IS 604 Assignment 5

David Stern

October 27, 2015

Problem 1

a) Give the Markov transition matrix for X.

Here we will construct our transition matrix, t.

```
suppressWarnings(suppressMessages(library(expm)))
suppressWarnings(suppressMessages(library(knitr)))
t <-
matrix(c(0.9,0,0,1,0.05,0.85,0,0,0.03,0.09,0.9,0,0.02,0.06,0.1,0),nrow=
4)
rownames(t) <- c("from low","from medium","from high","from failed")
colnames(t) <- c("to low","to medium","to high","to failed")
kable(t)
```

	to low	to medium	to high	to failed
from low	0.9	0.05	0.03	0.02
from medium	0.0	0.85	0.09	0.06
from high	0.0	0.00	0.90	0.10
from failed	1.0	0.00	0.00	0.00

b) A new machine always starts in the low state. What is the probability that the machine is in the failed state three weeks after it is new?

Here we construct our initial state vector, i, and multiply it by the transition matrix to the third power. We will use the matrix exponentiation function in the expm package. The failure rate will be the last element in the distribution vector. The probability that the machine will be in a failed state three weeks after it is new is 0.0277.

```
i <- c(1,0,0,0)
i  %*% (t %^% 15)

##         to low to medium   to high  to failed
## [1,] 0.5129639 0.1794735 0.2608973 0.04666541
```

c) What is the probability that a machine has at least one failure three weeks after it is new?

With only four possible states and three transitions, we can approach this problem by summing the probabilities of all transitions from $X_0 \rightarrow X_3$ that include at least one failure.

For the four states, there are 11 transitions from $X_0 \rightarrow X_3$ that include 1 failure and 1 transition with 2 failures.

$$l \rightarrow f \rightarrow l \rightarrow f = 0.02 \times 1 \times 0.02 = 0.0004$$

$$l \rightarrow f \rightarrow l \rightarrow m = 0.02 \times 1 \times 0.05 = 0.001$$

$$l \rightarrow f \rightarrow l \rightarrow h = 0.02 \times 1 \times 0.03 = 0.0006$$

$$l \rightarrow f \rightarrow l \rightarrow l = 0.02 \times 1 \times 0.9 = 0.018$$

$$l \rightarrow h \rightarrow f \rightarrow l = 0.03 \times 0.1 \times 1 = 0.003$$

$$l \rightarrow h \rightarrow h \rightarrow f = 0.03 \times 0.9 \times 0.1 = 0.0027$$

$$l \rightarrow m \rightarrow f \rightarrow l = 0.05 \times 0.06 \times 1 = 0.003$$

$$l \rightarrow m \rightarrow m \rightarrow f = 0.05 \times 0.85 \times 0.06 = 0.0025$$

$$l \rightarrow l \rightarrow f \rightarrow l = 0.9 \times 0.02 \times 1 = 0.018$$

$$l \rightarrow l \rightarrow l \rightarrow f = 0.9 \times 0.9 \times 0.02 = 0.0162$$

$$l \rightarrow l \rightarrow m \rightarrow f = 0.9 \times 0.05 \times 0.06 = 0.0027$$

$$l \rightarrow l \rightarrow h \rightarrow f = 0.9 \times 0.03 \times 0.1 = 0.0027$$

The sum of these transitions give us: $P(N_{Failure} \geq 1) = 0.07085$

d) What is the expected number of weeks after a new machine is installed until the first failure occurs?

Here we can take the steady state probabilities, approximated after about 20 transitions, and divide the probability of failure by 1. If we round up the results, we would expect 21 weeks to pass until the first failure occurs.

```
PIj <- as.vector(i  %*% (t %^% 20))
failure <- PIj[4]
1/failure

## [1] 20.7137
```

e) On average, how many weeks per year is the machine working?

Here we will find the probability of the machine working - operating at a low, medium, or high state - for each of the 52 weeks of the year. We will then sum those probabilities to find that it is likely to operate 50 weeks of the year. We get the same result if we use the steady-state probabilities and multiple the probability that the machine is working by the number of weeks.

```
prob <- 0
for (week in 1:52){
  PIj <- as.vector(i  %*% (t %^% week))
```

```
  prob <- prob + sum(PIj[1:3])
  }
prob
```

```
## [1] 49.62941
```

```
PIj <- as.vector(i  %*% (t %^% 20))
sum(PIj[1:3])*52
```

```
## [1] 49.48958
```

f) Each week that the machine is in the low state, a profit of $1000 is realized; each week that the machine is in the medium state, a profit of $500 is realized; each week that the machine is in the high state, a profit of $400 is realized; and the week in which a failure is fixed, a cost of $700 is incurred. What is the long-run average profit per week realized by the machine?

Here we take the steady-state probabilities and multiply it by the vector of the earnings for each state. We will find a long-run average profit of $663 per week.

```
earnings <- c(1000,500,400,-700)
PIj %*% earnings
```

```
##          [,1]
## [1,] 662.6645
```

g) A suggestion has been made to change the maintenance policy for the machine. If at the start of a week the machine is in the high state, the machine will be taken out of service and repaired so that at the start of the next week it will again be in the low state. When a repair is made due to the machine being in the high state instead of a failed state, a cost of $600 is incurred. Is this new policy worthwhile?

Here we will modify the transition matrix and earnings vector and see how the long-run profit changes. Average weekly profit the machine is also repaired when it reaches a high state.

```
t2 <- t # copy transitions matrix
earnings2 <- earnings # copy earnings vector
t2[3,] <- c(1,0,0,0) # modify transitions matrix
earnings2[3] <- -600 # modify earnings vector
PIj2 <- as.vector(i  %*% (t2 %^% 20)) # also reach steady state at 20
transitions
PIj2 %*% earnings2
```

```
##          [,1]
## [1,] 770.8894
```

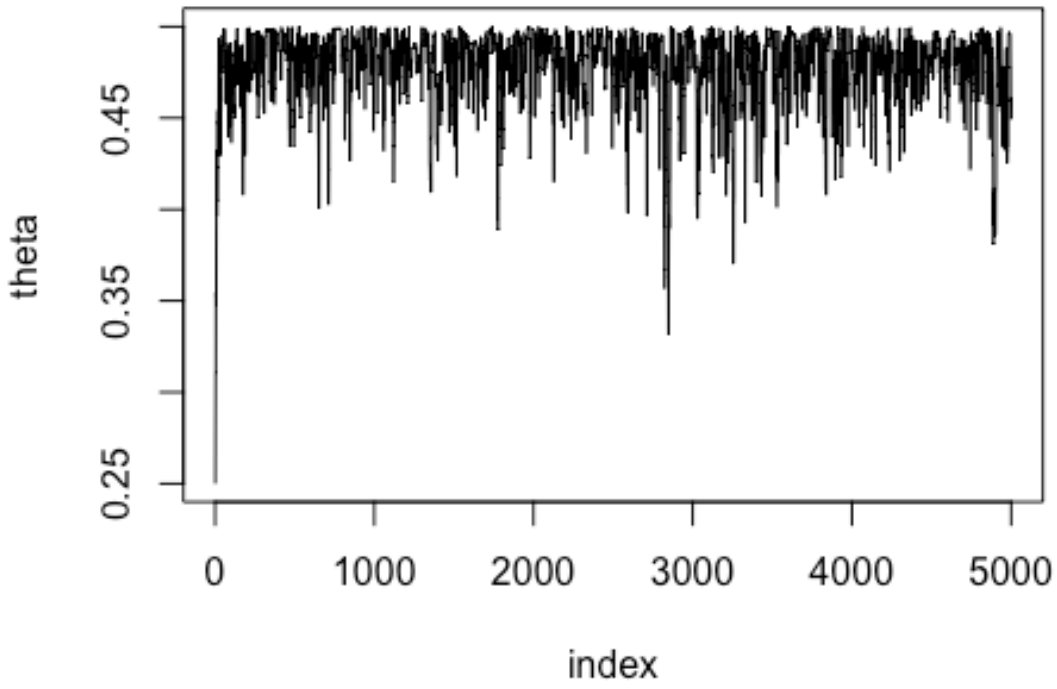
Problem 2

```
groups <- c(125,18,20,34)
```

```r
b <- mean(round(groups/197,3))
round(c(0.5+b/4,(1-b)/4,(1-b)/4,b/4),3)
```

```
## [1] 0.563 0.187 0.187 0.063
```

```r
w <- 0.05
m <- 5000
burn <- 2000
theta <- numeric(m)

prob <- function(y){
  if (y < 0 || y >= 0.5) return (0)
  return ((2+y)^groups[1])*((1-y)^(groups[2]+groups[3]))*(y^groups[4])
}

u <- runif(m)
v <- runif(m, -w, w)
theta[1] <- 0.25
  for (i in 2:m) {
    y <- theta[i-1] + v[i]
    if (u[i] <= min(1,prob(y) / prob(theta[i-1])))
      theta[i] <- y else
        theta[i] <- theta[i-1]
}

index <- 1:m
y1 <- theta[index]
plot(index,y1,type="l",main="",ylab="theta")
```



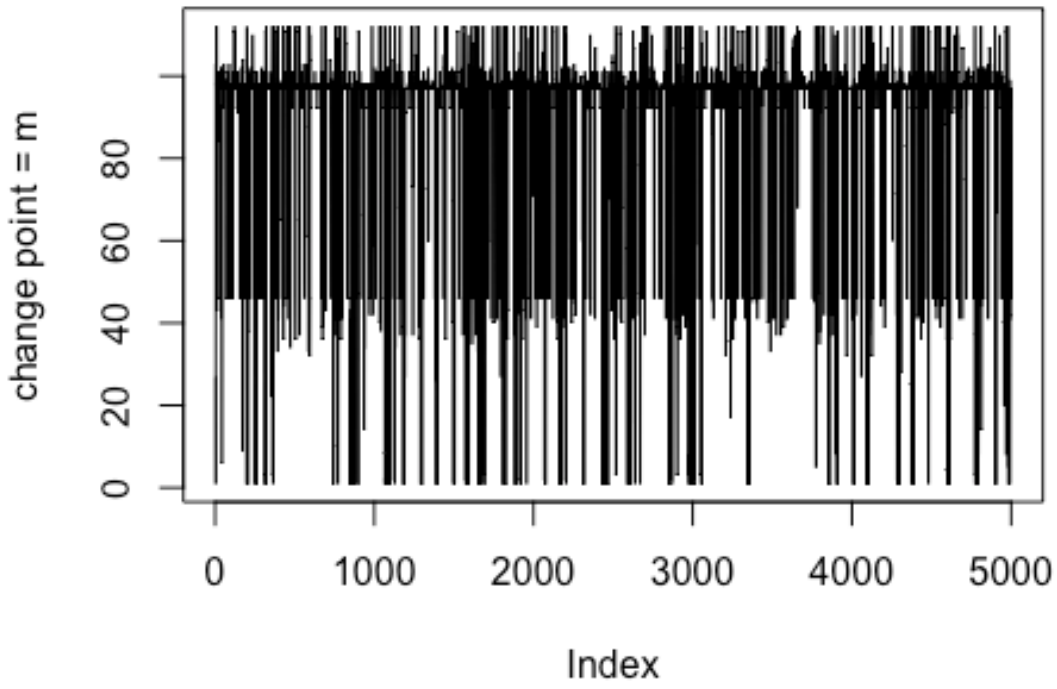

```
xb <- theta[(burn+1):m]
print(mean(xb))

## [1] 0.4783115

mean(xb)

## [1] 0.4783115
```

For our choice of step-value, we see that the chain is converges to the target distribution after a very short burn-in period and that it mixes will. If we discard the first 200 values, we will get sample mean of the generated chain of approximately 0.48.

Problem 3

```
Y <-
c(4,5,4,1,0,4,3,4,0,6,3,3,4,0,2,6,3,3,5,4,5,3,1,4,4,1,5,5,3,4,2,5,2,2,3
,4,2,1,3,2,2,1,1,1,1,3,0,0,1,0,1,1,0,0,3,1,0,3,2,2,0,1,1,1,0,1,0,1,0,0,
0,2,1,0,0,0,1,1,0,2,3,3,1,1,2,1,1,1,1,2,4,2,0,0,0,1,4,0,0,0,1,0,0,0,0,0
,1,0,0,1,0,1)

fullcondm <-
```

```
function(m0,lambda0,phi0,yval,lengthY,alpha0,beta0,gamma0,delta0){
  if (m0 > 1){ Lamexp <- sum(yval[1:m0])} else Lamexp <- 0
  if (m0 < lengthY){Phiexp <- sum(yval[(m0+1):lengthY])} else Phiexp <-
0
  result <- lambda0^(alpha0-1+Lamexp)*exp(-
(beta0+m0)*lambda0)*phi0^(gamma0-1+Phiexp)*exp(-(delta0+lengthY-
m0)*phi0)
  result
}

n <- length(Y)
iterations <- 5000
mu <- lambda <- phi <- beta <- delta <- numeric(iterations)
mprob <- rep(1/112,112)
mdist <- function(n) {sample(1:112,n,replace=TRUE,prob=mprob)}
mu[1] <- mdist(1)
lambda[1] <- 1
phi[1] <- 1
beta[1] <- 1
delta[1] <- 1
alpha <- 8
g <- 2

for (i in 2:iterations){
  m <- mu[i-1]
  sum1 = sum(Y[1:m])
  if (m+1 > n){sum2 <- sum(Y)}
    else {sum2 <- sum(Y[(m+1):n])}
  phi[i] <- rgamma(1,sum2+g,delta[i-1]*(n+m))
  lambda[i] <- rgamma(1,sum1+alpha,beta[i-1]*(n+m))

  for (j in 1:n){
    mprob[j] <- fullcondm(j, lambda[i-1], phi[i-1], Y, n, alpha,
beta[i-1], g, delta[i-1])
                }
  beta[i] <- 1 / rgamma(1,alpha,lambda[i-1]+1)
  delta[i] <- 1 / rgamma(1,g,phi[i-1]+1)
  while (beta[i] > 99){
    beta[i] <- 1/rgamma(1,alpha,lambda[i-1]+1)
    }
  while (delta[i] > 99){
    delta[i] <- 1/rgamma(1,g,phi[i-1]+1)
    }
  mprob <- mprob/sum(mprob)
  mu[i] <- mdist(1)
}
chain <- matrix(c(mu,lambda,phi,beta,delta),nrow=iterations)
colnames(chain) <- c("m","lambda","phi","beta","delta")
plot(mu, type="l", ylab="change point = m")
```



We see a distinct change between the parameters λ and ϕ.

```
print(mean(mu))
```

```
## [1] 89.2438
```

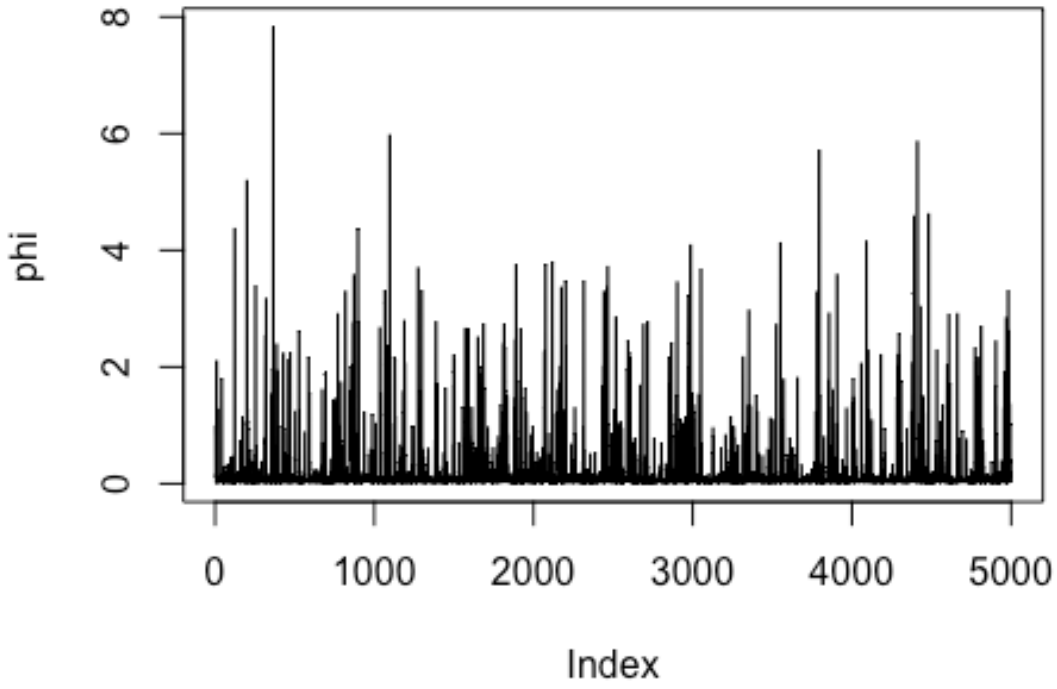
```
print(mean(lambda))
```

```
## [1] 2.329571
```

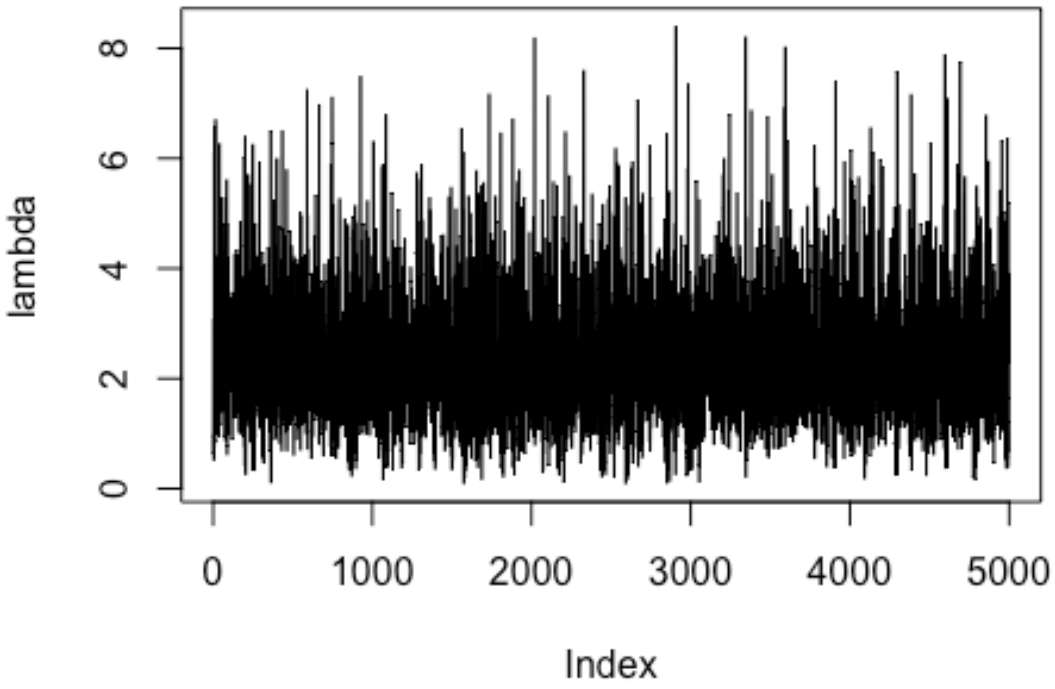
```
print(mean(phi))
```

```
## [1] 0.2024509
```

```
plot(phi, type="l", ylab="phi")
```

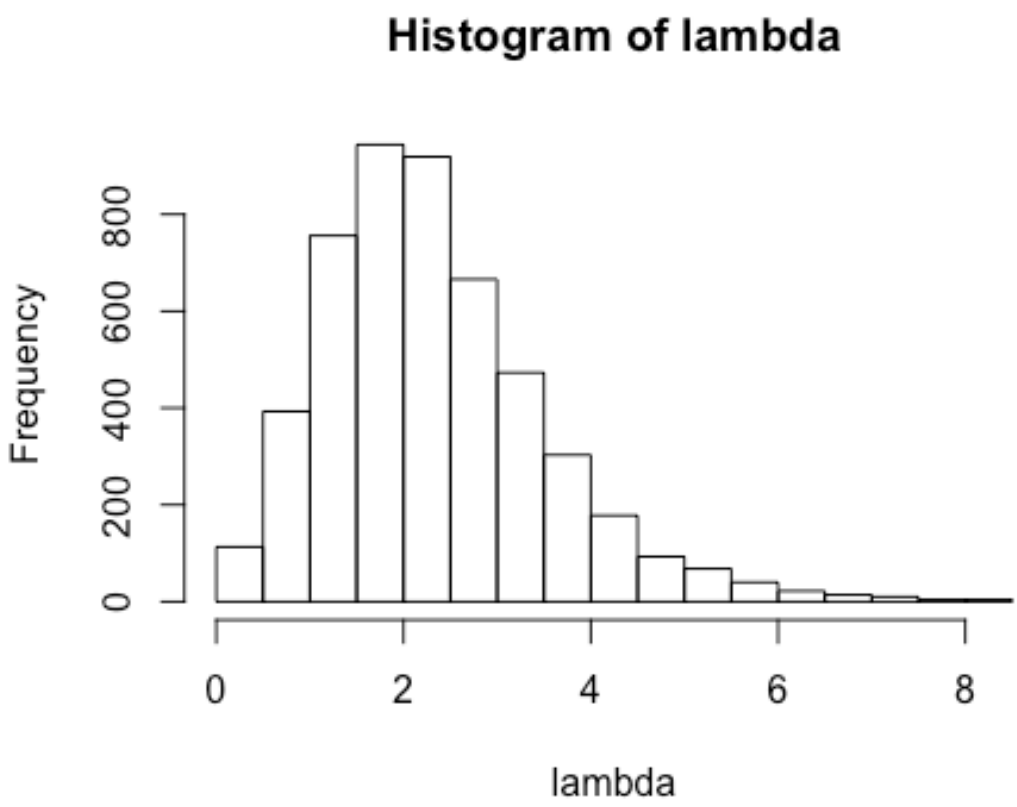



```
plot(lambda, type="l", ylab="lambda")
```

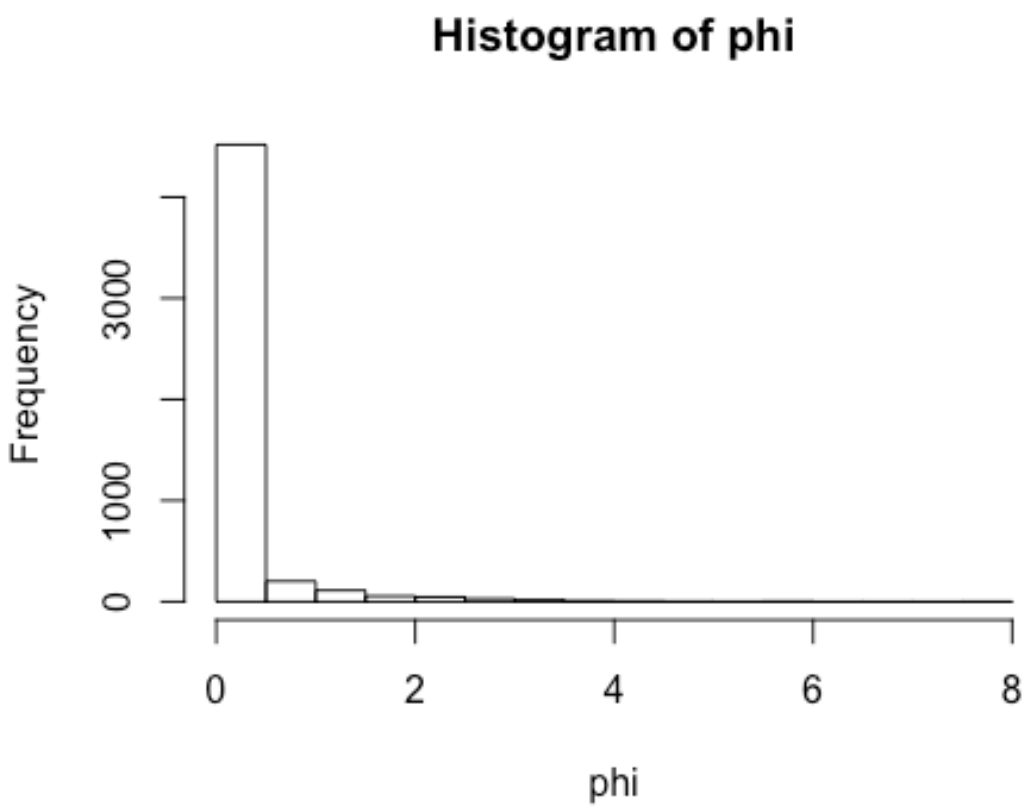


Now we will examine the distribution of the parameters.

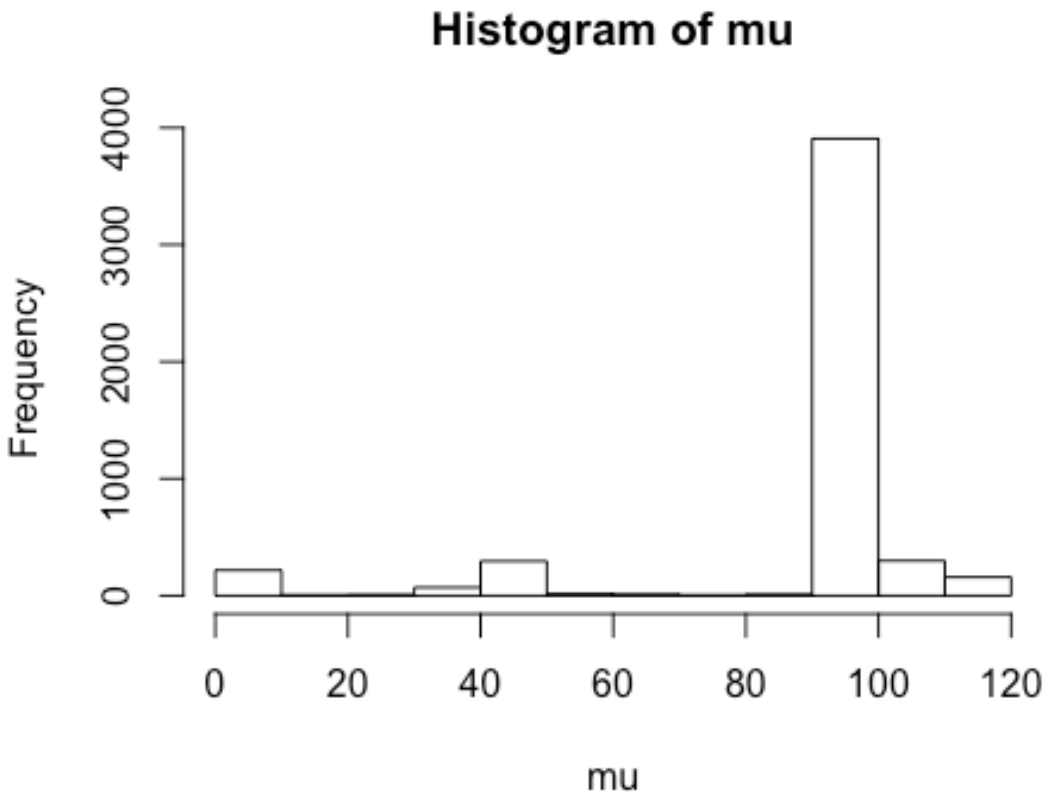
```
hist(lambda)
```


Histogram of lambda
Frequency
0
200
400
600
800
0
2
4
6
8
lambda

```
hist(phi)
```

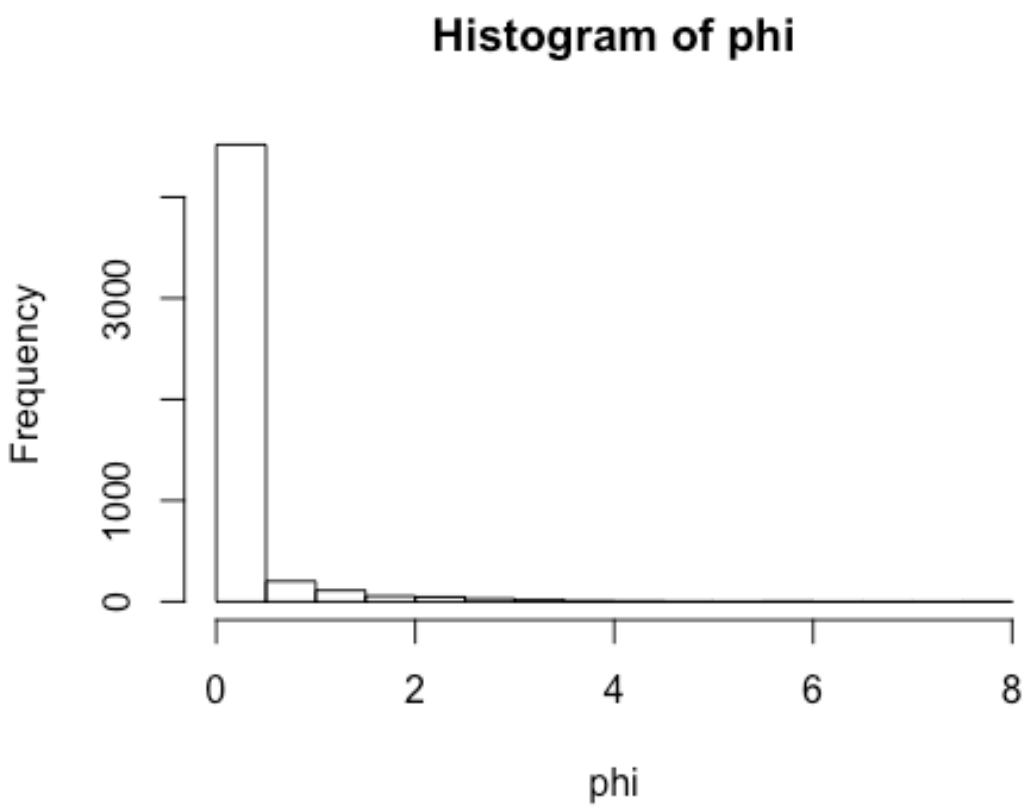



```
hist(mu)
```

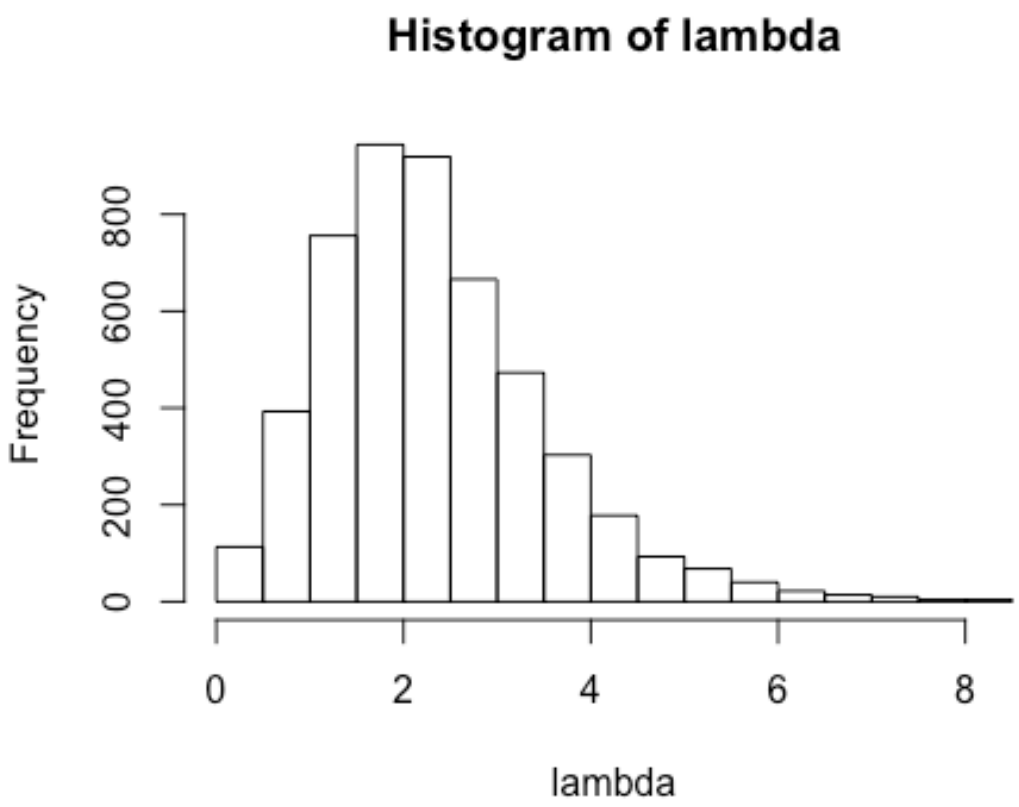


As we will see from the histograms, λ appears to be a skewed normal distribution, ϕ and μ seems to be highly centralized around the mean, with a very high variance.

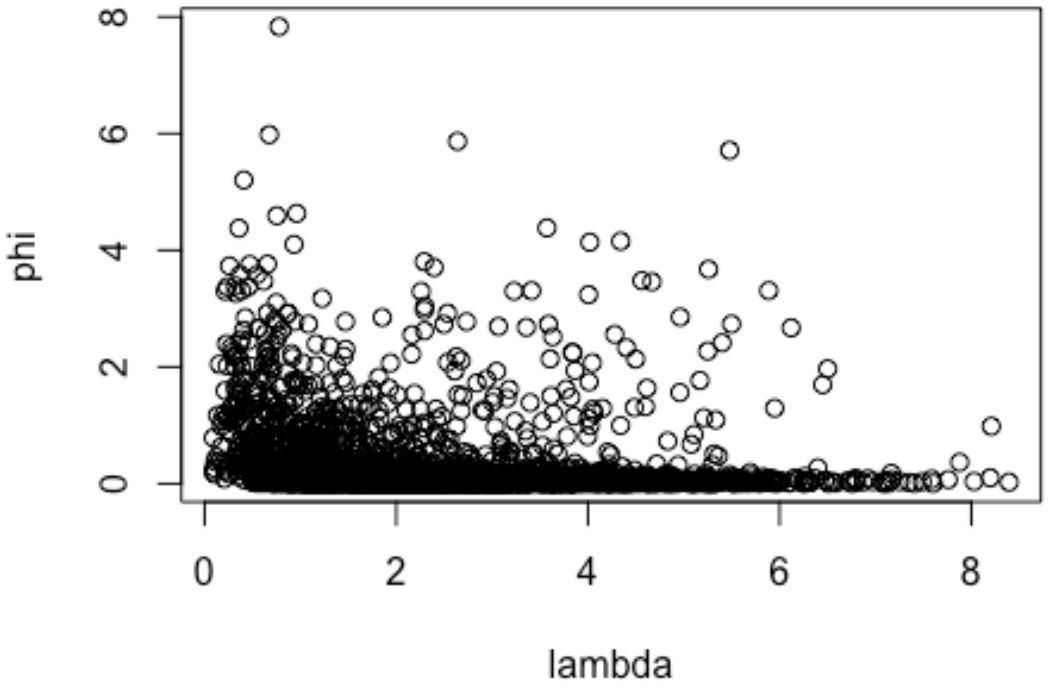
```
hist(phi)
```



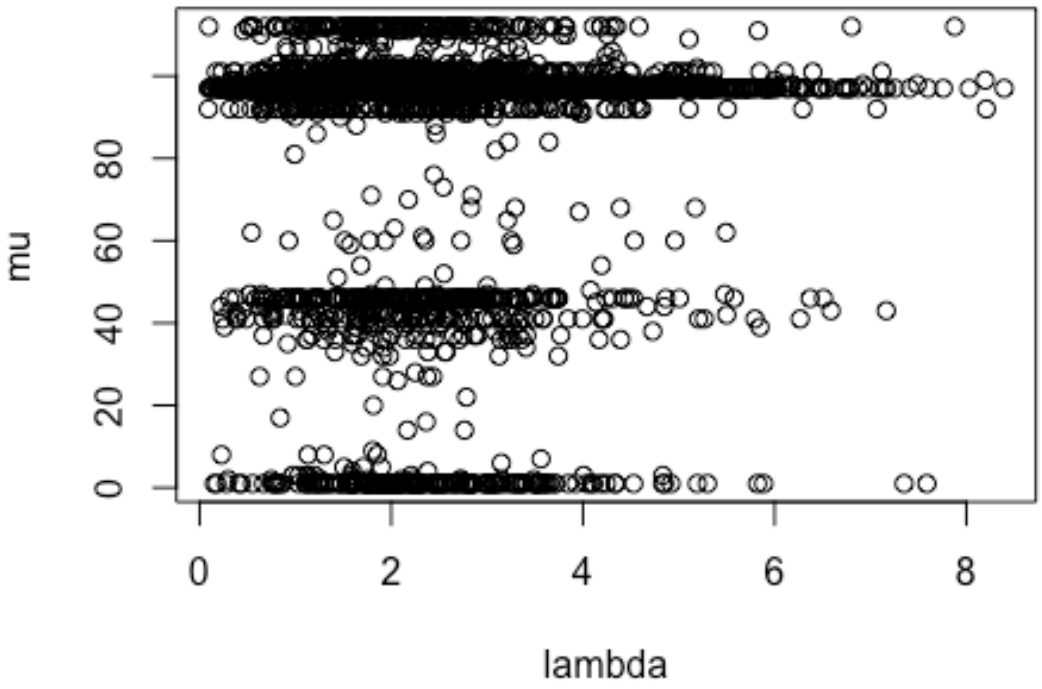

```
hist(lambda)
```



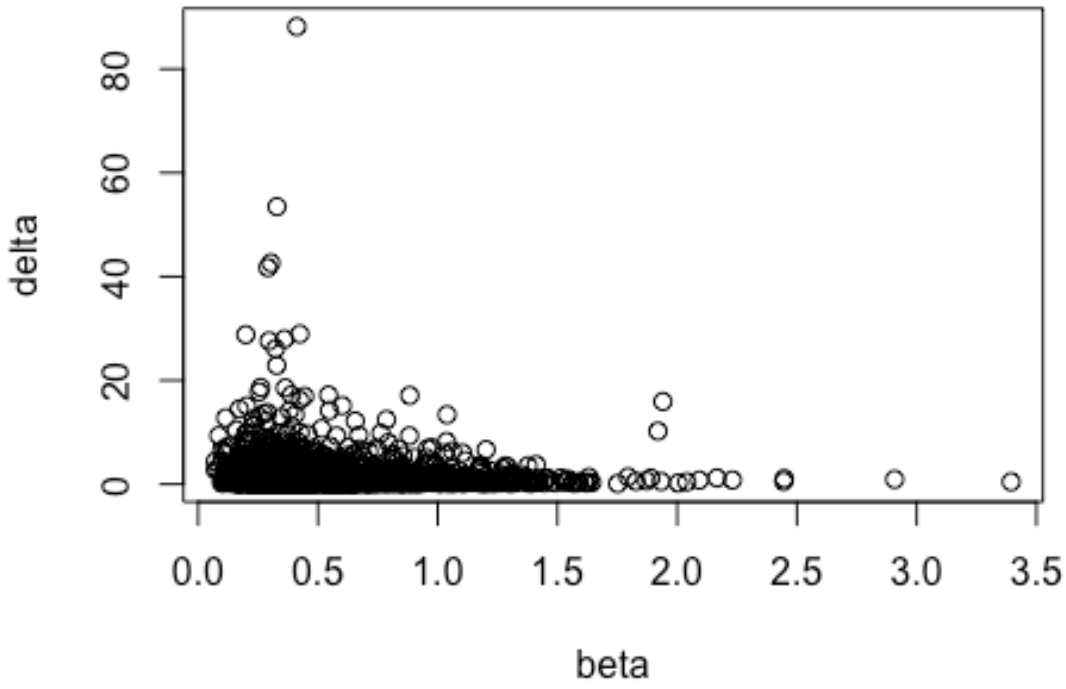

```
plot(lambda,phi)
```




```
plot(lambda,mu)
```

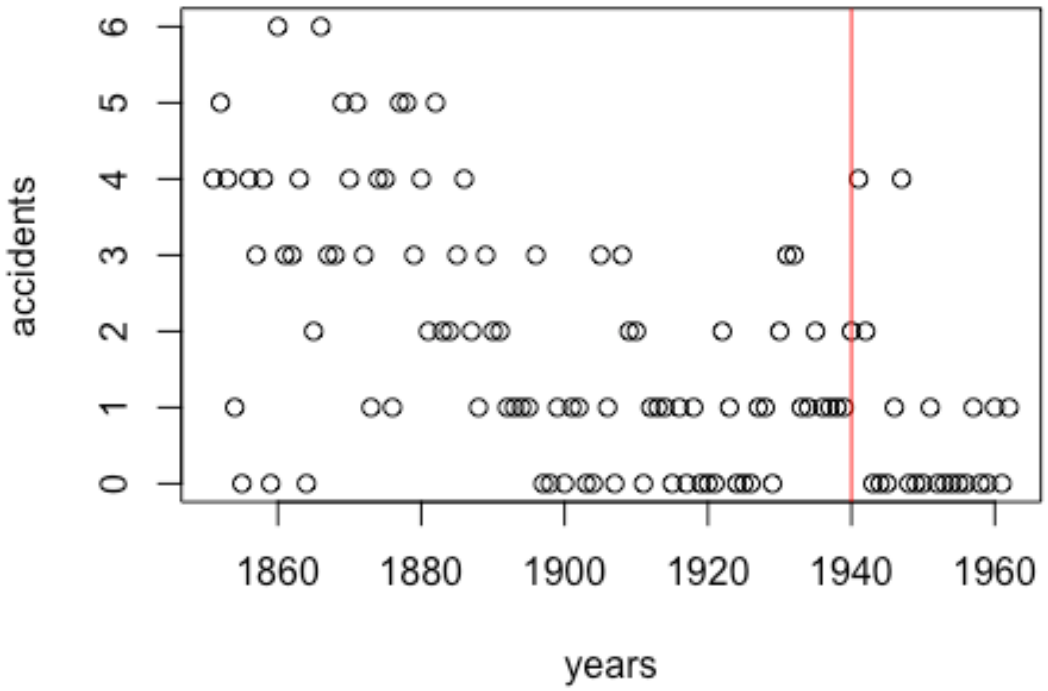



```
plot(beta,delta)
```



I believe the change point occurs at mean of our μ values,89.24. At this point, see a distinct change in our parameters ϕ and λ. The standard deviation of μ is: 24.12 Our 95% confidence interval is: 41, 137.49. If we plot the number of accidents by year, we can draw a vertical line at the change point and see how the average number of accidents changes. Right of the red line, the number of accidents per year seems to be much lower. From this, we can conclude that our results our consistent with the time-series of the observations.

```
years <- 1851:1962
change <- 1851+round(mean(mu),0)
plot(years,Y,ylab="accidents")
abline(v=change,col="red")
```



In Gibbs sampling we are able to sample from the joint distribution (the posterior distribution) by sampling from the full conditional distributions of each parameter. For this, we need to know the posterior distribution. The advantage to Metropolis-Hastings is that we do not need to know the posterior distribution. In this methods, we can choose the proposal distribution, but must be careful to check that the distribution is symmetric and to monitor the acceptance rate in the performance of the algorithm.

```
mean(mu[b:5000]) + 2*sd(mu[b:5000])
```

```
## [1] 137.4857
```

```
mean(mu[b:5000]) - 2*sd(mu[b:5000])
```

```
## [1] 41.01921
```

Problem 4

Implement a Metropolis-Hastings simulated annealing optimization to find a least cost solution to the traveling salesman problem (TSP). A sample problem is provided below, which is a 17x17 matrix of cost (C) between every pair of cities in a 17-city TSP problem.

```
C <- matrix(c(0, 633, 257, 91, 412, 150, 80, 134, 259, 505, 353, 324,
70, 211, 268, 246, 121, 633, 0, 390, 661, 227, 488, 572, 530, 555, 289,
282, 638, 567, 466, 420, 745, 518, 257, 390, 0, 228, 169, 112, 196,
154, 372, 262, 110, 437, 191, 74, 53, 472, 142, 91, 661, 228, 0, 383,
120, 77, 105, 175, 476, 324, 240, 27, 182, 239, 237, 84, 412, 227, 169,
383, 0, 267, 351, 309, 338, 196, 61, 421, 346, 243, 199, 528, 297, 150,
488, 112, 120, 267, 0, 63, 34, 264, 360, 208, 329, 83, 105, 123, 364,
35, 80, 572, 196, 77, 351, 63, 0, 29, 232, 444, 292, 297, 47, 150, 207,
332, 29, 134, 530, 154, 105, 309, 34, 29, 0, 249, 402, 250, 314, 68,
108, 165, 349, 36, 259, 555, 372, 175, 338, 264, 232, 249, 0, 495, 352,
95, 189, 326, 383, 202, 236, 505, 289, 262, 476, 196, 360, 444, 402,
495, 0, 154, 578, 439, 336, 240, 685, 390, 353, 282, 110, 324, 61, 208,
292, 250, 352, 154, 0, 435, 287, 184, 140, 542, 238, 324, 638, 437,
240, 421, 329, 297, 314, 95, 578, 435, 0, 254, 391, 448, 157, 301, 70,
567, 191, 27, 346, 83, 47, 68, 189, 439, 287, 254, 0, 145, 202, 289,
55, 211, 466, 74, 182, 243, 105, 150, 108, 326, 336, 184, 391, 145, 0,
57, 426, 96, 268, 420, 53, 239, 199, 123, 207, 165, 383, 240, 140, 448,
202, 57, 0, 483, 153, 246, 745, 472, 237, 528, 364, 332, 349, 202, 685,
542, 157, 289, 426, 483, 0, 336, 121, 518, 142, 84, 297, 35, 29, 36,
236, 390, 238, 301, 55, 96, 153, 336, 0),ncol=17)
```

First we will write a function that takes finds the total cost of a solution to the TSP. The function will take in a possible journey - a visit to each of the 17 cities - and the cost matrix. We will test this first by finding the total cost of a journey through all the cities in order, 1,2,3…17 and then a randomized journey through all 17.

```
cities <- 1:17
totalCost <- function(verticies,costMatrix){
     cost <- 0
     for (i in 1:(length(verticies)-1)){
     cost <- cost + costMatrix[verticies[i],verticies[i+1]]
     }
  return(cost)
}
totalCost(cities,C)
```

```
## [1] 4601
```

```
totalCost(sample(cities),C)
```

```
## [1] 4351
```

Now we will perform simulated annealing that takes three input parameters: the initial temperature T0, beta (determines annealing schedule), and number of iterations, and return two results: the solution vector x and the cost of the solution s.

```
annealing <- function(T0,beta,iterations){
    x <- sample(cities)
    Sx <- totalCost(x,C)
    xbest <- x
    sbest <- Sx
```

```
    results <- c()

for (i in 1:iterations){
    x <- sample(cities)
    I <- sort(x[1:2])
    Y <- c(x[1:(I[1]-1)],x[(I[2]-1):I[1]],x[I[2]:17])
    Sy = totalCost(Y,C)
    if (Sy < Sx){alpha <- 1} else {alpha <- exp(-(Sy-Sx)/T0)}
    u <- runif(1)
    if (u < alpha){
      x <- Y
      Sx <- Sy
      }
    T0 <- beta * T0
    xbest <- x
    sbest <- Sx
    results[i] <- sbest
    }
 solutions <- list("vector"=xbest,"cost"=sbest)
 return(solutions)
}
annealing(1,0.9999,10000)
```

```
## $vector
##  [1] 15 13  4  5  1  9 14  8 11 10  3 17  6  7 16  2 12
##
## $cost
## [1] 2670
```

```
totalCost(sample(cities),C)
```

```
## [1] 5129
```

As we can see from these two results, the cost of the minimum solution is far less than a random tour through all seven cities. Now we will repeat the cost of a random tour and the simulated annealing, four times each. We see the that the average minimum cost of the annealing process is far less than the average minimum cost of the random tour. The annealing process gives us a different solution vector and minium cost each time. The variance of the minimum cost seems to be quite low, making it a far better option than a random tour.

```
totalCost(sample(cities),C)
```

```
## [1] 4224
```

```
totalCost(sample(cities),C)
```

```
## [1] 5305
```

```
totalCost(sample(cities),C)
```

```
## [1] 3651
```

```
totalCost(sample(cities),C)
```

```
## [1] 4721
```

```
annealing(1,0.9999,10000)
```

```
## $vector
##  [1]  7 10  9  6  8  4 14 15  3 12  5 17  2 11  1 13 16
##
## $cost
## [1] 2609
```

```
annealing(1,0.9999,10000)
```

```
## $vector
##  [1]  2 17 16  8  4 11 15  1  9 10 14  6 13  3 12  5  7
##
## $cost
## [1] 2547
```

```
annealing(1,0.9999,10000)
```

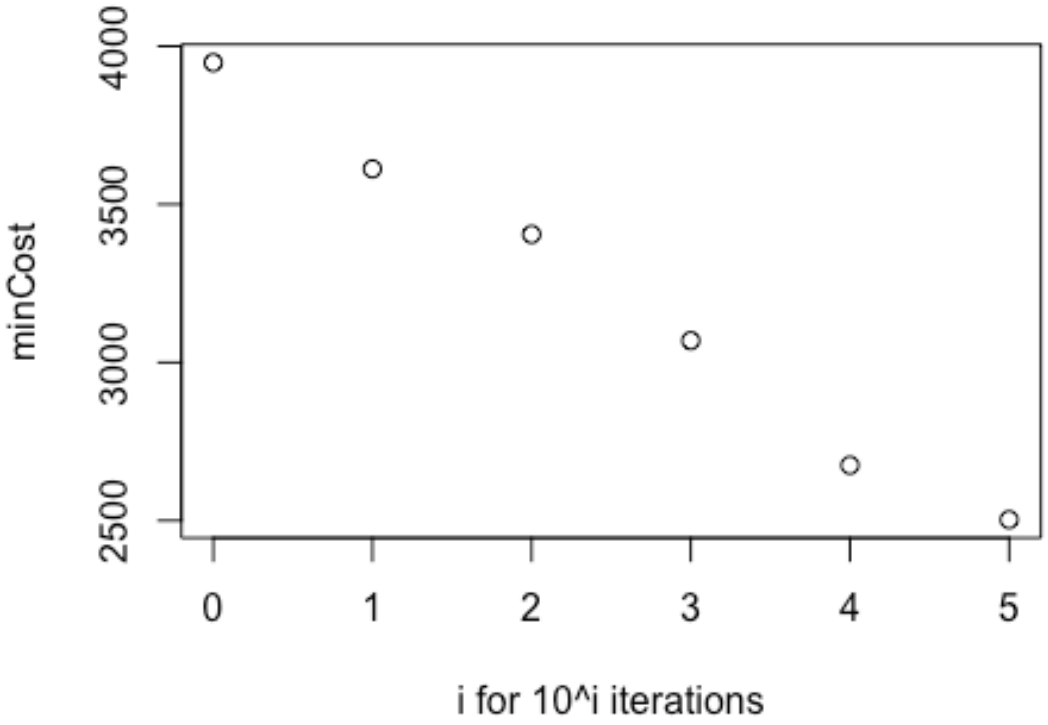
```
## $vector
##  [1]  7 15  9 14 11  3 12  6  5 13  4 10 16  1  2  8 17
##
## $cost
## [1] 2608
```

```
annealing(1,0.9999,10000)
```

```
## $vector
##  [1] 11 14 12  8  9 15  6 10 17  7  3  5  1 13 16  4  2
##
## $cost
## [1] 2533
```

Now we will plot the minimum cost for annealing process against the number of iterations. We will do this on a log scale, for 10^i iterations with i=0...5 iterations. We see that the minimum cost decreases greatly each time we raise the power. We have our lowest cost at 100,000 iterations. We can conclude from this example that the performance of the annealing simulations increases greatly with the number of iterations it runs.

```
power <- 0:5
minCost <- c()
for (i in 0:5){minCost[i+1] <-
as.numeric(annealing(1,0.9999,10^i)[[2]])}
plot(power,minCost,xlab="i for 10^i iterations")
```


minCost
4000
3500
3000
2500
0
1
2
3
4
5
i for 10^i iterations