Markov Chain Monte Carlo (MCMC) Analysis in the Epidemic Model

Shiyu Sun

2024-04-17

1 SIR Model - Metropolis-Hasting MCMC

We will be fitting our discrete-time SIR model to an epidemic dataset consisting of the first 20 days of data. The population from which this data was taken is of size $N = 10000$, with $I1 = 25$ and the remaining population wholly susceptible at the first time point at which data is recorded. Initially, we assume independent uniform priors for our parameters:

$$\beta \sim U[0, 0.5], \gamma \sim U[0, 0.5]$$

(i)

We would like to implement a random-walk Metropolis-Hastings (MH) algorithm for estimating the posterior distribution with the above data using a block update for β and γ, and thus tune the proposals in the algorithm so that the algorithm is relatively efficient. There are trace plots, as well as plots of the marginal and joint posterior distributions, and report posterior mean estimates and credible (e.g., percentile) intervals of the parameters.

```
options(width=60,length=200,digits=3)
data = read.csv("epidemic2024A3.csv",header = TRUE)
I_new = data$cases
R_new = data$removals
time = length(data$day)
N = 10000
S = c()
I = c()
R = c()
I[1] = 25
R[1] = 0
S[1] = N - I[1]
for (t in 1:time){
  S[t+1] = S[t] - I_new[t]
  I[t+1] = I[t] + I_new[t] - R_new[t]
  R[t+1] = R[t] + R_new[t]
}

# posterior distribution
post_dist = function(beta,gamma){
  ll = 1
  for (t in 1:time) {
  p1 = dbinom(I_new[t],size=S[t],prob = 1- exp(-beta * I[t]/N))
```

```
  p2 = dbinom(R_new[t],size=I[t],prob = 1- exp(-gamma))
  ll = ll*p1*p2
  }
  result = ll* dunif(beta,0,0.5)*dunif(gamma,0,0.5)
  return(result)
}
```

```
set.seed(11)
library(ggplot2)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ---- tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v lubridate 1.9.3     v tibble    3.2.1
## v purrr     1.0.2     v tidyr     1.3.1
## -- Conflicts ---------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
mh1 = function(n,beta0,gamma0,sigma){
  count =0
  beta = array(0,n)
  gamma = array(0,n)
  accept.prob = array(0,n)
  beta[1] = beta0
  gamma[1] = gamma0
  for (t in 2:n){
    #normal distribution as the proposal density
    y1 = rnorm(1,beta[t-1],sigma)
    y2= rnorm(1,gamma[t-1],sigma)
    #symmetric proposal density cancelled
    alpha = post_dist(y1,y2) / post_dist(beta[t-1],gamma[t-1])
    accept.prob[t] =alpha
    alpha = min(1,alpha)
    u = runif(1)
    if(u < alpha){
      beta[t] = y1
      gamma[t] = y2
      count=count+1
    }
    else{
      beta[t]= beta[t-1]
      gamma[t] = gamma[t-1]
    }
  }
  accept.rate = count/n
  x = cbind(beta,gamma,accept.rate,accept.prob)
  return (x)
}

n=10000
x = mh1(n,0.2,0.1,0.02)
```

```r
beta = x[,1]
gamma = x[,2]
accept.rate = unique(x[,3])
accept.prob = x[,4]

beta.mean = mean(beta)
gamma.mean = mean(gamma)

beta_perc = quantile(beta, probs = c(0.025, 0.975))
gamma_perc = quantile(gamma, probs = c(0.025, 0.975))
cat("The 95% confidence interval of beta is \n")
```

```
## The 95% confidence interval of beta is
```

```r
beta_perc
```

```
##  2.5% 97.5%
## 0.216 0.266
```

```r
cat("The 95% confidence interval of gamma is \n")
```

```
## The 95% confidence interval of gamma is
```

```r
gamma_perc
```

```
##  2.5% 97.5%
## 0.128 0.170
```

```r
cat("The posterior mean estimate of beta is", beta.mean,"\n")
```

```
## The posterior mean estimate of beta is 0.241
```

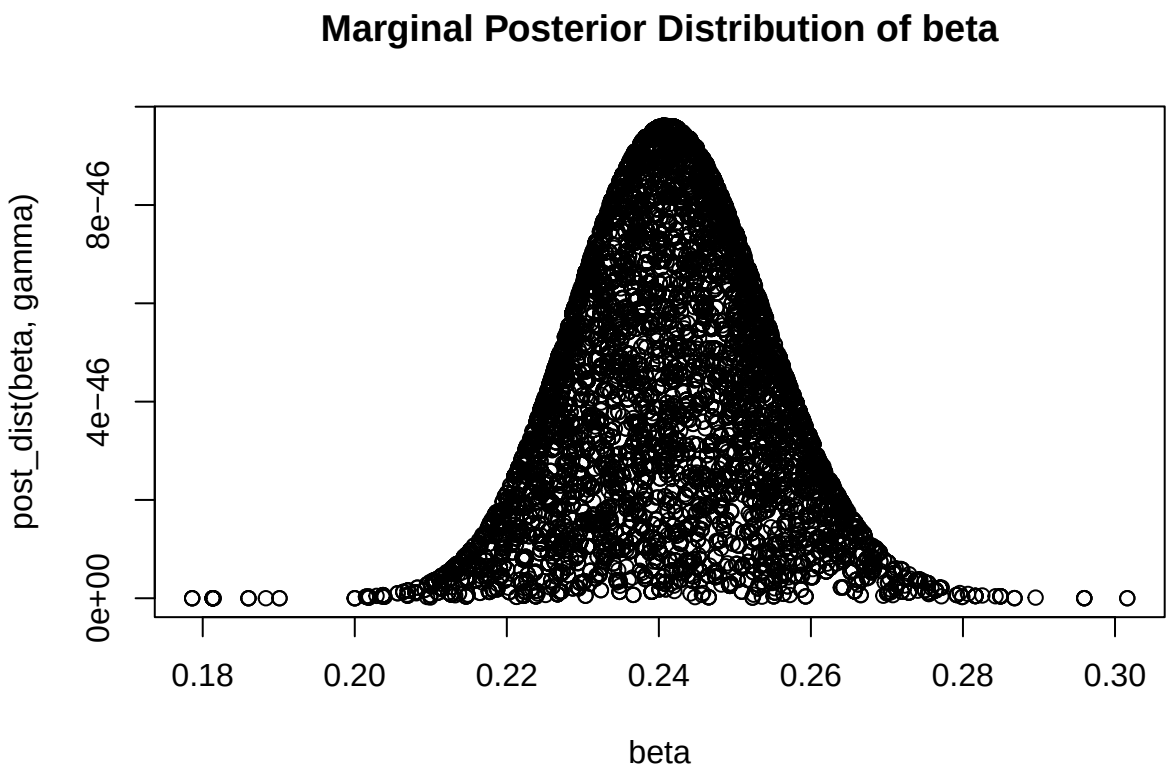
```r
cat("The posterior mean estimate of gamma is",gamma.mean,"\n")
```

```
## The posterior mean estimate of gamma is 0.148
```

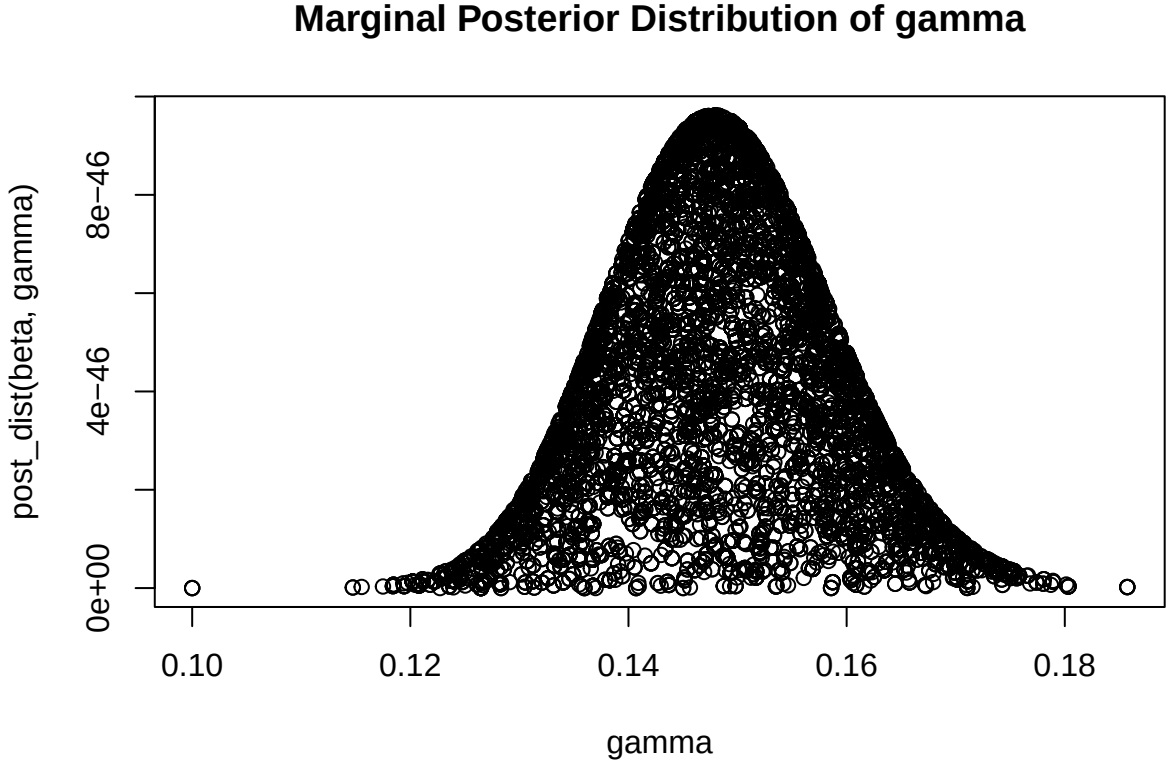
```r
cat("Acceptance rate is",accept.rate,"\n")
```

```
## Acceptance rate is 0.329
```

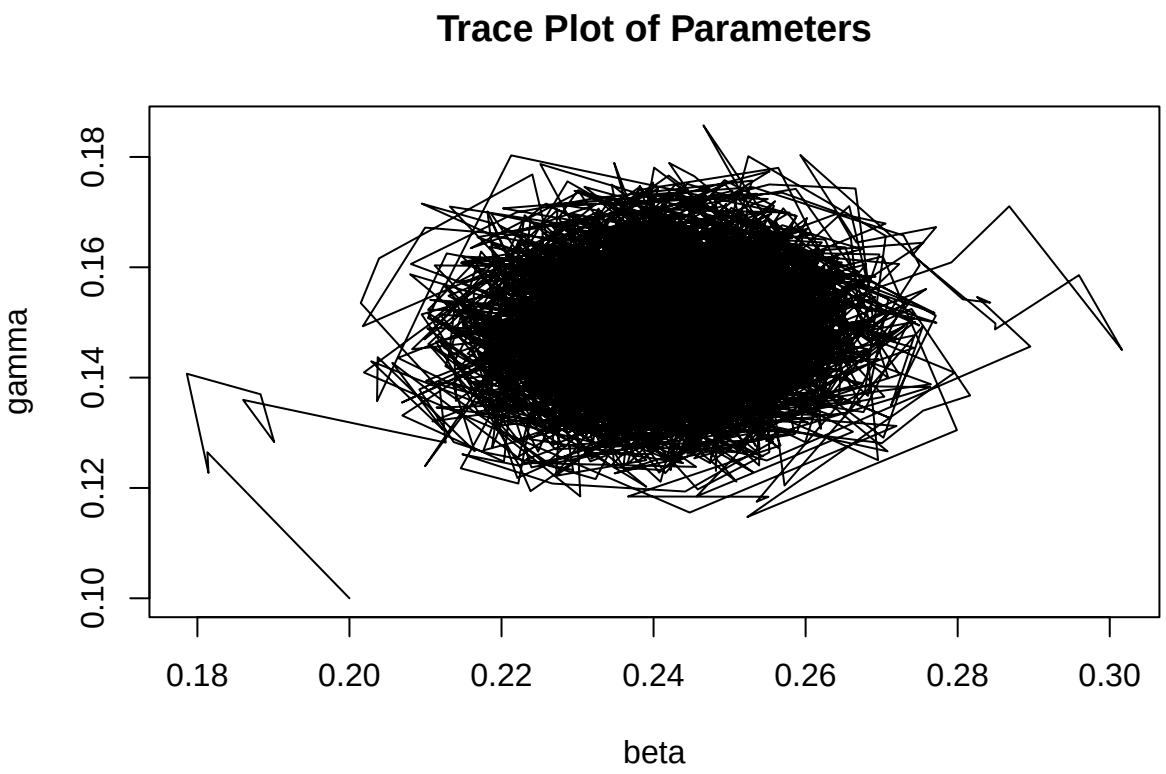
```r
#marginal distribution of beta and gamma
plot(beta,post_dist(beta,gamma), main="Marginal Posterior Distribution of beta")
```



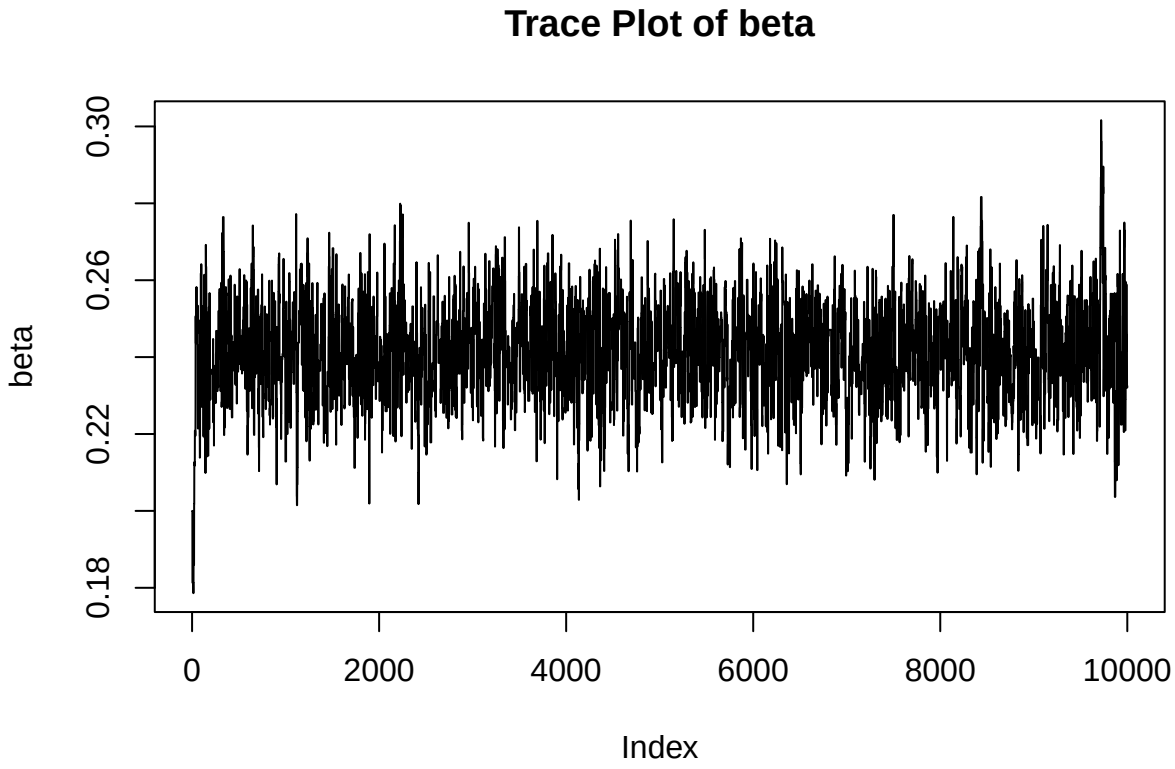

```
plot(gamma, post_dist(beta,gamma),main="Marginal Posterior Distribution of gamma")
```



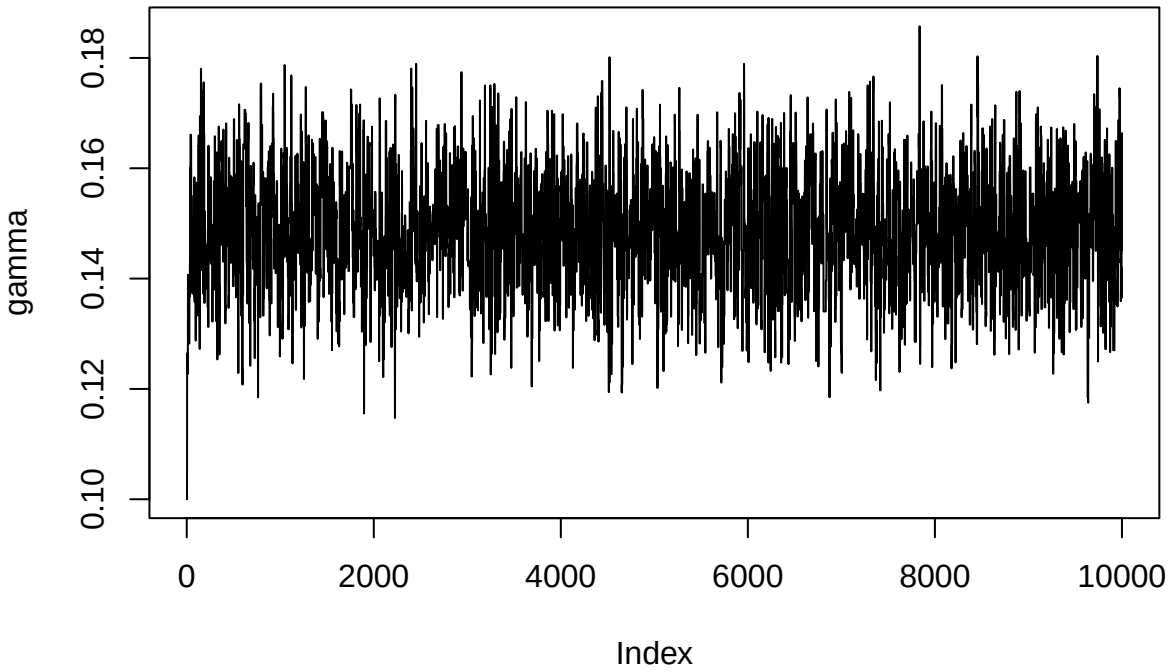

```
plot(x,type="l", main="Trace Plot of Parameters")
```




```
plot(beta,type="l",main="Trace Plot of beta")
```

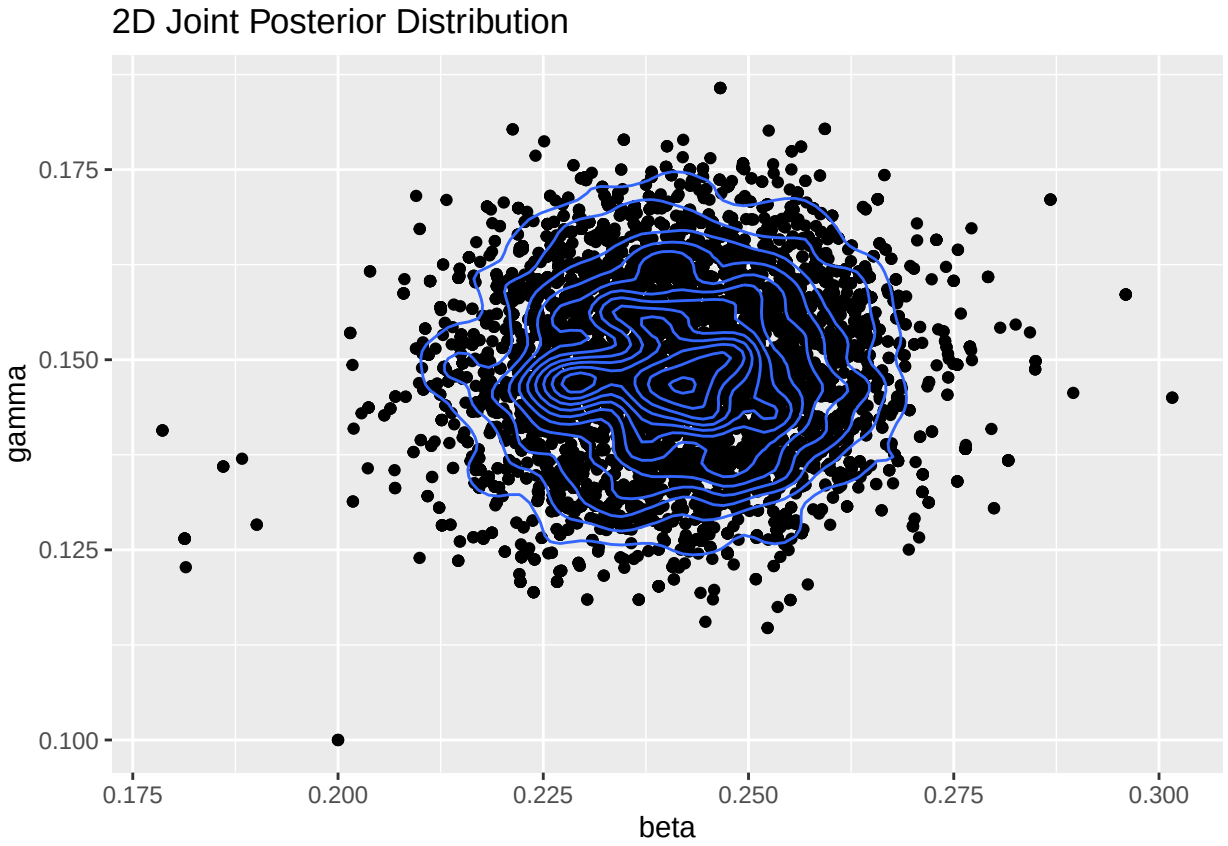



```
plot(gamma,type = "l",main="Trace Plot of gamma")
```




```
#joint posterior distribution of beta and gamma
data = data.frame(beta,gamma,post_dist(beta,gamma))

#joint posterior distribution plot in 2D plot
ggplot(data,aes(x=beta,y=gamma))+geom_point()+geom_density_2d()+ggtitle("2D Joint Posterior Distributio
```



As we can see from output, the posterior mean estimate for β is 0.241, and for γ is 0.148. The 95% percentile interval for β is (0.216,0.266), for γ is (0.128,0.170). Acceptance rate for β is 32.9%.

(ii)

Now we change the single-parameter independent sampler updates to make the MCMC algorithm efficiently. Trace plots, as well as plots of the marginal and joint posterior distributions, and report posterior mean estimates and credible (e.g., percentile) intervals of the parameters are included.

```
set.seed(12)
mh2 = function(n,beta0,gamma0){
  count1 = 0
  count2 = 0
  beta = array(0,n)
  gamma = array(0,n)
  beta[1] = beta0
  gamma[1] = gamma0
  for (t in 2:n){
    #block1
    #prior distribution (~u(0,0.5)) as the proposal distribution
    y.beta = runif(1,0,0.5)
    num1 = post_dist(y.beta,gamma[t-1])*runif(1,0,0.5)
    den1 = post_dist(beta[t-1],gamma[t-1])*runif(1,0,0.5)
    alpha1 = num1/den1
    alpha1 = min(1,alpha1)
    u1 = runif(1)

    if(u1 < alpha1){
```

```
      beta[t] = y.beta
      count1 = count1+1
    }
    else{
      beta[t]= beta[t-1]
      }
    #block 2
    y.gamma = runif(1,0,0.5)
    num2 = post_dist(beta[t],y.gamma)*runif(1,0,0.5)
    den2 = post_dist(beta[t],gamma[t-1])*runif(1,0,0.5)
    alpha2 = num2/den2
    alpha2 = min(1,alpha2)
    u2 = runif(1)

    if(u2 <alpha2){
      gamma[t] = y.gamma
      count2 = count2 +1
    }
    else{
      gamma[t]= gamma[t-1]
      }
  }

  accept.rate.beta = count1/n
  accept.rate.gamma = count2/n
  x = cbind(beta,gamma,accept.rate.beta,accept.rate.gamma)
  return (x)
}


x=mh2(10000,0.2,0.1)

beta = x[,1]
gamma = x[,2]
accept.rate.beta = unique(x[,3])
accept.rate.gamma = unique(x[,4])


beta.mean = mean(beta)
gamma.mean = mean(gamma)

beta_perc = quantile(beta, probs = c(0.025, 0.975))
gamma_perc = quantile(gamma, probs = c(0.025, 0.975))
cat("The 95% confidence interval of beta is \n")
```

```
## The 95% confidence interval of beta is
```

```
beta_perc
```

```
##  2.5% 97.5%
## 0.214 0.271
```

```
cat("The 95% confidence interval of gamma is \n")
```

```
## The 95% confidence interval of gamma is
```

```
gamma_perc
```

```
##  2.5% 97.5%
## 0.127 0.172
```

```
cat("The posterior mean estimate of beta is", beta.mean,"\n")
```

```
## The posterior mean estimate of beta is 0.242
```

```
cat("The posterior mean estimate of gamma is",gamma.mean,"\n")
```

```
## The posterior mean estimate of gamma is 0.149
```

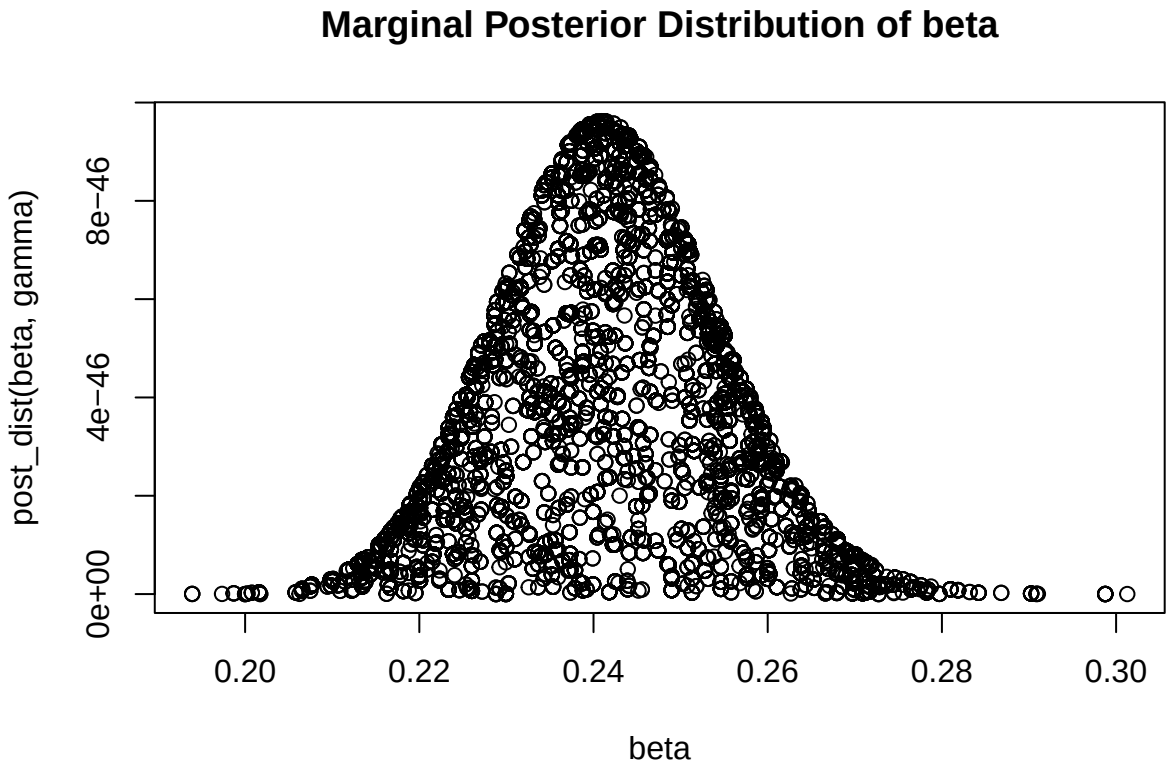
```
cat("Acceptance rate of beta is",accept.rate.beta,"\n")
```

```
## Acceptance rate of beta is 0.0813
```

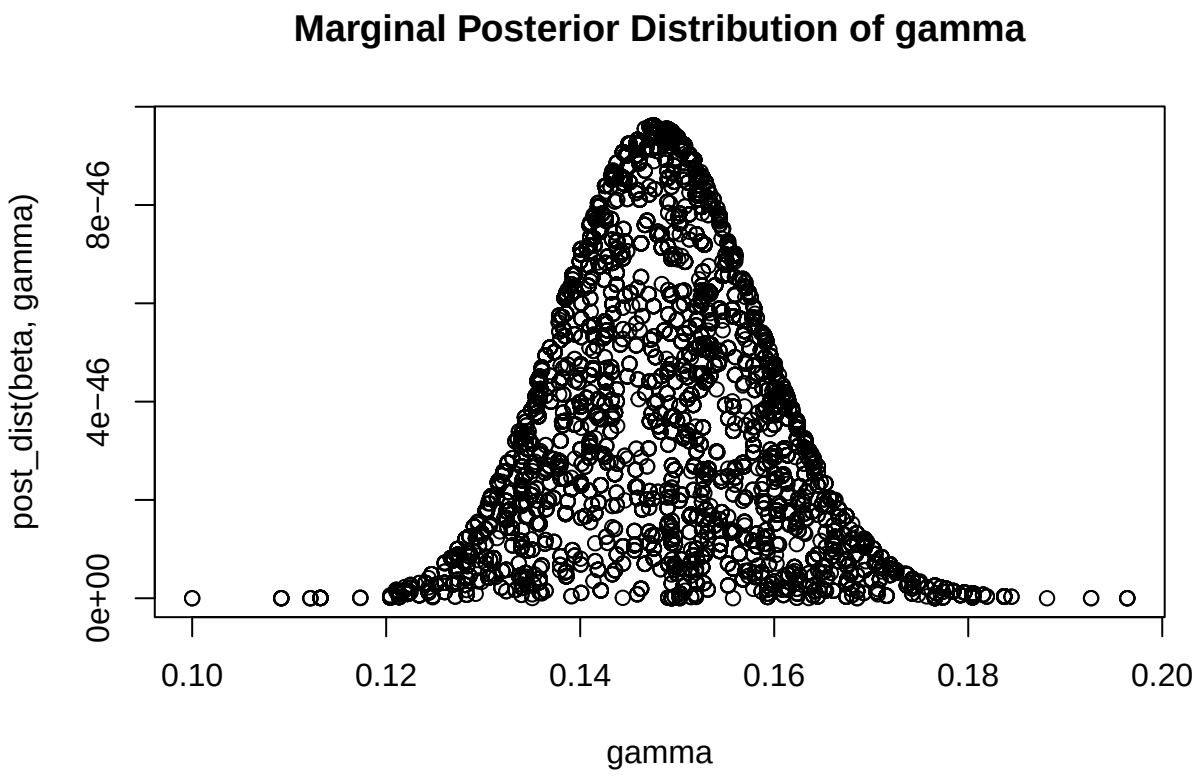
```
cat("Acceptance rate of gamma is",accept.rate.gamma,"\n")
```

```
## Acceptance rate of gamma is 0.0641
```

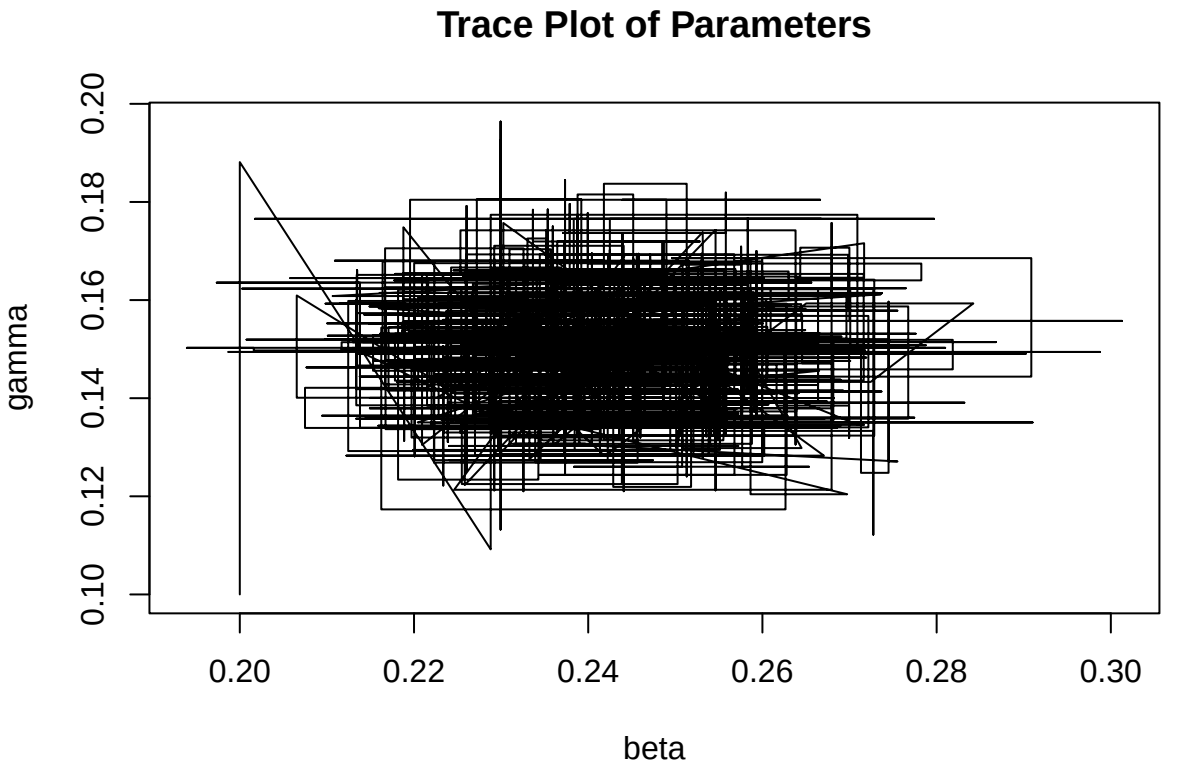
```
#marginal distribution of beta and gamma
plot(beta,post_dist(beta,gamma), main="Marginal Posterior Distribution of beta")
```



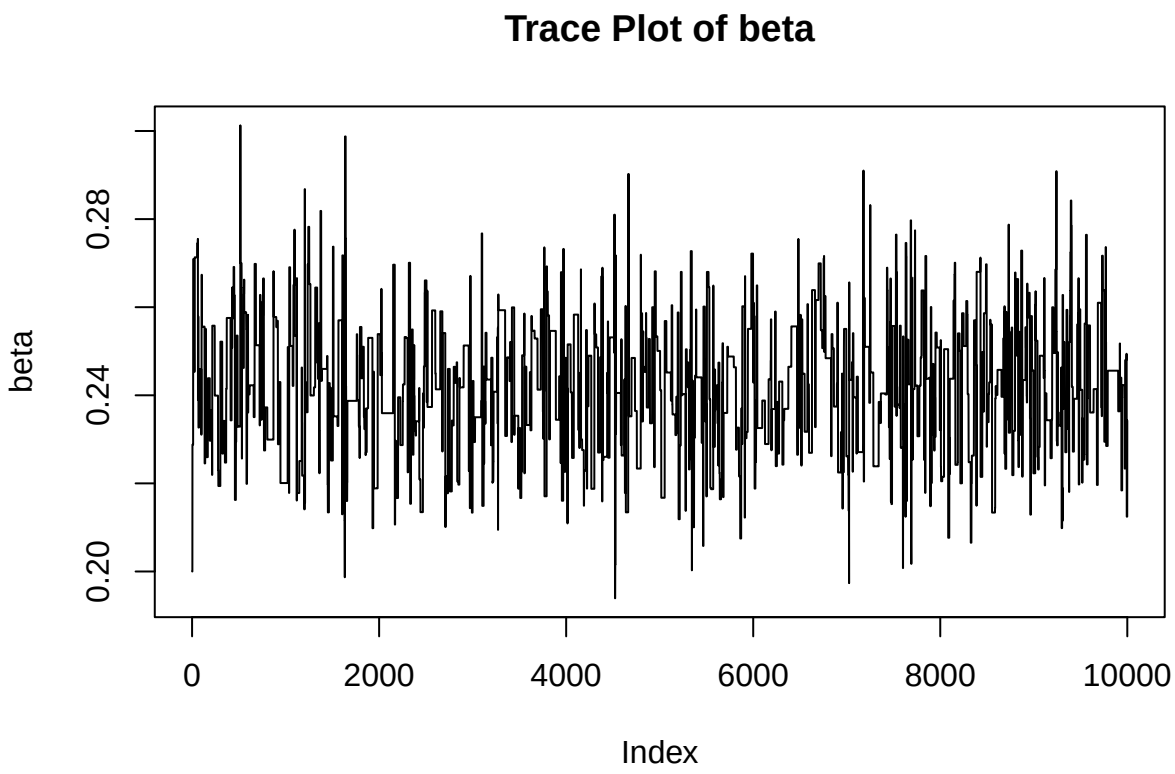

```
plot(gamma, post_dist(beta,gamma),main="Marginal Posterior Distribution of gamma")
```



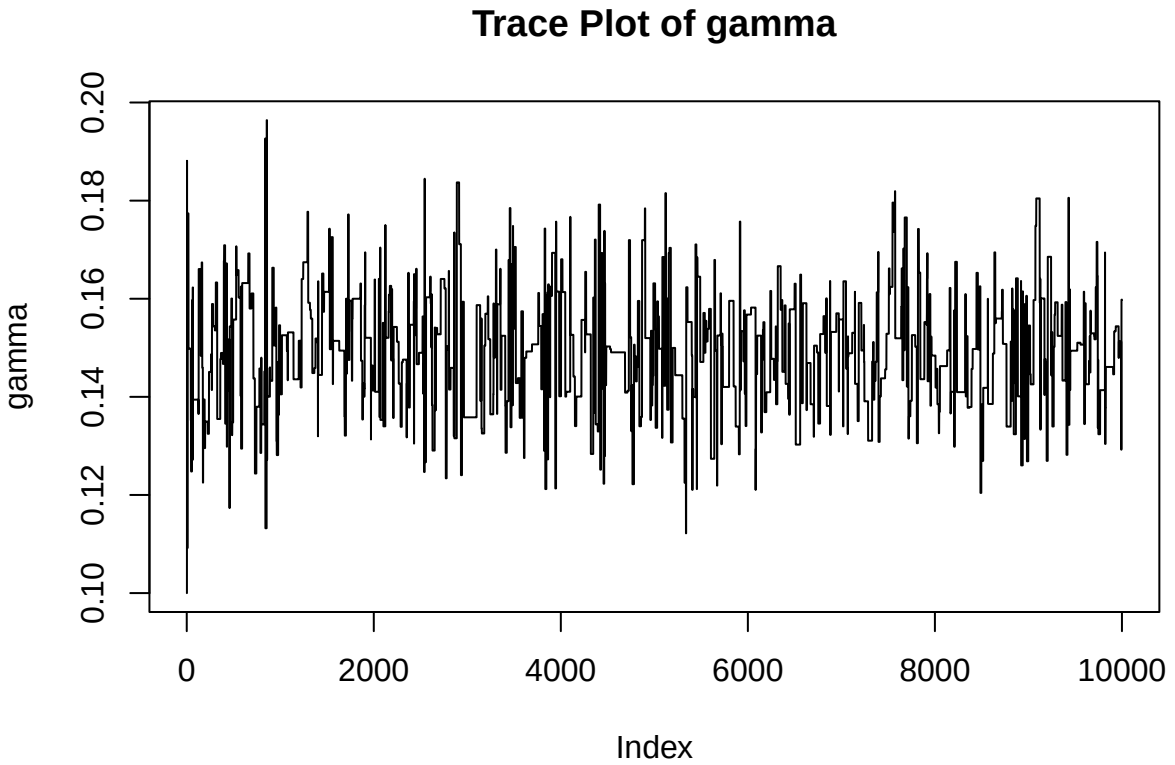

```
plot(x,type="l", main="Trace Plot of Parameters")
```




```
plot(beta,type="l",main="Trace Plot of beta")
```



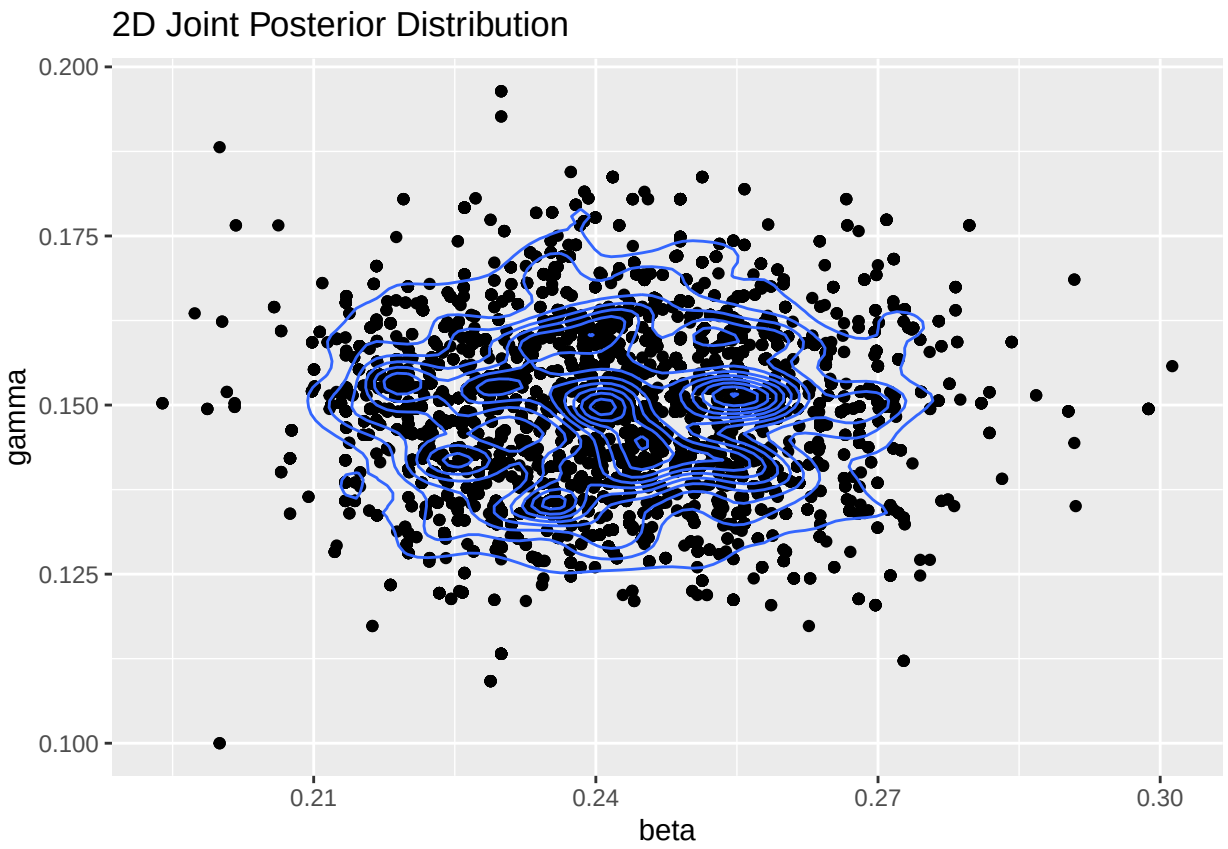

```
plot(gamma,type = "l",main="Trace Plot of gamma")
```




```
#joint posterior distribution of beta and gamma
data = data.frame(beta,gamma,post_dist(beta,gamma))

#joint posterior distribution plot in 2D plot
```

```
ggplot(data,aes(x=beta,y=gamma))+ geom_point()+
  geom_density_2d()+ggtitle("2D Joint Posterior Distribution")
```



The posterior mean estimate for β is 0.242, and for γ is 0.149. The 95% percentile interval for β is (0.214,0.271), for γ is (0.127,0.172). Acceptance rate for β and γ are 8.13% and 6.41% respectively.

(iii)

```
set.seed(13)
mh3 = function(n,beta0,gamma0,sigma1,sigma2){
  count1 = 0
  count2 = 0
  beta = array(0,n)
  gamma = array(0,n)
  beta[1] = beta0
  gamma[1] = gamma0
  for (t in 2:n){
    #block1
    y.beta = rnorm(1,0.24,sigma1)
    #numerator
    num1 = post_dist(y.beta,gamma[t-1])*dnorm(beta[t-1],0.24,sigma1)
    #denominator
    den1 = post_dist(beta[t-1],gamma[t-1])*dnorm(y.beta,0.24,sigma1)
    alpha1 = num1/den1
    alpha1 = min(1,alpha1)
    u1 = runif(1)
```

```
    if(u1 < alpha1){
      beta[t] = y.beta
      count1 = count1+1
    }
    else{
      beta[t]= beta[t-1]
      }
    #block 2
    y.gamma = rnorm(1,0.15,sigma2)
    #numerator
    num2 = post_dist(beta[t],y.gamma)*dnorm(gamma[t-1],0.15,sigma2)
    #denominator
    den2 = post_dist(beta[t],gamma[t-1])*dnorm(y.gamma,0.15,sigma2)
    alpha2 = num2/den2
    alpha2 = min(1,alpha2)
    u2 = runif(1)
    if(u2 <alpha2){
      gamma[t] = y.gamma
      count2 = count2+1
    }
    else{
      gamma[t]= gamma[t-1]
      }
  }

  accept.rate.beta = count1/n
  accept.rate.gamma = count2/n
  x = cbind(beta,gamma,accept.rate.beta,accept.rate.gamma)
  return (x)
}

#sigma=0.01
x=mh3(10000,0.2,0.1,0.01,0.01)

beta = x[,1]
gamma = x[,2]
accept.rate.beta = unique(x[,3])
accept.rate.gamma = unique(x[,4])


beta.mean = mean(beta)
gamma.mean = mean(gamma)

beta_perc = quantile(beta, probs = c(0.025, 0.975))
gamma_perc = quantile(gamma, probs = c(0.025, 0.975))
cat("The 95% confidence interval of beta is \n")
```

```
## The 95% confidence interval of beta is
```

```
beta_perc
```

```
##  2.5% 97.5%
## 0.217 0.267
```

```r
cat("The 95% confidence interval of gamma is \n")
```

```
## The 95% confidence interval of gamma is
```

```r
gamma_perc
```

```
##  2.5% 97.5%
## 0.129 0.169
```

```r
cat("The posterior mean estimate of beta is", beta.mean,"\n")
```

```
## The posterior mean estimate of beta is 0.241
```

```r
cat("The posterior mean estimate of gamma is",gamma.mean,"\n")
```

```
## The posterior mean estimate of gamma is 0.148
```

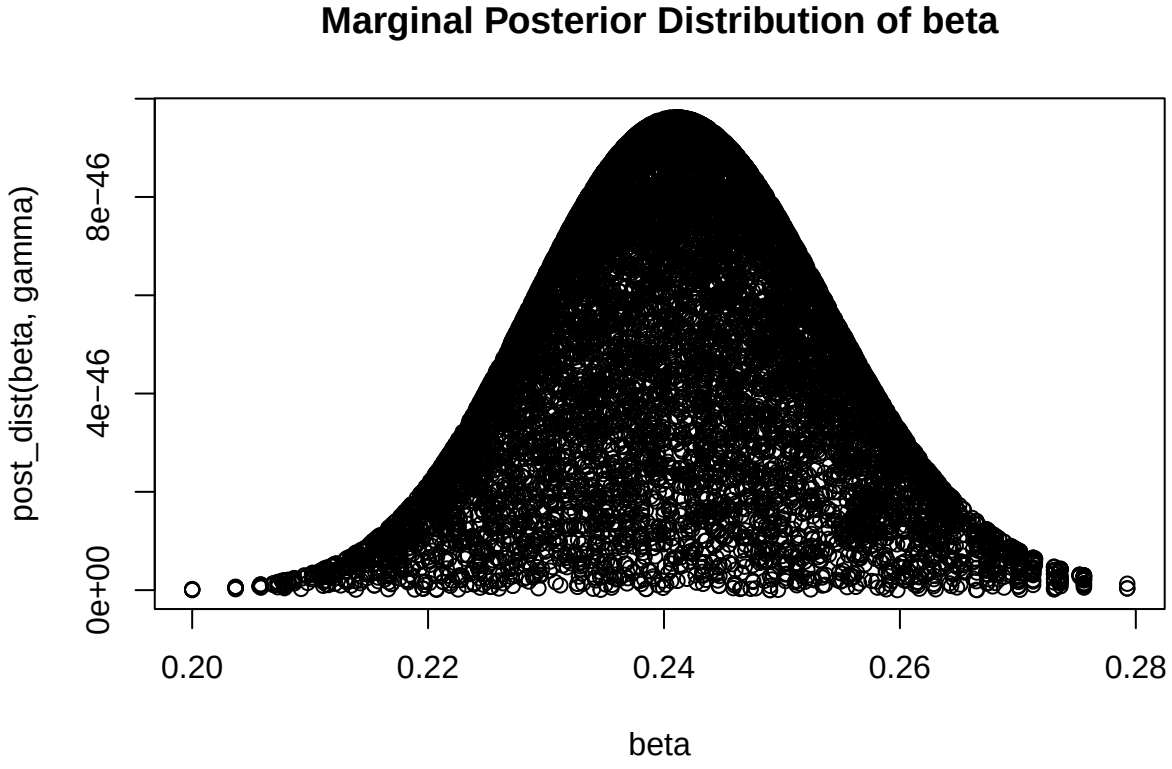
```r
cat("Acceptance rate of beta is",accept.rate.beta,"\n")
```

```
## Acceptance rate of beta is 0.846
```

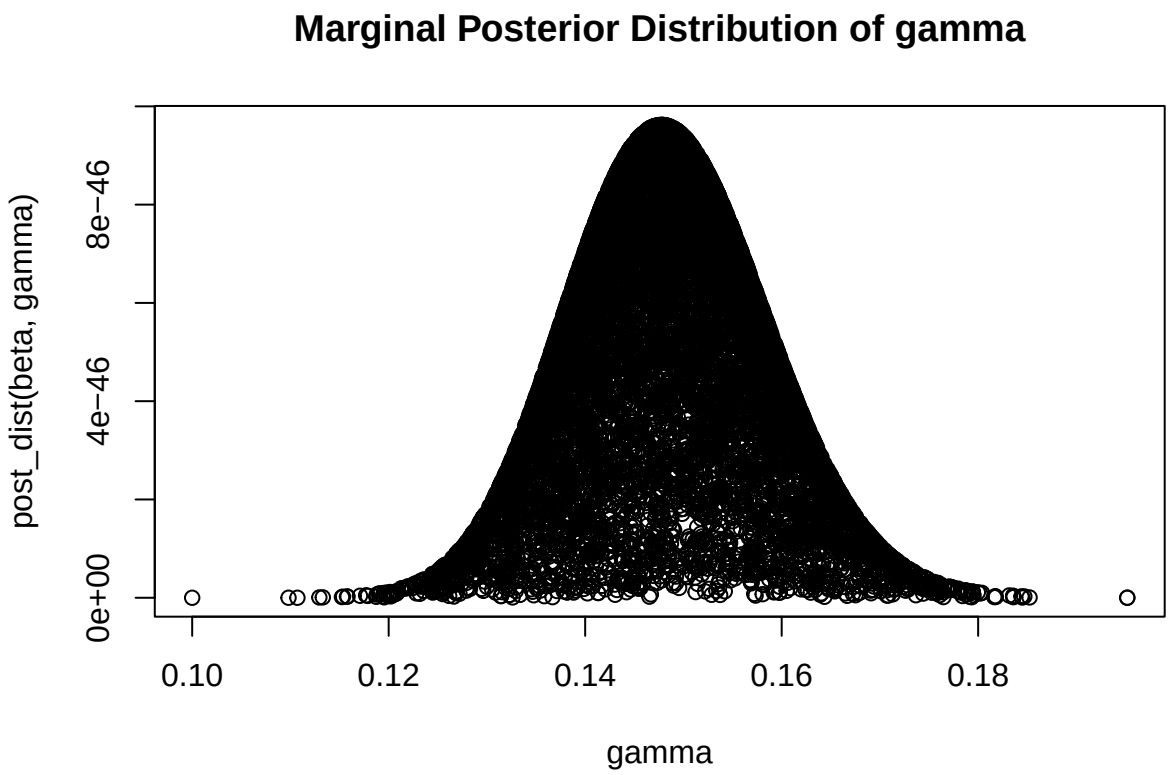
```r
cat("Acceptance rate of gamma is",accept.rate.gamma,"\n")
```

```
## Acceptance rate of gamma is 0.901
```

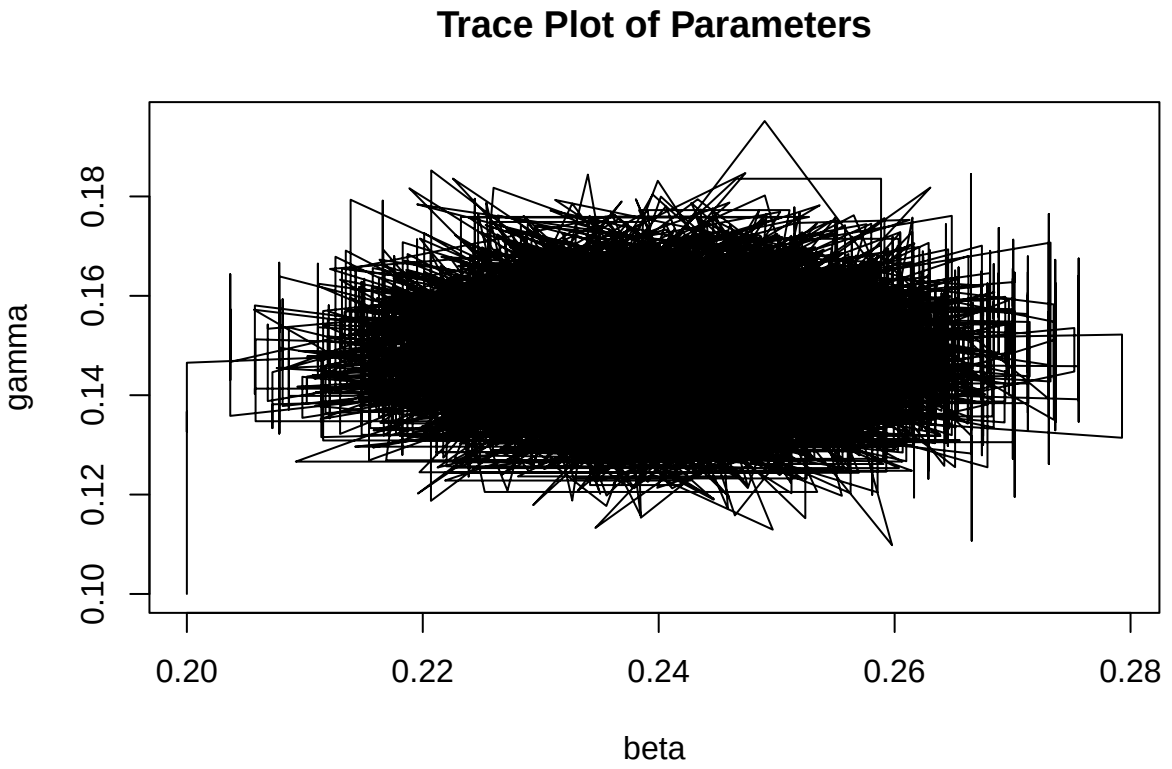
```r
#marginal distribution of beta and gamma
plot(beta,post_dist(beta,gamma), main="Marginal Posterior Distribution of beta")
```



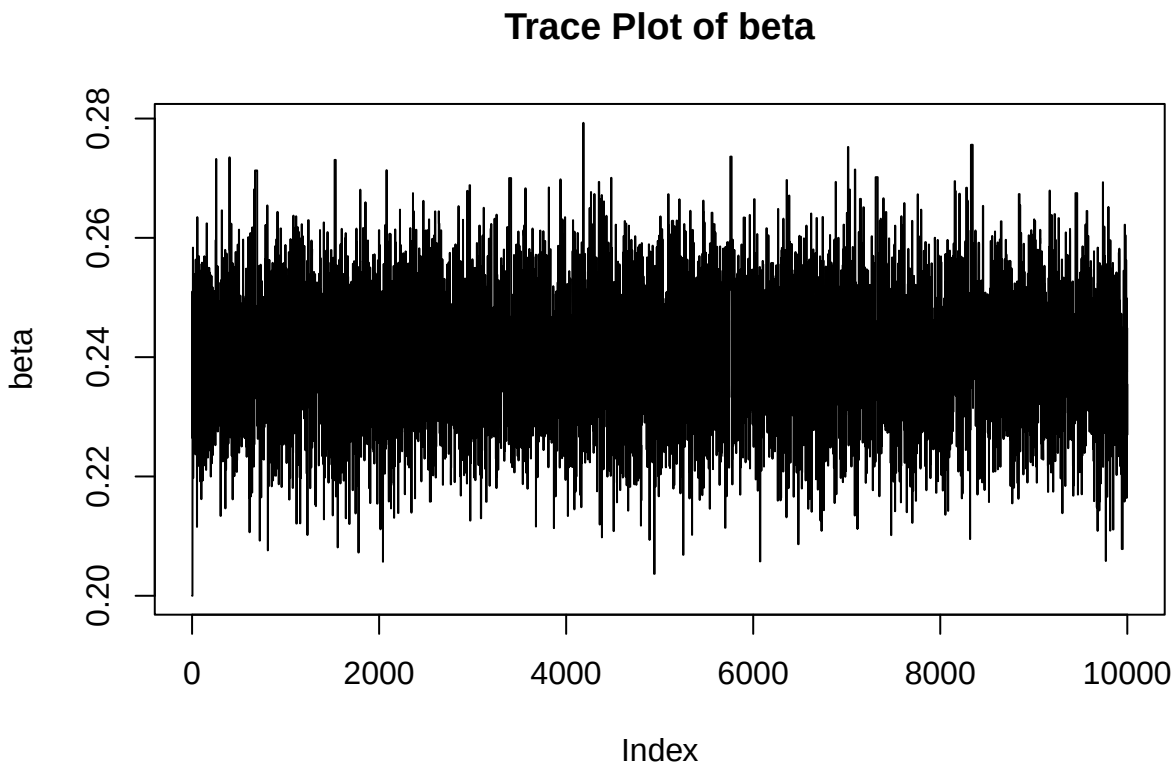
```
plot(gamma, post_dist(beta,gamma),main="Marginal Posterior Distribution of gamma")
```



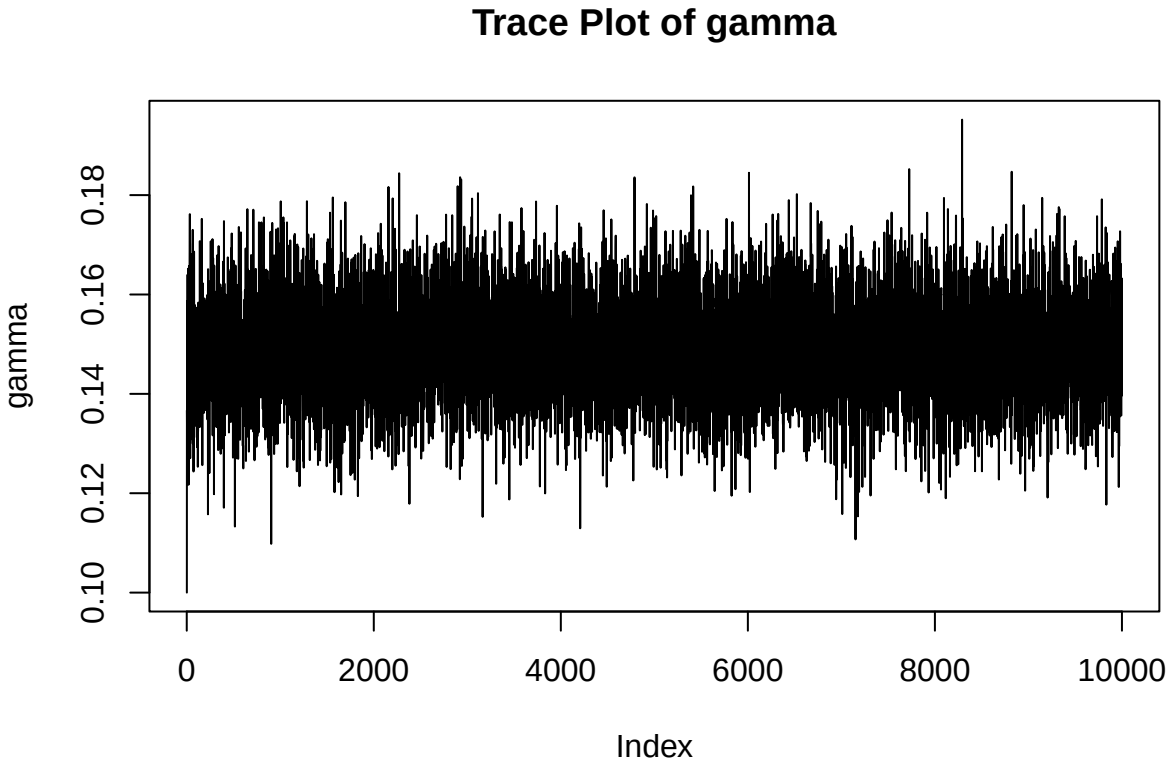

```
plot(x,type="l", main="Trace Plot of Parameters")
```



```r
plot(beta,type="l",main="Trace Plot of beta")
```

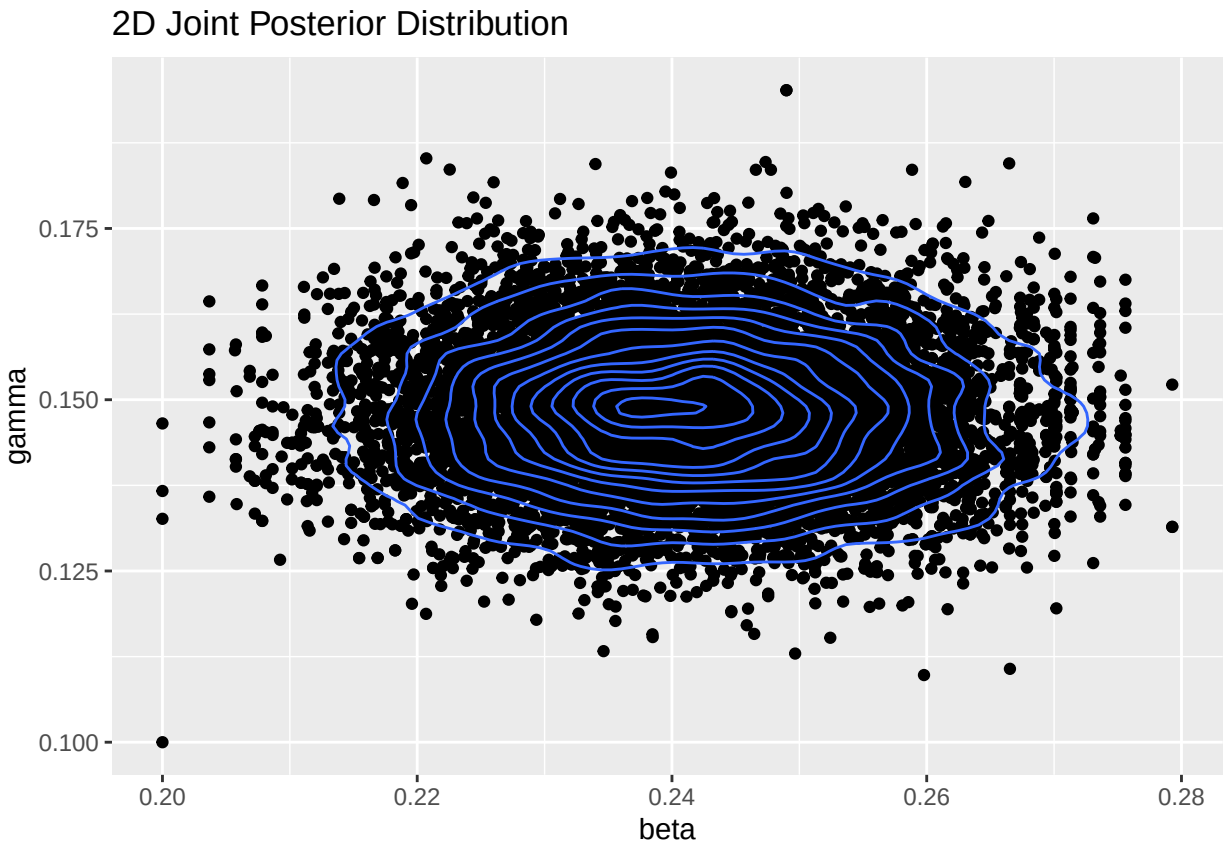



```r
plot(gamma,type = "l",main="Trace Plot of gamma")
```



```r
#joint posterior distribution of beta and gamma
data = data.frame(beta,gamma,post_dist(beta,gamma))

#joint posterior distribution plot in 2D plot
ggplot(data,aes(x=beta,y=gamma))+ geom_point()+
geom_density_2d()+ggtitle("2D Joint Posterior Distribution")
```



The posterior mean estimate for β is 0.241, and for γ is 0.148. The 95% percentile interval for β is (0.217,0.267), and for γ is (0.129,0.169). Acceptance rate for β is 84.6%, and for γ is 90.1%.

From the acceptance probability rates, we can see that model in (iii) is more efficient than the model in (ii). Also trace plot of β and γ from (iii) reached a state of convergence (stationarity) much more quickly than (ii), and more like a hairy caterpillar compared to (ii). Number of rejection is much lower and the burn-in period is not long.

2 SIR Model - Data-augmented MCMC

Imagine that two of the data points we have in our data set were not observed. Specifically, we shall assume that we do not know the number of new cases on day 7, and the number of removals on day 14.

We will implement data augmented MCMC to sample from the joint posterior of the model parameters and the two missing data points above. We will assume we have no prior information about the two missing data points.

Also, below are trace plots of the model and missing data parameters, and summarize both numerically and graphically information about the four marginal posterior distributions for our model and missing data parameters.

```r
set.seed(21)
library(R.utils)
```

```
## Loading required package: R.oo
```

```
## Loading required package: R.methodsS3
```

```
## R.methodsS3 v1.8.2 (2022-06-13 22:00:14 UTC) successfully loaded. See ?R.methodsS3 for help.
```

```
## R.oo v1.26.0 (2024-01-24 05:12:50 UTC) successfully loaded. See ?R.oo for help.
```

```
##
## Attaching package: 'R.oo'
```

```
## The following object is masked from 'package:R.methodsS3':
##
##     throw
```

```
## The following objects are masked from 'package:methods':
##
##     getClasses, getMethods
```

```
## The following objects are masked from 'package:base':
##
##     attach, detach, load, save
```

```
## R.utils v2.12.3 (2023-11-18 01:00:02 UTC) successfully loaded. See ?R.utils for help.
```

```
##
## Attaching package: 'R.utils'
```

```
## The following object is masked from 'package:tidyr':
##
##     extract
```

```
## The following object is masked from 'package:utils':
##
##     timestamp
```

```
## The following objects are masked from 'package:base':
##
##     cat, commandArgs, getOption, isOpen, nullfile,
##     parse, use, warnings
```

```r
data = read.csv("epidemic2024A3.csv",header = TRUE)
I_new = data$cases
R_new = data$removals

#7th new case, and 14th new removal from the original data set are missing
I_new = I_new[-7]
R_new = R_new[-14]

#joint posterior distribution of model parameters and missing data
joint_dist = function(beta,gamma,I7,R14){
  #I7,R14 are the proposal values to be inserted
  I_new = insert(I_new,ats=7,value = I7)
  R_new = insert(R_new,ats=14,value = R14)
  time = length(I_new)
  N = 10000
  S = c()
```

```
  I = c()
  R = c()
  I[1] = 25
  R[1] = 0
  S[1] = N - I[1]
  for (t in 1:time){
  S[t+1] = S[t] - I_new[t]
  I[t+1] = I[t] + I_new[t] - R_new[t]
  R[t+1] = R[t] + R_new[t]
  }

  ll = 1
  for (t in 1:time) {
  p1 = dbinom(I_new[t],size=S[t],prob = 1- exp(-beta * I[t]/N))
  p2 = dbinom(R_new[t],size=I[t],prob = 1- exp(-gamma))
  ll = ll*p1*p2
  }
  #calculate likelihood
  result = ll* dunif(beta,0,0.5)*dunif(gamma,0,0.5)
  return(result)
}


#data augmented MCMC
mh_aug = function(n,beta0,gamma0,I7_0,R14_0,sigma1,sigma2){
  count =0
  beta = array(0,n)
  gamma = array(0,n)
  miss1 = array(0,n)
  miss2 = array(0,n)
  beta[1] = beta0
  gamma[1] = gamma0
  miss1[1] = I7_0
  miss2[1] = R14_0
  for (t in 2:n){
    #normal distribution as the proposal density of parameters
    y1 = rnorm(1,beta[t-1],sigma1)
    y2= rnorm(1,gamma[t-1],sigma1)
    #missing data need to be integers
    y.miss1 = round(rnorm(1,miss1[t-1],sigma2))
    y.miss2 = round(rnorm(1,miss2[t-1],sigma2))
    #symmetric proposal density (normal) is cancelled
    alpha = joint_dist(y1,y2,y.miss1,y.miss2)/joint_dist(beta[t-1],gamma[t-1],miss1[t-1],miss2[t-1])
    alpha = min(1,alpha)
    u = runif(1)
    if(u < alpha){
      beta[t] = y1
      gamma[t] = y2
      miss1[t] = y.miss1
      miss2[t] = y.miss2
      count=count+1
    }
    else{
```

```
      beta[t]= beta[t-1]
      gamma[t] = gamma[t-1]
      miss1[t] = miss1[t-1]
      miss2[t] = miss2[t-1]
    }
  }
  accept.rate = count/n
  x = cbind(beta,gamma,miss1,miss2,accept.rate)
  return (x)
}

n=10000
x=mh_aug(n,0.2,0.1,10,10,0.01,1)
beta = x[,1]
gamma = x[,2]
I7 = x[,3]
R14 = x[,4]
accept.rate = unique(x[,5])
joint = NULL

#joint posterior probability up to proportionality
for (i in 1:n){
  joint[i] = joint_dist(beta[i],gamma[i],I7[i],R14[i])
}

#numerical information of 4 parameters
cat("The mean of estimates for beta is",round(mean(beta),3),"\n")
```

```
## The mean of estimates for beta is 0.243
```

```
cat("The mean of estimates for gamma is",round(mean(gamma),3),"\n")
```

```
## The mean of estimates for gamma is 0.15
```

```
cat("The mean of estimates for I7 is",round(mean(I7)),"\n")
```

```
## The mean of estimates for I7 is 13
```

```
cat("The mean of estimates for R14 is",round(mean(R14)),"\n")
```

```
## The mean of estimates for R14 is 15
```

```
cat("The acceptance rate is",accept.rate,"\n")
```

```
## The acceptance rate is 0.553
```

```
cat("The 95% interval for beta is (",quantile(beta,c(0.025,0.975))[1],",",quantile(beta,c(0.025,0.975))
```

```
## The 95% interval for beta is ( 0.217 , 0.27 )
```

```
cat("The 95% interval for gamma is (",quantile(gamma,c(0.025,0.975))[1],",",quantile(gamma,c(0.025,0.975
```

```
## The 95% interval for gamma is ( 0.128 , 0.175 )
```

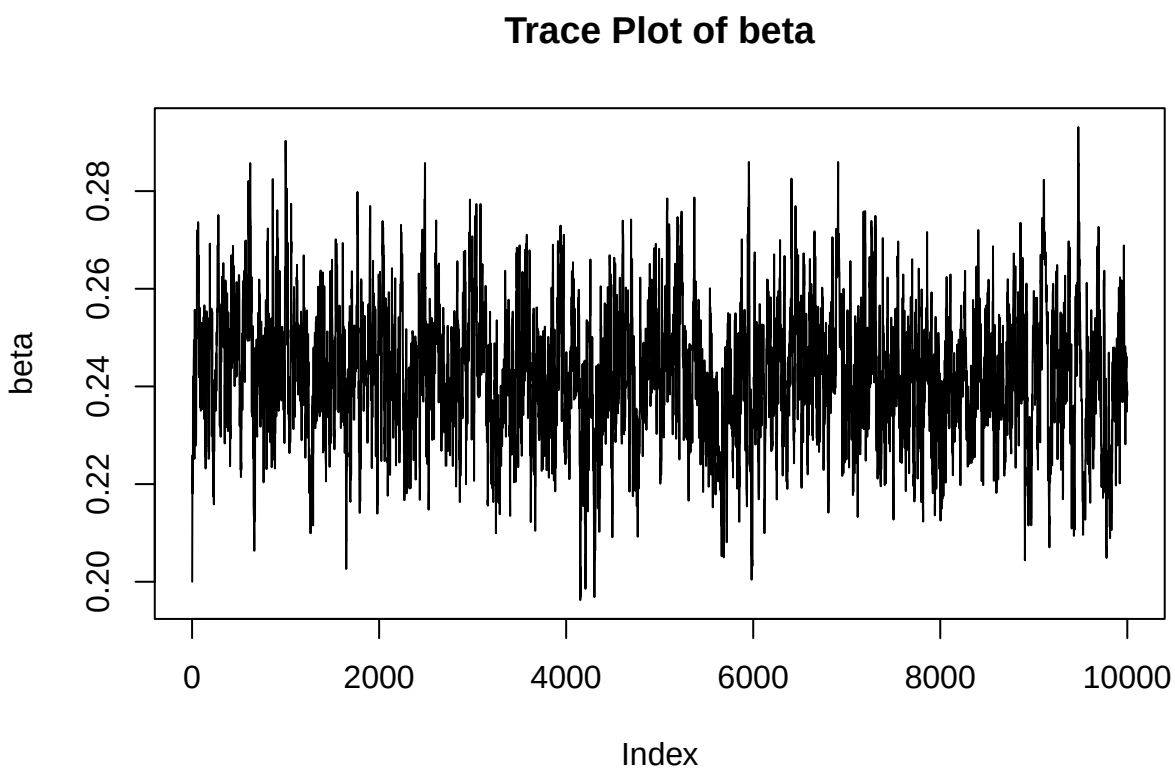
```
cat("The 95% interval for I7 is (",quantile(I7,c(0.025,0.975))[1],",",quantile(I7,c(0.025,0.975))[2],")"
```

```
## The 95% interval for I7 is ( 7 , 19 )
```

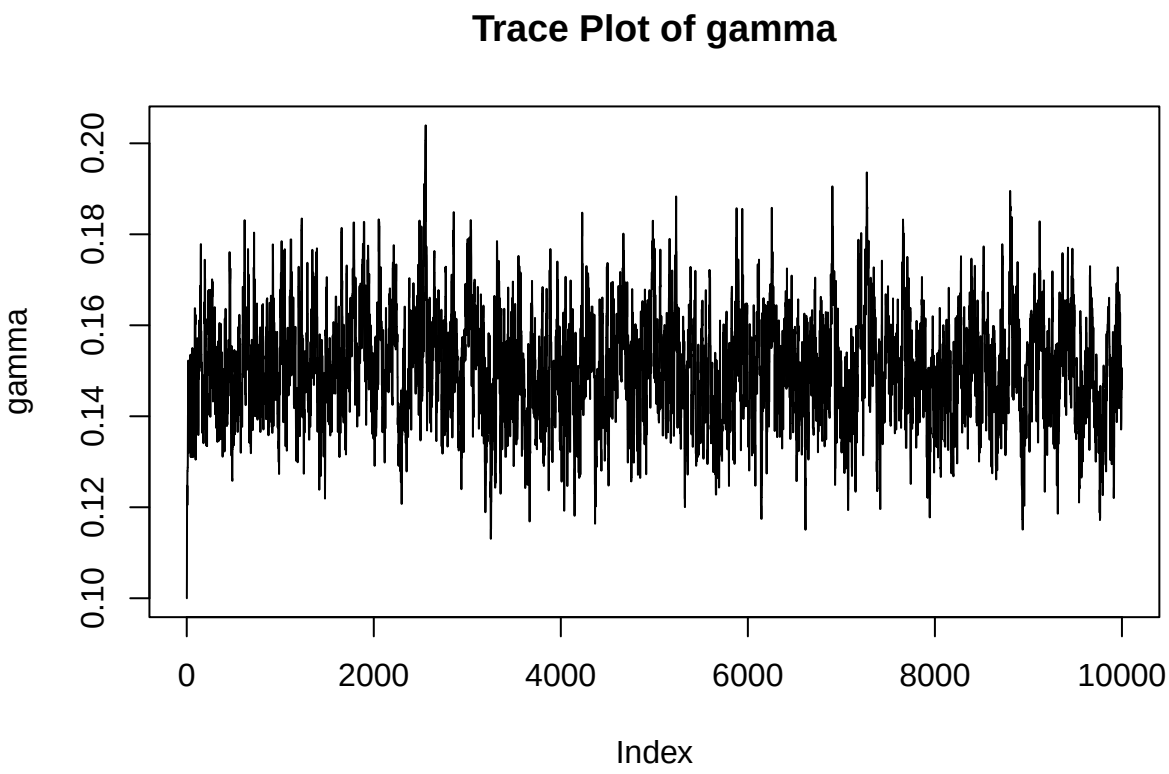
```r
cat("The 95% interval for R14 is (",quantile(R14,c(0.025,0.975))[1],",",quantile(R14,c(0.025,0.975))[2]
```

```
## The 95% interval for R14 is ( 8 , 21 )
```

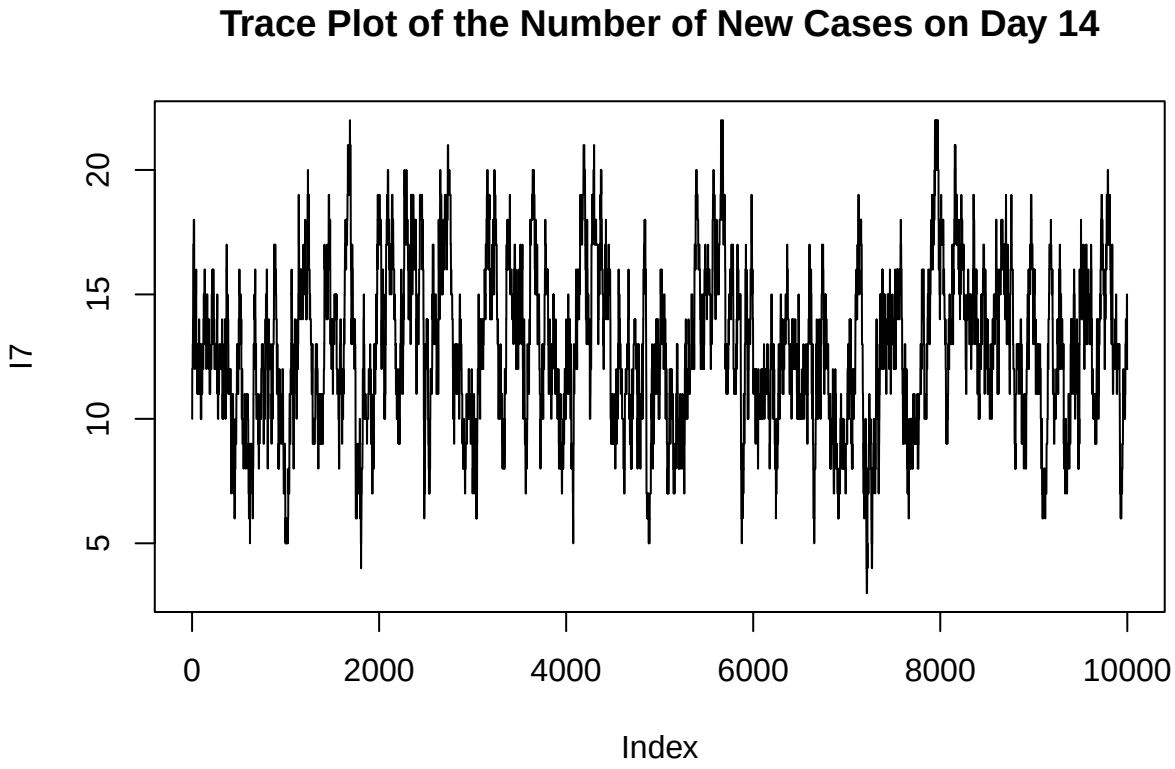
```r
#trace plots
plot(beta,type = "l", main = "Trace Plot of beta")
```



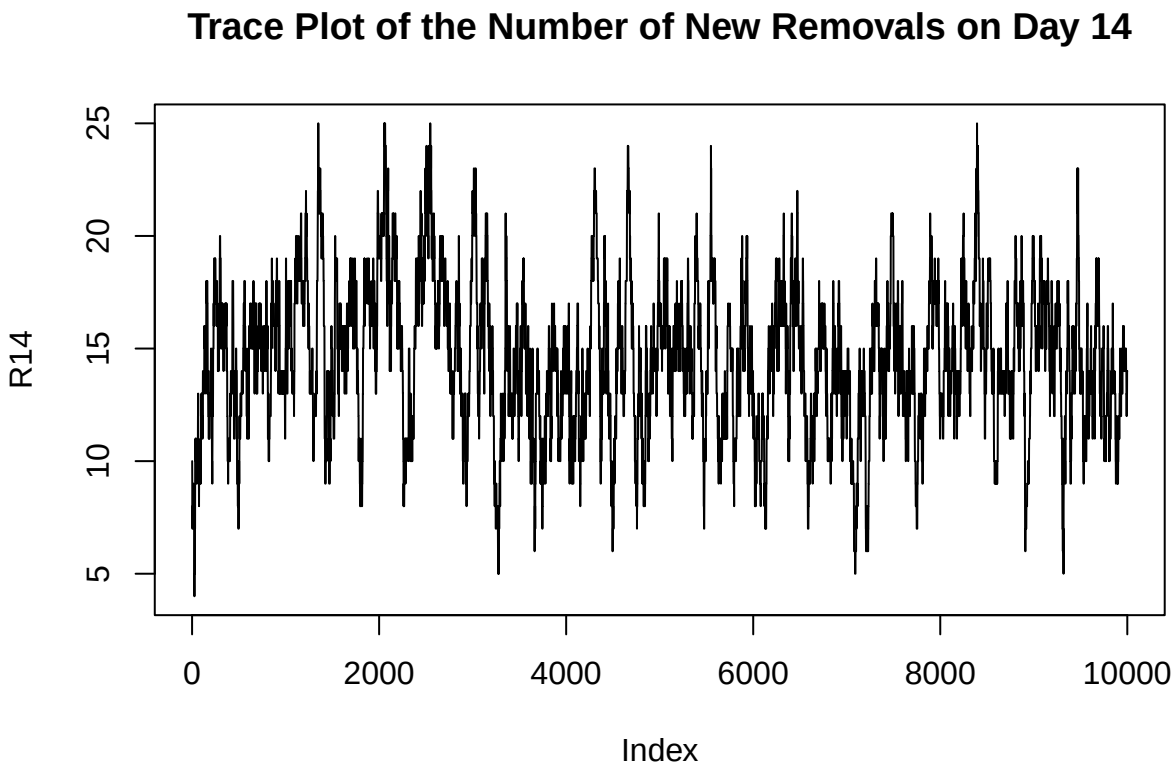

```r
plot(gamma,type = "l", main = "Trace Plot of gamma")
```



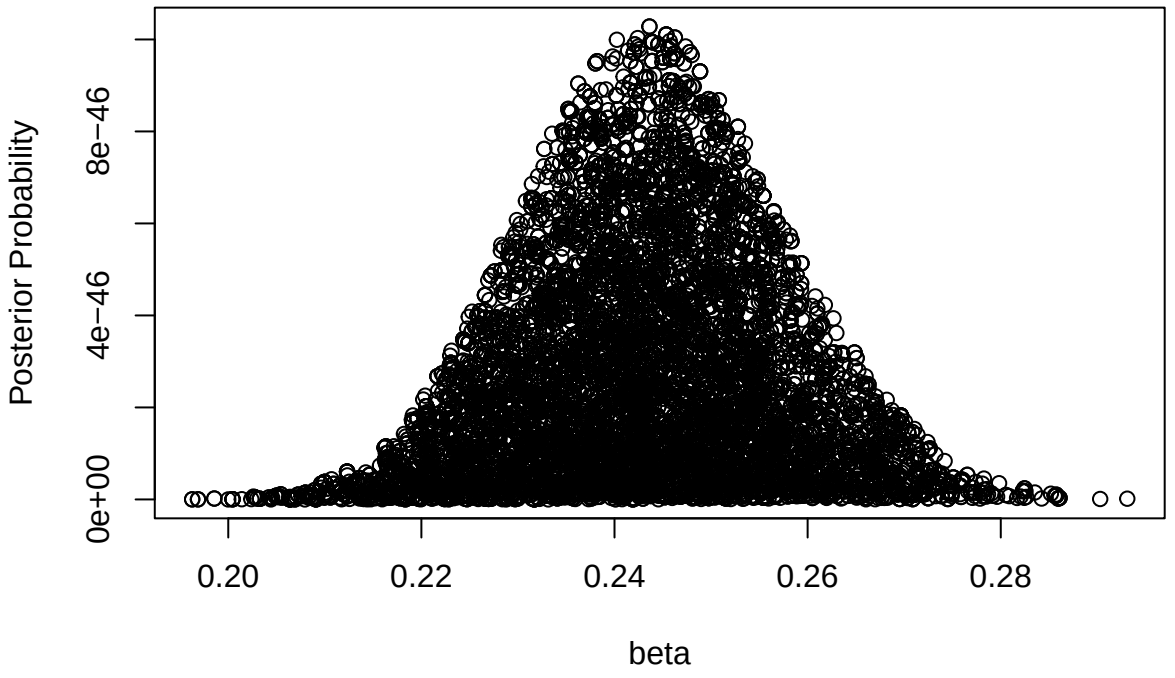

```
plot(I7,type = "l",main="Trace Plot of the Number of New Cases on Day 14")
```



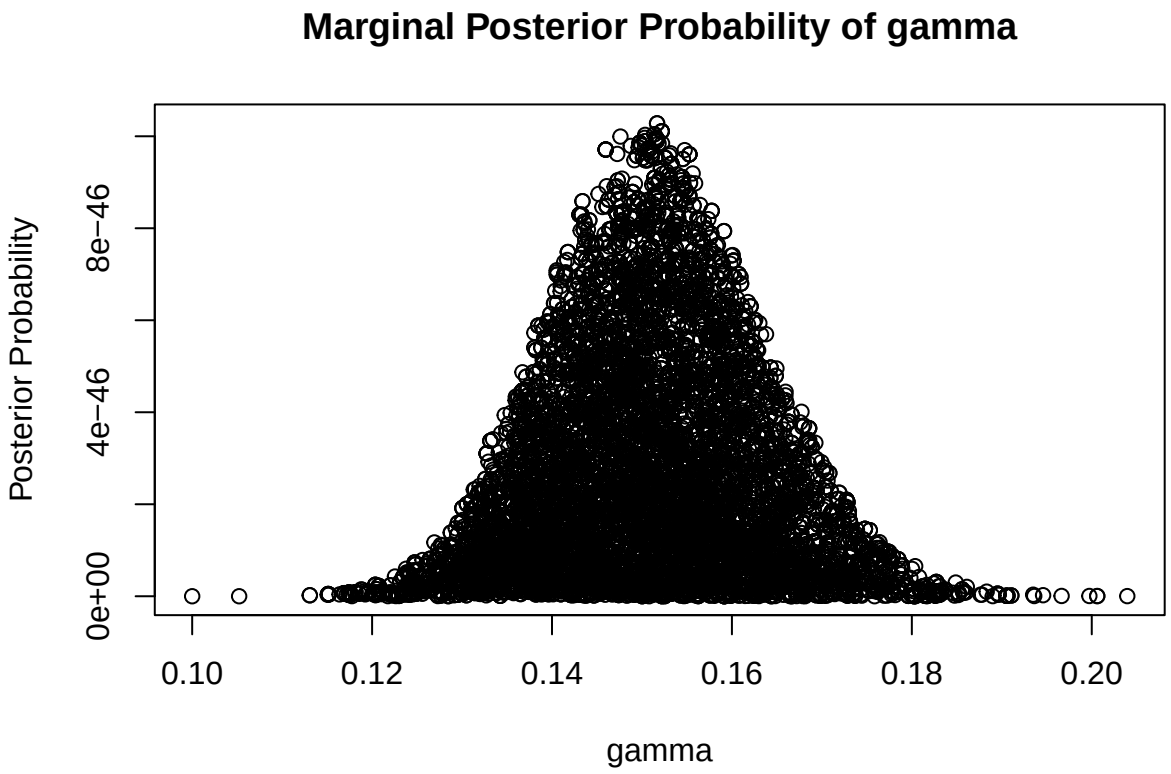

```
plot(R14,type="l",main="Trace Plot of the Number of New Removals on Day 14")
```



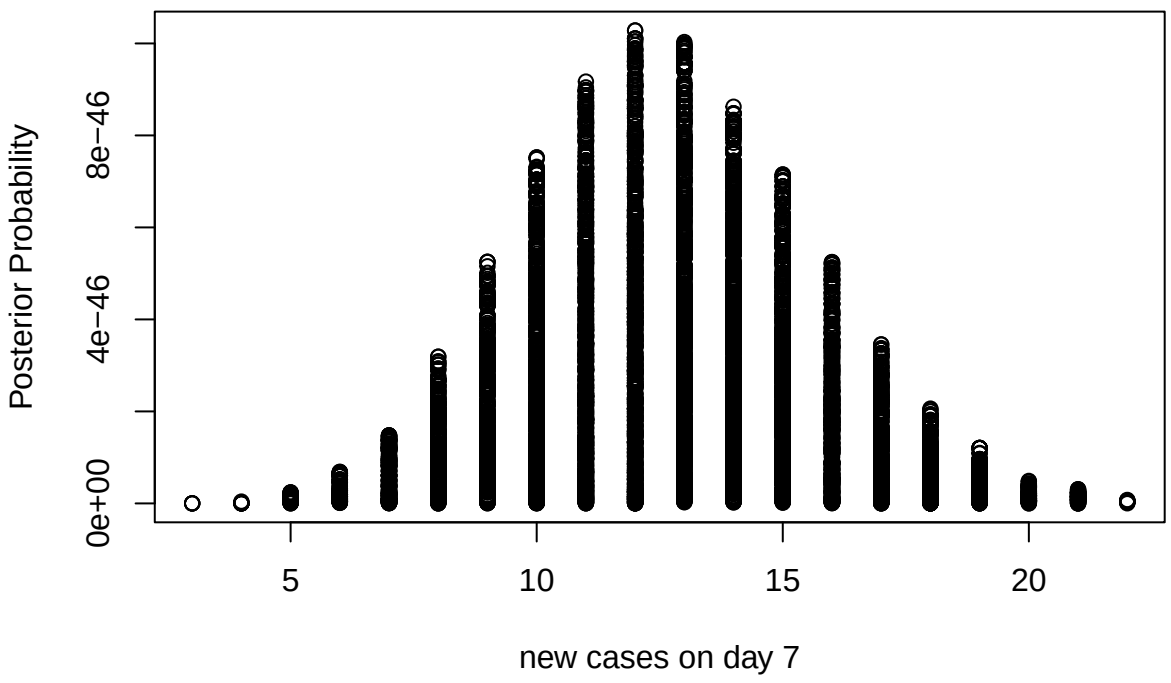

```
#marginal posterior distributions
plot(beta,joint,xlab="beta",ylab="Posterior Probability",main="Marginal Posterior Probability of beta")
```



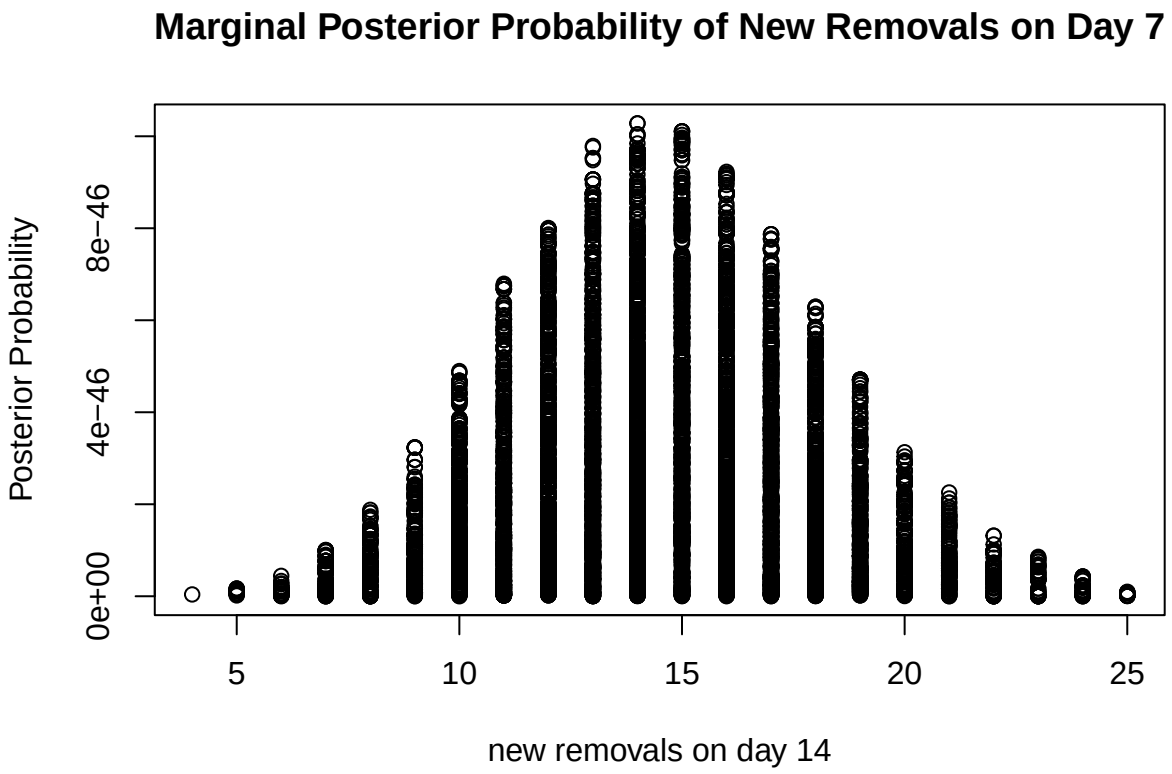

```
plot(gamma,joint,xlab="gamma",ylab="Posterior Probability",main="Marginal Posterior Probability of gamma
```




```
plot(I7,joint,xlab="new cases on day 7",ylab="Posterior Probability",main="Marginal Posterior Probabili
```




```
plot(R14,joint,xlab="new removals on day 14",ylab="Posterior Probability",main="Marginal Posterior Proba
```



3 SIR Model - Parametric Bootstrap Confidence Intervals

For this part, we want to find maximum likelihood estimates for the param- eters of our discrete-time SIR model. Using these estimates, we can derive parametric bootstrap-based 95% percentile confidence intervals for the two model parameters, and then compare these intervals with 95% percentile credible intervals for each of the two model parameters using the analysis from 1 (i).

```
options(width=60,length=200,digits=3)
library(stats)
set.seed(31)
data = read.csv("epidemic2024A3.csv",header = TRUE)
I_new = data$cases
R_new = data$removals
time = length(data$day)
N = 10000
S = c()
I = c()
R = c()
I[1] = 25
R[1] = 0
S[1] = N - I[1]
for (t in 1:time){
  S[t+1] = S[t] - I_new[t]
  I[t+1] = I[t] + I_new[t] - R_new[t]
  R[t+1] = R[t] + R_new[t]
}

### function of estimating MLE for parameters beta and gamma
estimate_param = function(I_star, R_star, Sus, Infec, n, time) {
```

```
 # Define log likelihood function for MLE
 likelihood = function(param) {
   beta = param[1]
   gamma = param[2]
    ll = 0
    for (t in 1:time) {
      p1 = dbinom(I_star[t], size = Sus[t], prob = 1 - exp(-beta * Infec[t] / n))
      p2 = dbinom(R_star[t], size = Infec[t], prob = 1 - exp(-gamma))
      ll = ll + log(p1) + log(p2)
    }
    #add prior distribution
    ll = ll+log(dunif(beta,0,0.5))+log(dunif(gamma,0,0.5))
    return(-ll)  # Return negative log-likelihood for optimization
 }

  # Perform MLE
  mle_result = optim(c(0.2, 0.1), likelihood, method = "Nelder-Mead")
  beta_mle = mle_result$par[1]
  gamma_mle = mle_result$par[2]
  return(c(beta_mle, gamma_mle))
}

para.est = estimate_param(I_new,R_new,S,I,N,time)
beta.est = para.est[1]
gamma.est = para.est[2]
cat("The maximum likelihood estimate for beta is",beta.est,"\n")
```

```
## The maximum likelihood estimate for beta is 0.241
```

```
cat("The maximum likelihood estimate for gamma is",gamma.est,"\n")
```

```
## The maximum likelihood estimate for gamma is 0.148
```

```
#### function for simulating data from the SIR model
simulate_data = function(beta, gamma, n, I0, time) {
  S = array(0,time + 1)
  I = array(0,time + 1)
  R = array(0,time + 1)
  I_star = array(0,time)
  R_star = array(0,time)
  S[1] = n - I0
  I[1] = I0
  R[1] = 0

  for (t in 1:time) {
    I_star[t] = rbinom(1, S[t], 1 - exp(-beta * I[t] / n))
    R_star[t] = rbinom(1, I[t], 1 - exp(-gamma))
    S[t+1] = S[t] - I_star[t]
    R[t+1] = R[t] + R_star[t]
    I[t+1] = n - S[t+1] - R[t+1]
  }

  #discard the last value
  return(list(S = S[-1], I = I[-1], I_star = I_star, R_star = R_star))
}
```

```r
# Number of bootstrap samples
nboot = 1000
bootstrap_est <- matrix(nrow = nboot, ncol = 2)

#perform bootstrap samples
for (i in 1:nboot) {
simulated_data <- simulate_data(beta.est, gamma.est, N, I[1], time)
bootstrap_est[i, ] <- estimate_param(simulated_data$I_star, simulated_data$R_star, simulated_data$S, si
}

#sort the samples by ascending order
sort.beta <- sort(bootstrap_est[,1])
sort.gamma <- sort(bootstrap_est[,2])

# Calculate confidence intervals
beta.ci.lower <- round(sort.beta[0.025*nboot],3)
beta.ci.upper <- round(sort.beta[0.975*nboot],3)
gamma.ci.lower <- round(sort.gamma[0.025*nboot],3)
gamma.ci.upper <- round(sort.gamma[0.975*nboot],3)

# Print results
cat("The mean of beta estimates is",mean(bootstrap_est[,1]),"\n")
```

```
## The mean of beta estimates is 0.218
```

```r
cat("The mean of gamma estimates is",mean(bootstrap_est[,2]),"\n")
```

```
## The mean of gamma estimates is 0.135
```

```r
cat("Bootstrap 95% Confidence Intervals for Beta: (", beta.ci.lower,",",beta.ci.upper, ")\n")
```

```
## Bootstrap 95% Confidence Intervals for Beta: ( 0.196 , 0.237 )
```

```r
cat("Bootstrap 95% Confidence Intervals for Gamma:(", gamma.ci.lower,",",gamma.ci.upper, ")\n")
```

```
## Bootstrap 95% Confidence Intervals for Gamma:( 0.114 , 0.161 )
```

In question 1 part (i), the 95% percentile interval for β is (0.219,0.268), for γ is (0.130,0.172). However, our bootstrap 95% confidence interval for β is (0.196,0.237), for γ is (0.114,0.161).

We can see that mean of β and γ from bootstrap samples are smaller than MLE, and lower and upper bounds of bootstrap confidence intervals are smaller than the those of percentile interval in Q1 (i), that could be caused by the sampling method difference. Because the bootstrap method resamples from the observed data to estimate the sampling distribution of the parameters, while MCMC generates samples from the posterior distribution directly. When we simulate data sets under the MLE of β and γ, there is more uncertainty. The width of intervals is similar.