Sekhar_Mekala_HW5

Sekhar Mekala

Tuesday, November 01, 2016

Problem 1

a)

The transition matrix is given below:

```
p <- matrix(c(.9,.05,.03,.02,0,.85,.09,.06,0,0,.9,.1,1,0,0,0),byrow=TRUE,nrow=4)
print(p)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]  0.9 0.05 0.03 0.02
## [2,]  0.0 0.85 0.09 0.06
## [3,]  0.0 0.00 0.90 0.10
## [4,]  1.0 0.00 0.00 0.00
```

b)

Given that the new state always starts in low state. Therefore the probabilities of various states after 3 weeks are given below:

```
library(expm)
```

```
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
##
## The following objects are masked from 'package:base':
##
##     crossprod, tcrossprod
##
##
## Attaching package: 'expm'
##
## The following object is masked from 'package:Matrix':
##
##     expm
```

```
#Initial state
x <- matrix(c(1,0,0,0),byrow=TRUE,nrow=1)
x %*% (p %^% 3)
```

```
##       [,1]     [,2]     [,3]   [,4]
## [1,] 0.771 0.115875 0.085425 0.0277
```

Therefore, the probability that the machine will be in failed state after 3 weeks is 0.0277

c)

Let us find the probabilities of failures in the first, second and third weeks:

First week's probability of failure is obtained as follows:

```
x %*% (p %^%1)[,4]
```

```
##      [,1]
## [1,] 0.02
```

Second week's probability of failure is obtained as follows:

```
x %*% (p %^%2)[,4]
```

```
##       [,1]
## [1,] 0.024
```

Third week's probability of failure is obtained as follows:

```
x %*% (p %^%3)[,4]
```

```
##        [,1]
## [1,] 0.0277
```

Let us assume the following events:

- *E1* that the machine fails in the first week. Its probability = *P(E1)* = 0.02
- *E2* that the machine fails in the second week. Its probability = *P(E2)* = 0.024
- *E3* that the machine fails in the third week. Its probability = *P(E3)* = 0.0277

Therefore the probability of *E1* or *E2* or *E3* is obtained as:

$$P(E_1 \cup E_2 \cup E_3) = P(E_1) + P(E_2) + P(E_3) - P(E_1 \cap E_2) - P(E_2 \cap E_3) - P(E_1 \cap E_3) + P(E_1 \cap E_2 \cap E_3)$$

Since *E1*, *E2* and *E3* are independent,

$$P(E_1 \cap E_2) = (0.02)(0.024) = 0.00048$$

$$P(E_2 \cap E_3) = (0.024)(0.0277) = 0.0006648$$

$$P(E_1 \cap E_3) = (0.02)(0.0277) = 0.000554$$

$$P(E_1 \cap E_2 \cap E_3) = (0.02)(0.024)(0.0277) = 0.000013296$$

Hence, the probability of at least one failure in first 3 weeks is 0.0700145

d)

Let us find the steady state probability can be found by solving the following equation, where a, b, c and d represent the low, medium, high and failed states probabilities respectively in steady state:

$$\begin{bmatrix} a & b & c & d \end{bmatrix} \begin{bmatrix} 0.9 & 0.05 & 0.03 & 0.02 \\ 0.0 & 0.85 & 0.09 & 0.06 \\ 0.0 & 0.00 & 0.90 & 0.10 \\ 1.0 & 0.00 & 0.00 & 0.00 \end{bmatrix} = \begin{bmatrix} a & b & c & d \end{bmatrix}$$

After solving the above equation, we obtained

$$a = p(low) = 0.4918033$$

$$b = p(medium) = 0.1639344$$

$$c = p(high) = 0.295082$$

$$d = p(failed) = 0.04918033$$

Hence the average number of weeks for the first failure = 1/0.04918033 = 20.3333325 weeks

e)

On an average the number of weeks the machine works in 1 year = 52X0.4918033 = 25.5737716 weeks

f)

Average profit in long run is:

$$\begin{bmatrix} 0.4918033 & 0.1639344 & 0.295082 & 0.04918033 \end{bmatrix} . \begin{bmatrix} 1000 \\ 500 \\ 400 \\ -700 \end{bmatrix}$$

```
matrix(c(0.4918033, 0.1639344, 0.295082, 0.04918033),nrow=1,byrow=TRUE) %*%
  matrix(c(1000, 500, 400, -700),nrow=4,byrow=TRUE)
```

```
##          [,1]
## [1,] 657.3771
```

Hence, a profit of $657.3771 is obtained per week.

g)

When the machine is repaired soon after it is in the high state, then the tarnsition matrix will be:

```
p <- matrix(c(.9,.05,.03,.02,0,.85,.09,.06,1,0,0,0,1,0,0,0),byrow=TRUE,nrow=4)
p
```

```
##      [,1] [,2] [,3] [,4]
## [1,]  0.9 0.05 0.03 0.02
## [2,]  0.0 0.85 0.09 0.06
## [3,]  1.0 0.00 0.00 0.00
## [4,]  1.0 0.00 0.00 0.00
```

Let us find the steady state probability for the new transition matrix. I can be found by solving the following equation, where a, b, c and d represent the low, medium, high and failed states probabilities respectively in steady state:

$$\begin{bmatrix} a & b & c & d \end{bmatrix} \begin{bmatrix} 0.9 & 0.05 & 0.03 & 0.02 \\ 0.0 & 0.85 & 0.09 & 0.06 \\ 1.0 & 0 & 0 & 0 \\ 1.0 & 0.00 & 0.00 & 0.00 \end{bmatrix} = \begin{bmatrix} a & b & c & d \end{bmatrix}$$

After solving the above equation, we obtained

$$a = p(low) = 0.6976744$$

$$b = p(medium) = 0.2325581$$

$$c = p(high) = 0.04186047$$

$$d = p(failed) = 0.02790698$$

Hence, the average profit per week will be:

$$\begin{bmatrix} 0.6976744 & 0.2325581 & 0.04186047 & 0.02790698 \end{bmatrix} . \begin{bmatrix} 1000 \\ 500 \\ 400 \\ -700 \end{bmatrix}$$

```r
matrix(c(0.6976744, 0.2325581, 0.04186047, 0.02790698),nrow=1,byrow=TRUE) %*%
  matrix(c(1000, 500, -600, -700),nrow=4,byrow=TRUE)
```

```
##          [,1]
## [1,] 769.3023
```

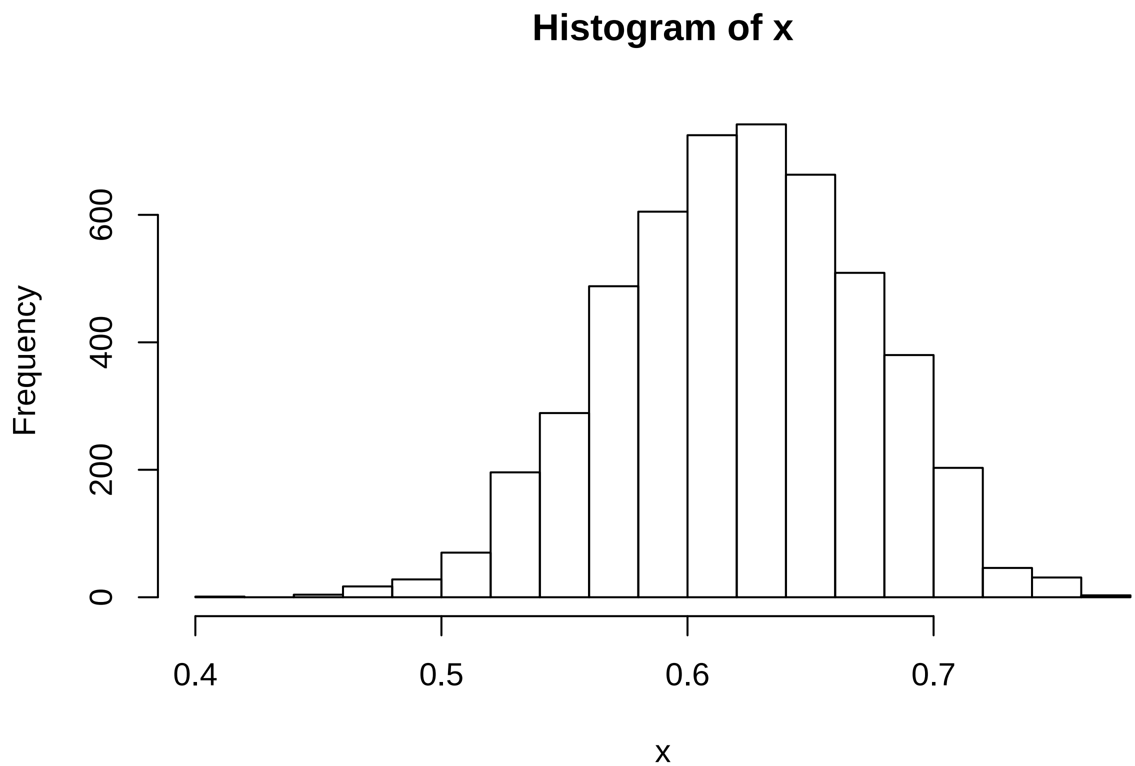
Hence, a profit of $769.3023 is obtained per week. Since the profit is more in this case, it is suggested to repair the machine soon after it reaches the high state.

Problem-2

The following R code uses Metropolis-Hastings algorithm to estimate the θ value:

```r
#Initialize the observed counts
count <- c(125,18,20,34)

#Create a probability function
prob <- function(y, count)
{
  #Computes the target density
  if(y < 0 || y >= 1)
```

```
    return(0)
  return((2+y)^count[1]*(1-y)^(count[2]+count[3])*y^count[4])

}


w <- 0.5 #width of uniform support set
m <- 5000 #length of the chain
burn <- 1000 #burn in time
animals <- 197


u <- runif(m)
v <- runif(m,-w,w)

x[1] <- .5
for(i in 2:m)
{
  y <- x[i-1] + v[i]
  if(u[i] <= prob(y,count)/prob(x[i-1],count))
    x[i] <- y else
      x[i] <- x[i-1]

}

hist(x)
```




```r
xb <- x[(burn+1):m]
theta <- mean(xb)

theta
```

```
## [1] 0.6209864
```

```r
#posterior distribution
p <- c(.5+theta/4,(1-theta)/4,(1-theta)/4,theta/4)
p
```

```
## [1] 0.65524659 0.09475341 0.09475341 0.15524659
```

```r
#p*197
```

Hence the $\theta = 0.6187743$

Therefore the posterior distribution will be: (0.6552466, 0.0947534, 0.0947534, 0.1552466)

Problem 3:

$\lambda =$ Mean of the data before the change point

$\phi =$ Mean of the data after the change point

$m =$ Change point

$\beta =$ Scale parameter for the distribution of λ

$\delta =$ Scale parameter for the distribution of ϕ

The following R code will obtain the following parameters (will run for 5000 iterations):

```
#Read the coal data set

library(boot)
data(coal)

year <- floor(coal)
y <- table(year)

#plot(y)


y <- floor(coal[[1]])

y <- tabulate(y)

y <- y[1851:length(y)]


#plot(y)

# Initialization
n <- length(y) #Length of data

m <- 5000 #length of chain

mu <- lambda <- k <- b1 <- b2 <- numeric(m)

L <- numeric(n)

k[1] <- sample(1:n,1)
mu[1] <- 1
lambda[1] <- 1
b1[1] <- 1
b2[1] <- 1

#Run the Gibbs sampler

for(i in 2:m)
{
  kt <- k[i-1]

  #generate mu
  r <- .5 + sum(y[1:kt])
  mu[i] <- rgamma(1, shape=r,rate=kt+b1[i-1])

  #generate lambda
  if(kt+1 > n)
    r <- .5 +sum(y)
```

```r
  else
    r <- .5 + sum(y[(kt+1):n])
  lambda[i] <- rgamma(1,shape=r,rate=n-kt+b2[i-1])

  #generate b1 and b2
  b1[i] <- rgamma(1,shape=.5,rate=mu[i]+1)
  b2[i] <- rgamma(1,shape=.5,rate=lambda[i]+1)

  for(j in 1:n)
  {
    L[j] <- exp((lambda[i] - mu[i]) * j) *
              (mu[i] / lambda[i])^sum(y[1:j])

  }

  L <- L / sum(L)

  #Generate k from discrete dist L on 1:n
  k[i] <- sample(1:n,prob=L, size=1)
}


phi <- lambda
lambda <- mu
m <- k
beta <- b1
delta <- b2

mean(phi)
```

```
## [1] 0.926059
```

```r
sd(phi)
```

```
## [1] 0.1203184
```

```r
mean(lambda)
```

```
## [1] 3.124374
```

```r
sd(lambda)
```

```
## [1] 0.2945038
```

```r
mean(m)
```

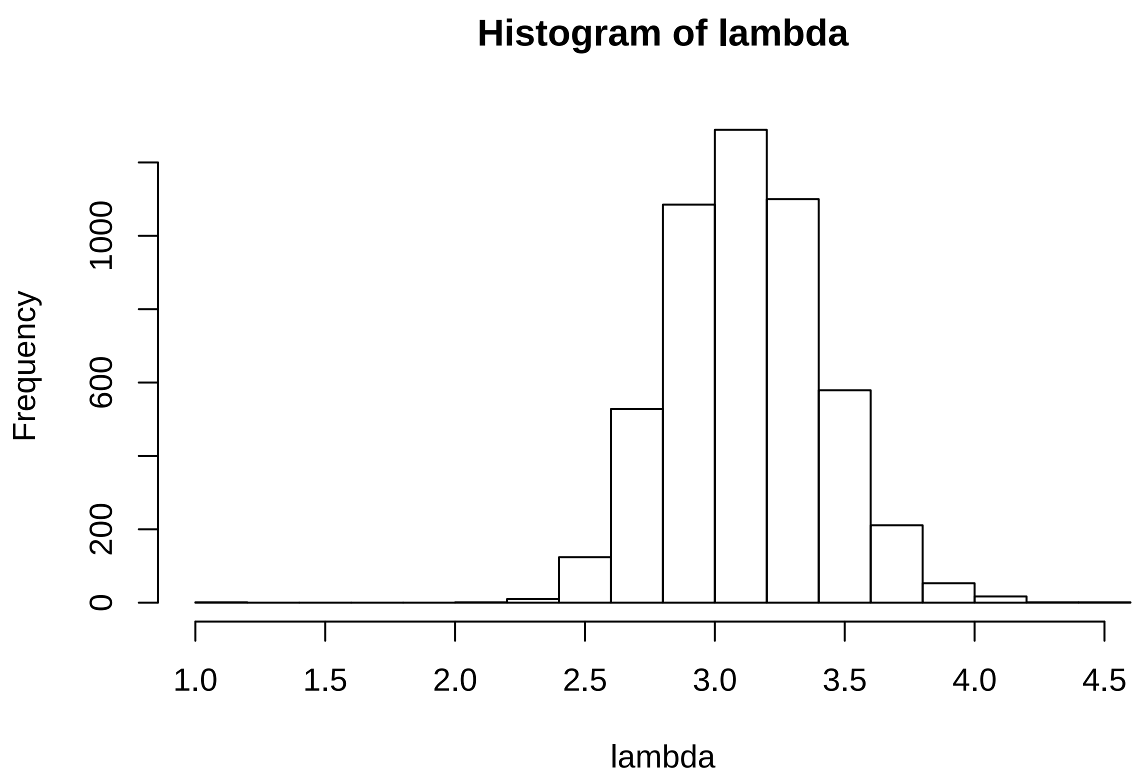
```
## [1] 39.9146
```

```r
sd(m)
```

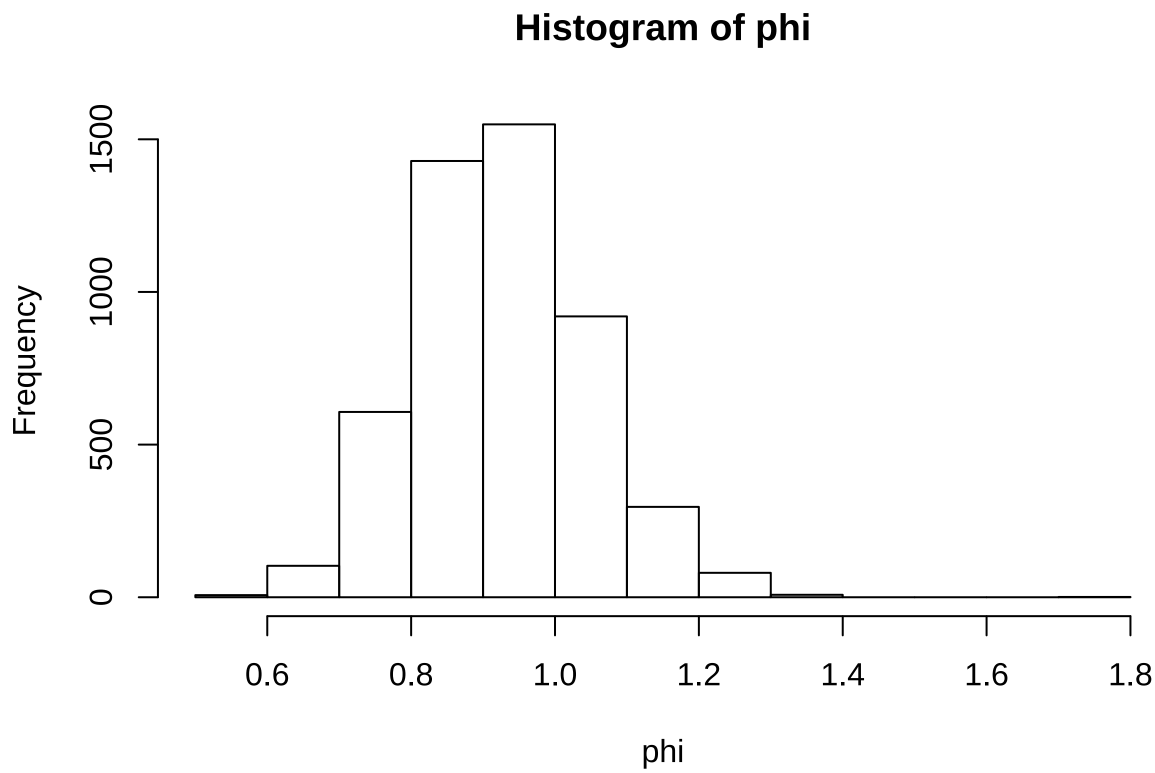
```
## [1] 2.504189
```

a)

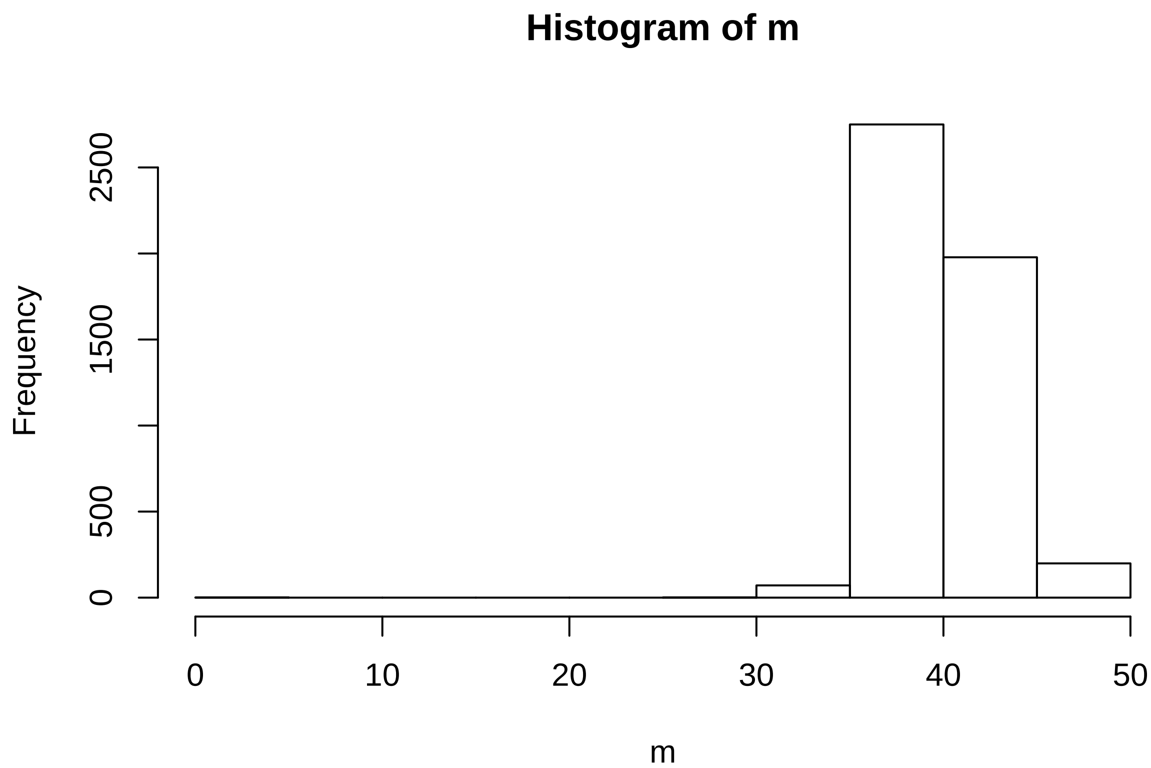
```r
#par(mfrow=c(1,1))
#Histogram of lambda
hist(lambda)
```



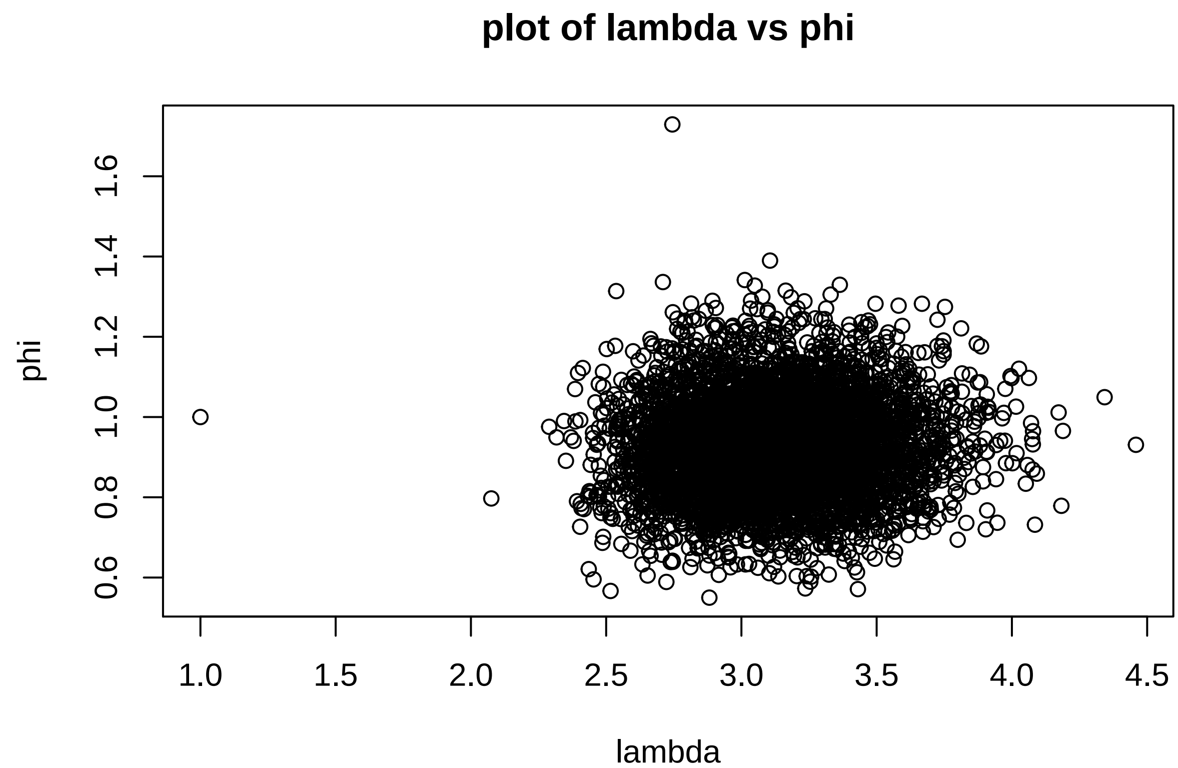

```r
#Histogram of phi
hist(phi)
```



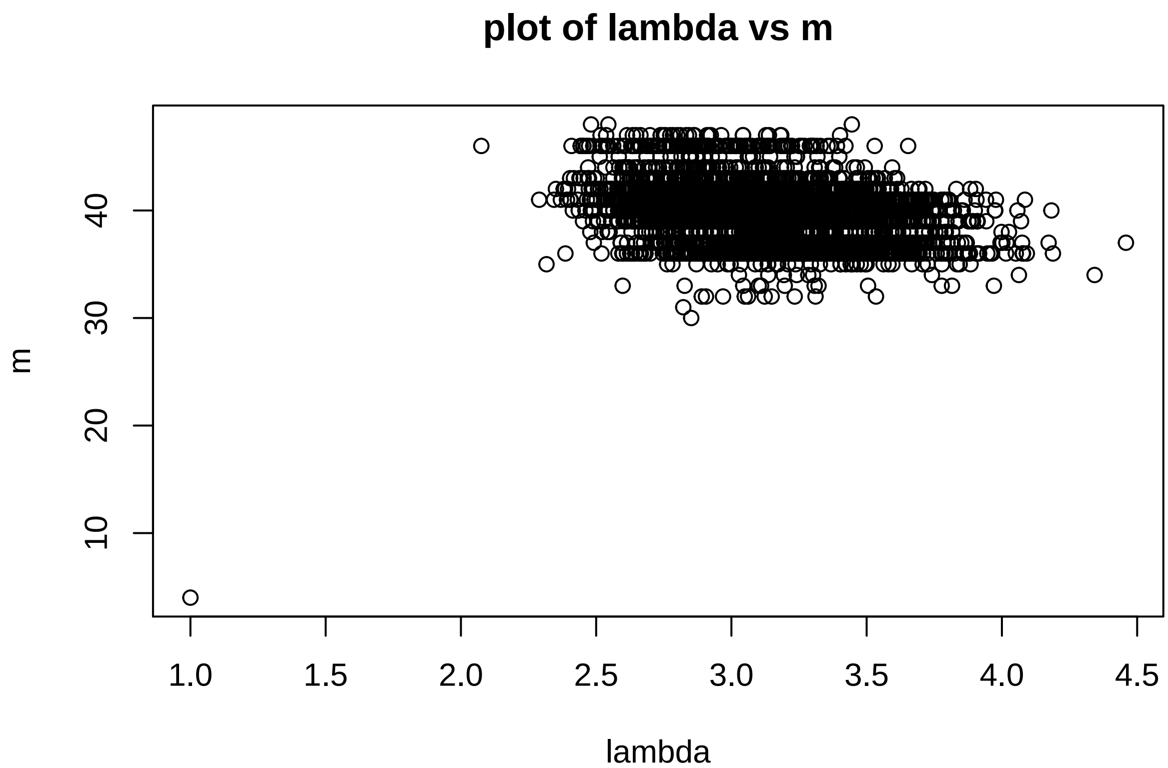

```
#Histogram of m
hist(m)
```



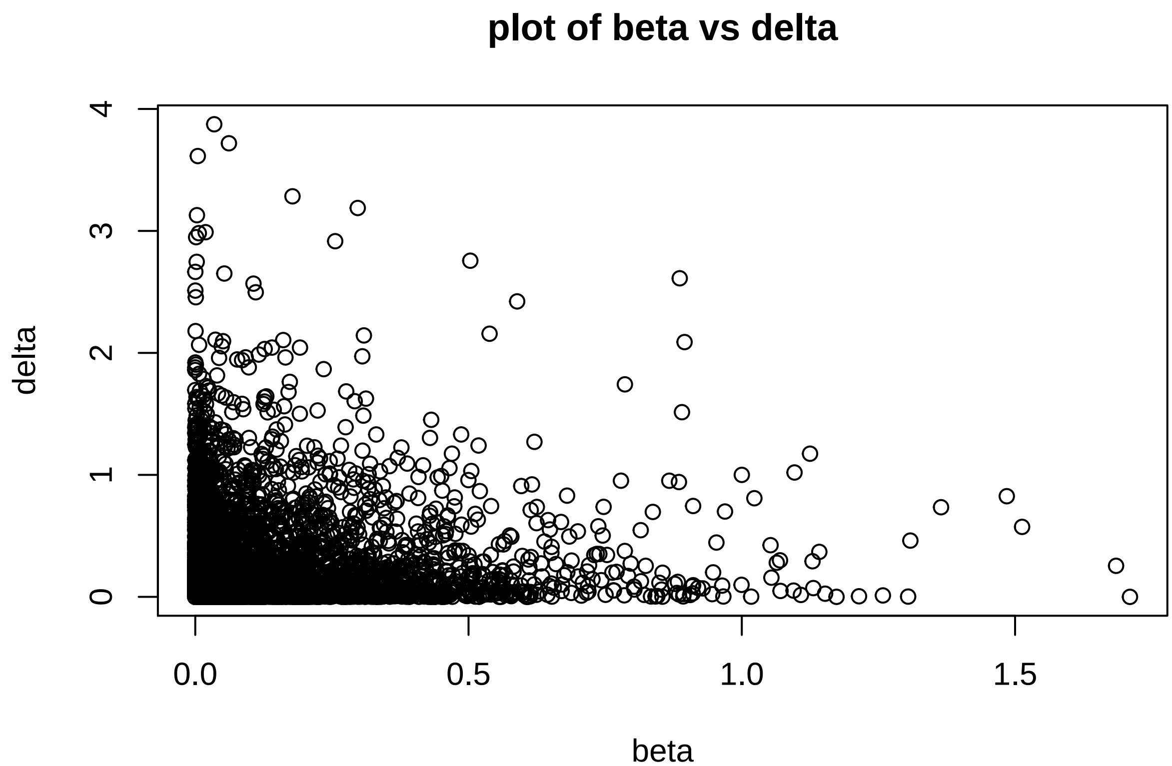

```r
#Plot of lambda vs phi
plot(lambda,phi,main="plot of lambda vs phi")
```




```r
#Plot of lambda vs m
plot(lambda,m,main="plot of lambda vs m")
```




```r
#Plot of beta vs delta
plot(beta,delta,main="plot of beta vs delta")
```



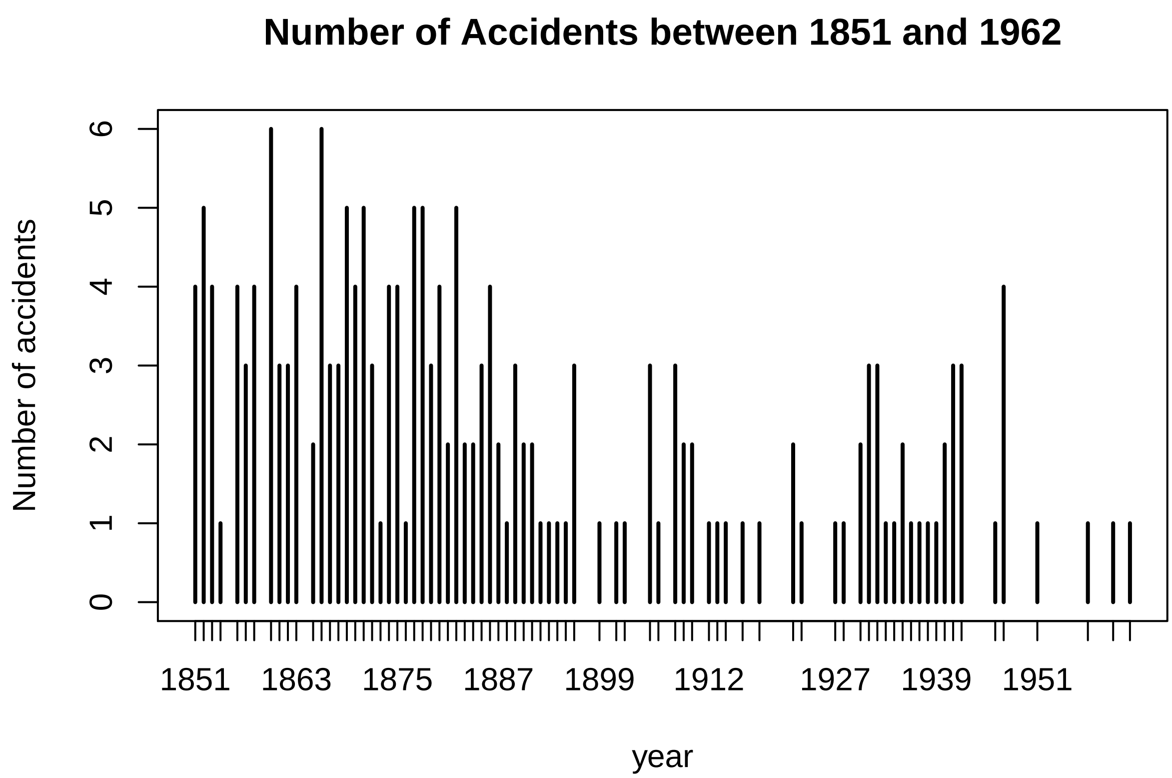
b)

The change point has occured after 39.9146 years. The 95% confidence interval is 39.9146 $\pm$(1.96) (2.5041887) = [35.0063902,44.8228098]

The average number of accidents before change point = 3.1243736/year and average number of accidents after change point = 0.926059/year

Let us plot the time series of the observations:

```
year <- floor(coal)
y <- table(year)

plot(y,main="Number of Accidents between 1851 and 1962",ylab="Number of accidents")
```



We can clearly see that after approximately 40 years (around 1891), the number of accidents have drastically decreased. Our results obtained are consistent with the time series of observed data, since we see a drastic decrease in the accidents after 40 years.

c)

Gibbs sampling is a special case of Metropolis-Hastings method. In Gibbs sampling we do not reject any sample, and Gibbs sampling is often applied when the target distribution is multi-variate.

Problem-4

The cost matrix is displayed below:

```
cost
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
##  [1,]    0  633  257   91  412  150   80  134  259   505   353   324    70
##  [2,]  633    0  390  661  227  488  572  530  555   289   282   638   567
##  [3,]  257  390    0  228  169  112  196  154  372   262   110   437   191
##  [4,]   91  661  228    0  383  120   77  105  175   476   324   240    27
##  [5,]  412  227  169  383    0  267  351  309  338   196    61   421   346
##  [6,]  150  488  112  120  267    0   63   34  264   360   208   329    83
##  [7,]   80  572  196   77  351   63    0   29  232   444   292   297    47
##  [8,]  134  530  154  105  309   34   29    0  249   402   250   314    68
```

```
##  [9,]  259  555  372  175  338  264  232  249    0   495   352    95   189
## [10,]  505  289  262  476  196  360  444  402  495     0   154   578   439
## [11,]  353  282  110  324   61  208  292  250  352   154     0   435   287
## [12,]  324  638  437  240  421  329  297  314   95   578   435     0   254
## [13,]   70  567  191   27  346   83   47   68  189   439   287   254     0
## [14,]  211  466   74  182  243  105  150  108  326   336   184   391   145
## [15,]  268  420   53  239  199  123  207  165  383   240   140   448   202
## [16,]  246  745  472  237  528  364  332  349  202   685   542   157   289
## [17,]  121  518  142   84  297   35   29   36  236   390   238   301    55
##       [,14] [,15] [,16] [,17]
##  [1,]   211   268   246   121
##  [2,]   466   420   745   518
##  [3,]    74    53   472   142
##  [4,]   182   239   237    84
##  [5,]   243   199   528   297
##  [6,]   105   123   364    35
##  [7,]   150   207   332    29
##  [8,]   108   165   349    36
##  [9,]   326   383   202   236
## [10,]   336   240   685   390
## [11,]   184   140   542   238
## [12,]   391   448   157   301
## [13,]   145   202   289    55
## [14,]     0    57   426    96
## [15,]    57     0   483   153
## [16,]   426   483     0   336
## [17,]    96   153   336     0
```

Let us write a function that finds the cost of a potential solution:

```
find_cost <- function(cost_matrix,solution)
  {
    y <- numeric(16)
    for(i in 1:16)
    {
          y[i] <- cost_matrix[solution[i],solution[i+1]]

    }
    return(sum(y))
  }
```

The simulated annealing algorithm is defined below:

```
simulated_annealing <- function(T0, drop,n,cost_matrix)
  {

    #Get the number of cities
    s <- nrow(cost_matrix)

    #Declare a matrix to store all the solutions
    accepted_solution <- matrix(rep(0,170000),
                                byrow=TRUE,nrow=10000)
```

```r
  #Vector to store all costs
  c_1 <- vector()

  #Initial solution
  solution_1 <- sample(1:s,s)

  #Initial cost
  c_1[1] <- find_cost(cost_matrix,solution_1)

  #Initial accepted solution
  accepted_solution[1,] <- solution_1

  for(i in 2:n)
    {
      #Get the 2 elements randomly
      index <-  sort(sample(1:s,s)[1:2])

      #Prepare a second solution
      solution_2 <-
        c(solution_1[1:ifelse(index[1] == 1, 1,
                          (index[1]-1))],
          solution_1[(ifelse(index[2]==17,16,
                          index[2])):ifelse(index[1]==1,2,index[1])],
          solution_1[ifelse((index[2])==17,17,
                          (index[2]+1)):17])

       #Find the cost of the current solution or solution-2
       c_1[i] <- find_cost(cost_matrix,solution_2)

      #If solution-2 is better than solution-1, then accept solution-2 blindly
      if(c_1[i-1] > c_1[i])
              {
              accepted_solution[i,] <- solution_2
              T0 <- T0 * drop
              solution_1 <- solution_2
              } else
              {
                  p <- exp((c_1[i-1] - c_1[i])/T0)
                  u <- runif(1)
                  if(p > u)
                    {
                  accepted_solution[i,] <- solution_2
                   }
                  else{
                    c_1[i] <- c_1[i-1]
                    accepted_solution[i,] <- solution_1
                   }
                 T0 <- T0 * drop
             }

    }
 return(list(solution=accepted_solution,cost=c_1,T0=T0))
```

```
}

#Get the cost of the best solution
l <- simulated_annealing(1, 0.9999,10000,cost)

l$cost[10000]
```

```
## [1] 1948
```

```
#Let us get the cost of a random path
x <- sample(1:17,17)

find_cost(cost,x)
```

```
## [1] 3307
```

The cost of the best solution is 1948, while the cost of the random path is: 3307.

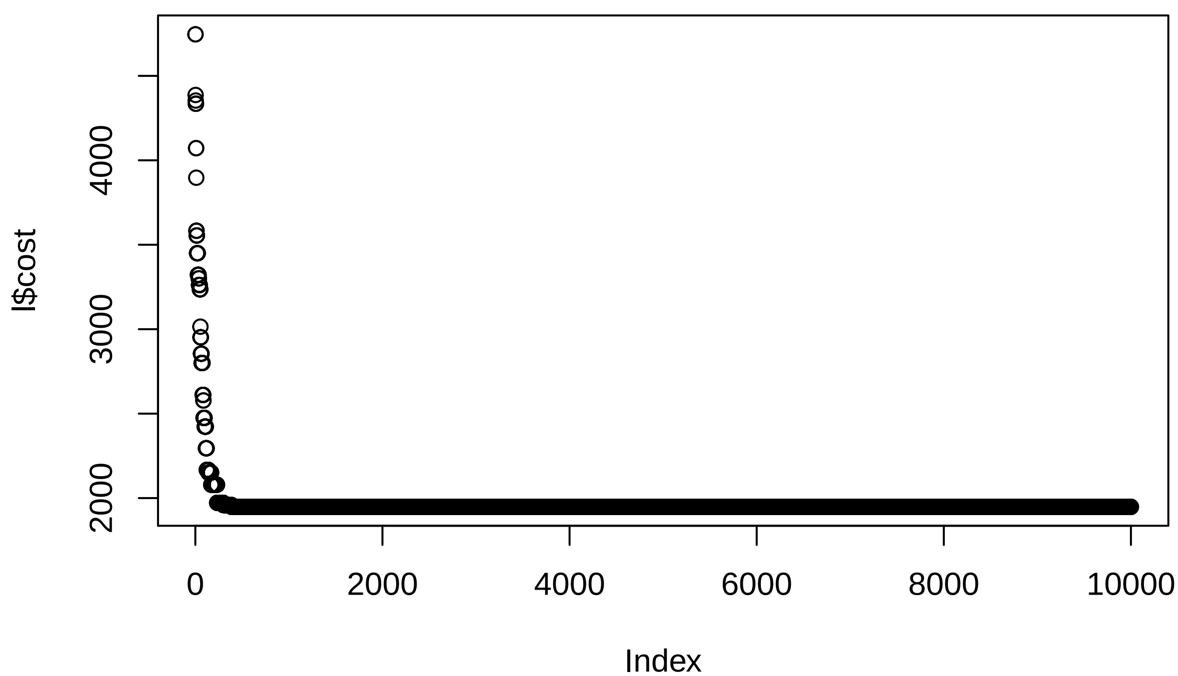
The optimal path obtained using the algorithm is:

```
l$solution[10000,]
```

```
##  [1] 11  5  2 10 15  3 14  6 17 13  4  9 12 16  1  7  8
```

Let us plot the iterations vs the cost of the solution in the respective iteration:

```
plot(l$cost)
```



We can infer that after approximately 300 iterations, the cost has stabilized.

Repeating the simulated annealing for 4 times to find how the cost of the optimal solution is varying

```
set.seed(1234)
l1 <- simulated_annealing(1, 0.9999,10000,cost)
l2 <- simulated_annealing(1, 0.9999,10000,cost)
l3 <- simulated_annealing(1, 0.9999,10000,cost)
l4 <- simulated_annealing(1, 0.9999,10000,cost)

l1$solution[10000,]
```

```
##  [1]  2 10  5 11  3  6  8  7  1 16 12  9  4 13 17 14 15
```

```
l1$cost[10000]
```

```
## [1] 1819
```

```
l2$solution[10000,]
```

```
##  [1]  3  6  8  7  1 13  4 16 12  9 17 14 15 11 10  2  5
```

```
l2$cost[10000]
```

```
## [1] 2040
```

```
l3$solution[10000,]
```

```
##  [1] 15  3 14 17  6  8  7 13  4  1 16 12  9 11  5  2 10
```

```
l3$cost[10000]
```

```
## [1] 1913
```

```
l4$solution[10000,]
```

```
##  [1]  9 12 16  4 13 17 14 15 10  2  5 11  3  6  8  7  1
```

```
l4$cost[10000]
```

```
## [1] 1906
```

The optimal solution's cost does not vary a lot, but the optimal solution is drastically different in different repetitions.