mathbb{V}\left[h(X)\right] = \frac{1}{n}\left\{\mathbb{E}_\pi[h(X)^2] - \mathbb{E}_\pi[h(X)]^2\right\} = \frac{K}{n}$$

Since we don't know tha value of K, we can approximate it:

$$\hat{K} = \hat{V}[h(X)] = \frac{1}{n}\sum_{1=1}^{n} h(X_i)^2 - \hat{I}_n^2$$

```
# estimates for each parameter
alpha_est=mean(alpha)
beta_est=mean(beta)
gamma_est=mean(gamma)
tau_square_est=mean(tau_square)
rbind(alpha_est,beta_est,gamma_est,tau_square_est)
```

```
##                       [,1]
## alpha_est      2.652829897
## beta_est       0.980438750
## gamma_est      0.861488428
## tau_square_est 0.008714126
```

```r
# approximation error for each parameter
approx_err_alpha=var(alpha)/length(alpha)
approx_err_beta=var(beta)/length(beta)
approx_err_gamma=var(gamma)/length(gamma)
approx_err_tau_square=var(tau_square)/length(tau_square)
rbind(approx_err_alpha,approx_err_beta,approx_err_gamma,approx_err_tau_square)
```

```
##                               [,1]
## approx_err_alpha      3.996503e-07
## approx_err_beta       5.316134e-07
## approx_err_gamma      1.016836e-07
## approx_err_tau_square 8.679969e-10
```

5g answer)

To find the parameter with the largest posterior uncertainty we use the coefficient of variation given by the ration between σ and the absolute value of μ

```r
# coefficient of variation for each parameter
var_coeff_alpha=sd(alpha)/abs(alpha_est)
var_coeff_beta=sd(beta)/abs(beta_est)
var_coeff_gamma=sd(gamma)/abs(gamma_est)
var_coeff_tau_square=sd(tau_square)/abs(tau_square_est)
rbind(var_coeff_alpha,var_coeff_beta,var_coeff_gamma,var_coeff_tau_square)
```

```
##                            [,1]
## var_coeff_alpha      0.02383155
## var_coeff_beta       0.07437024
## var_coeff_gamma      0.03701670
## var_coeff_tau_square 0.33810918
```

The parameter with the largest posterior uncertainty is tau square with a value of:

```r
var_coeff_tau_square
```

```
## [1] 0.3381092
```

5h answer)

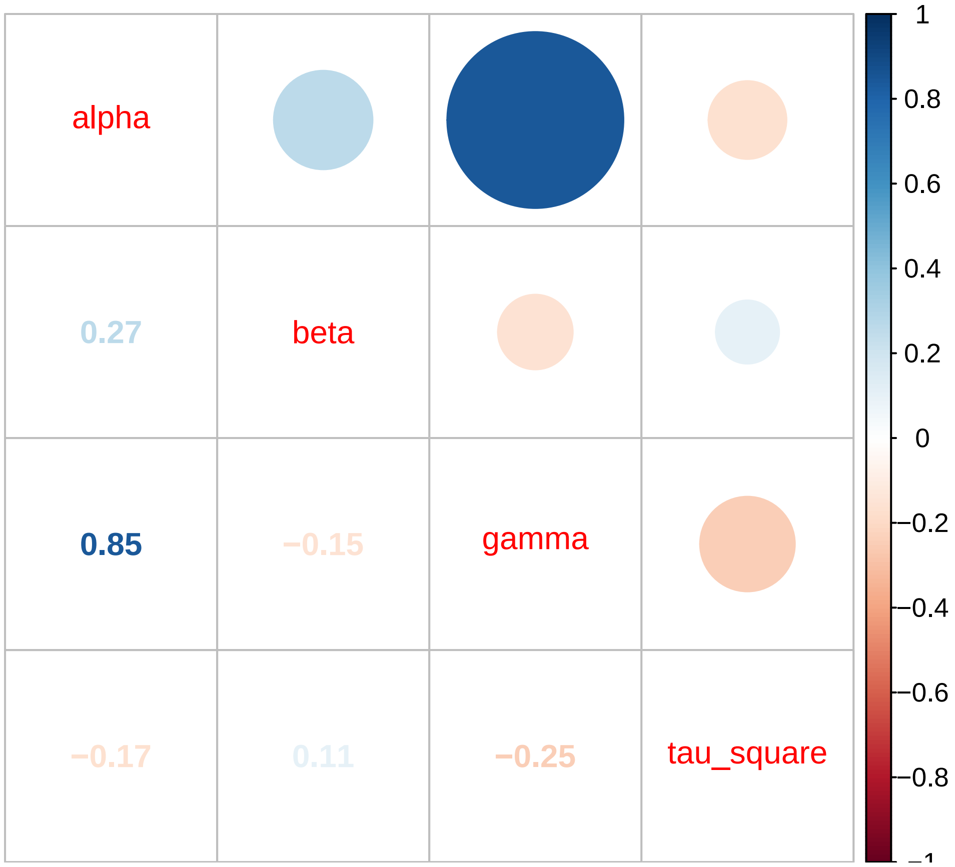
```r
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 3.3.3
```

```r
par(mfrow=c(1,1))
correlation=cor(MG)
correlation
```

```
##                 alpha       beta      gamma tau_square
## alpha       1.0000000  0.2656226  0.8487753 -0.1657500
## beta        0.2656226  1.0000000 -0.1526013  0.1091138
## gamma       0.8487753 -0.1526013  1.0000000 -0.2452141
## tau_square -0.1657500  0.1091138 -0.2452141  1.0000000
```

```r
corrplot.mixed(correlation)
```



The couple of parameters that has the largest correlation (in absolute value) is alpha-gamma with a value of:

```
correlation[1,3]
```

```
## [1] 0.8487753
```

5i answer)

Let's use the Markov chain to approximate the posterior predictive distribution of the length of a dugong with age of 20 years.

```
pred_age_20=rep(NA, iterations)
for(i in 1:iterations){
  pred_age_20[i]=rnorm(1, alpha[i] - beta[i]*gamma[i]^20, sqrt(tau_square[i]))
}
head(pred_age_20,15)
```

```
##  [1] 2.111253 2.679360 2.641200 2.449258 2.316839 2.330715 2.615791
##  [8] 2.405395 2.568770 2.457307 2.569034 2.527328 2.318086 2.459438
## [15] 2.431486
```

A dugong with age of 20 years has approximately length of:

```
mean(pred_age_20)
```

```
## [1] 2.591154
```

5j answer)

Let's do the same thing, i.e. use the Markov chain to approximate the posterior predictive distribution of the length of a dugong with age of 30 years.

```
pred_age_30=rep(NA, iterations)
for(i in 1:iterations){
  pred_age_30[i]=rnorm(1, alpha[i] - beta[i]*gamma[i]^30, sqrt(tau_square[i]))
}
head(pred_age_30,15)
```

```
##  [1] 1.685278 2.487845 2.320534 2.387498 2.424514 2.372556 2.235253
##  [8] 2.433141 2.432766 2.631394 2.506701 2.463721 2.342649 2.261573
## [15] 2.401197
```

A dugong with age of 20 years has approximately length of:

```
mean(pred_age_30)
```

```
## [1] 2.635081
```

5k answer)

```
precision_20=1/var(pred_age_20)
precision_20
```

```
## [1] 101.1176
```

```
precision_30=1/var(pred_age_30)
precision_30
```

```
## [1] 86.84177
```

```
precision_20/precision_30>1
```

```
## [1] TRUE
```

We can conclude that the prediction for dugongs of 20 year is more precise than dugongs of 30 year.