IS604_charleyferrari_hw5

Charley Ferrari

October 27, 2015

Question 1

At the start of each week, the condition of a machine is determined by measuring the amount of electrical current it uses. According to its amperage reading, the machine is categorized as being in one of the following four states: low, medium, high and failed. A machine in the low state has a probability of 0.05, 0.03, and 0.02 of being in the medium, high, or failed state, respectively, at the start of the next week. A machine in the medium state has a probability of 0.09 and 0.06 of being in the high or failed state, respectively, at the start of the next week (it cannot, by itself, go to the low state). And, a machine in the high state has a probability of 0.1 of being in the failed state at the start of the next week (it cannot, by itself, go to the low or medium state). If a machine is in the failed state at the start of a week, repair is immediately begun on the machine so that it will (with probability 1) be in the low state at the start of the following week. Let X be a Markov chain where Xn is the state of the machine at the start of week n.

a. Give the Markov Transition matrix for X

```
P <- 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(P) <- c("L","M","H","F")
colnames(P) <- rownames(P)

P
```

```
##     L    M    H    F
## L 0.9 0.05 0.03 0.02
## M 0.0 0.85 0.09 0.06
## H 0.0 0.00 0.90 0.10
## F 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?

```
X0 <- matrix(c(1,0,0,0),nrow=1)
colnames(X0) <- colnames(P)

X0 %*% P %*% P %*% P
```

```
##          L        M        H      F
## [1,] 0.771 0.115875 0.085425 0.0277
```

There is a 0.0277 chance that a new machine starting in the low state will be in the failed state three weeks after it is new.

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

To model this, I'll change my underlying probability matrix to make the fail state an absorbing state. By doing this, I don't lose any information (If a machine that has previously failed will fail again, it will still be counted.)

```
Pc <- P
Pc["F","F"] <- 1
Pc["F","L"] <- 0

Pc
```

```
##     L    M    H    F
## L 0.9 0.05 0.03 0.02
## M 0.0 0.85 0.09 0.06
## H 0.0 0.00 0.90 0.10
## F 0.0 0.00 0.00 1.00
```

There's my new matrix, and below is how I'd calculate the probability of failing at least once three weeks after it's new

```
X0 %*% Pc %*% Pc %*% Pc
```

```
##          L        M        H      F
## [1,] 0.729 0.114875 0.084825 0.0713
```

So there's a 0.0713 chance of ending up failing at least once.

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

First, I'm going to define a function to make it easy to get the "power" of a matrix:

```
matpower <- function(M,p){
  Mret <- M
  for(i in 2:p){
    Mret <- M %*% Mret
  }
  return(Mret)
}
```

Now, I can use the same tweak I made before, making F an absorbing state

I'm also going to use a another trick, defining a horizontal vector V_F, which will give me the percentage at the nth period of being in a filed state. Given the condition that the failed state is absorbing, this will be the probability of a machine starting out as low will have had its first failure by the nth stage.

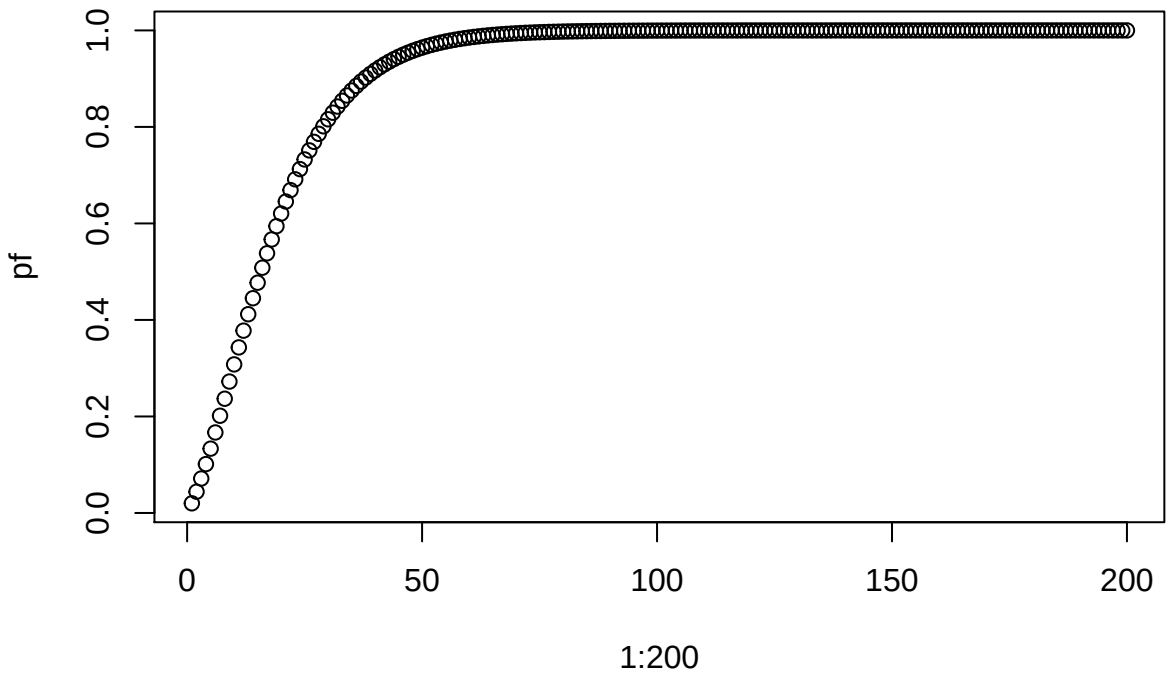
So, the probability of having had a first failure by n=4 (with the initial state as n=0) is:

```
n <- 4
vf <- matrix(c(0,0,0,1),nrow=4)

X0 %*% matpower(Pc,4) %*% vf
```

```
##          [,1]
## [1,] 0.101255
```

Now I'm going to calculate a vector of the probability of success at each stage. Because fail is an absorbing state, this will be the probability of the first failure having not yet occurred at that stage

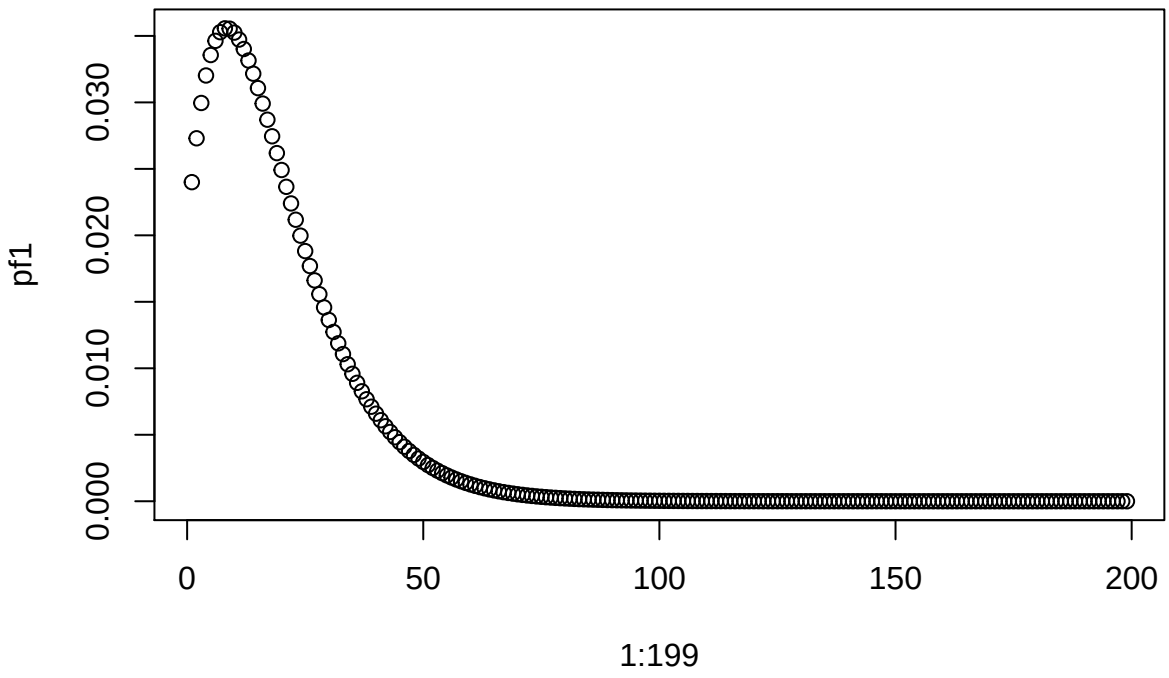
```
pf <- X0 %*% Pc %*% vf

for(i in 2:200){
  pf <- c(pf, X0 %*% matpower(Pc,i) %*% vf)
}

plot(1:200,pf)
```



As expected, the probability of remaining having the first failure at or before state n increases assymptotically towards 1.

Now, I'm going to subtract each fail probability by the probability of failure at the $n-1^{th}$ stage. This will give me the probability that the failure occurred at the n^{th} stage.

```
pf1 <- tail(pf,-1)-head(pf,-1)

plot(1:199,pf1)
```



Starting at stage n=1, pf1 is a vector of of the probabilities that the first failure occurs at a particular stage. This plot represents a discrete probability distribution. The shape makes sense intuitively: the probability starts as increasing. The probability of failing isn't that high, and it's more likely to take some time. Then, it starts decreasing, since the probability of continued success is constantly falling.

As a sanity check, the sum of the probabilities of this discrete distribution should add up to 1 for an infinite number of states. Lets test that out:

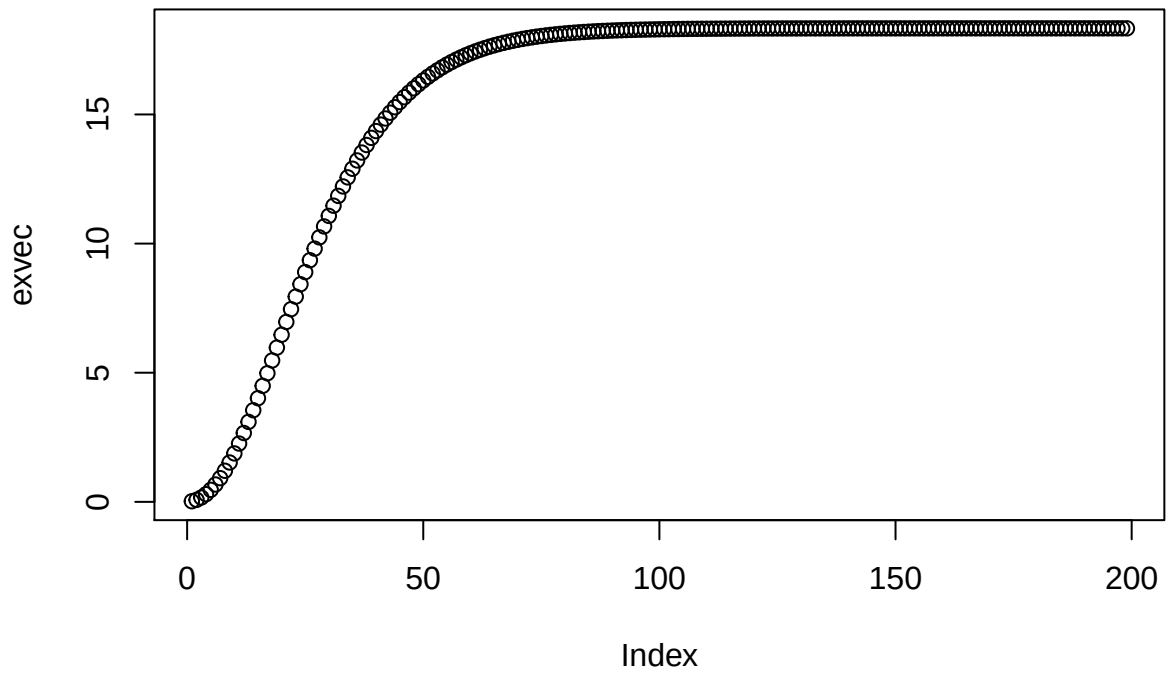
```
sum(pf1)
```

```
## [1] 0.98
```

That's close enough, and would be higher n's.

The expected value is simply $\sum p_i x_i$. This sum is infinite, but I can come to a reasonable approximation of where this is converging by calculating its value for the 200 probabilities I have:

```
exvec <- cumsum(1:199*pf1)

plot(exvec)
```



This definitely seems to be converging, lets find out where:

```
tail(exvec,1)
```

```
## [1] 18.33333
```

It looks like this is converging at around $18\frac{1}{3}$ weeks.

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

I'm going to assume we're talking about a machine starting low, and running for 52 weeks.

I'll go back to using the original P matrix. For this problem, the probability at each stage matters. Since we're using the original matrix, a machine that has failed one week will be in the low position the next week with a probability of 1, and we'd like to record the fact that at that week the machine was in the filed state. To do this, I'll run through 52 iterations of this markov chain to get a collection of probabilities of failed states. I just need to find the average of these, each week there's a probability of that week being a failed state. If the probability of being failed in a certain week n is 1, then that would be 1 week out of 52 that the machine was failed.

```
pf <- c(0,X0 %*% P %*% vf)

for(i in 2:51){
  pf <- c(pf,X0 %*% matpower(P,i) %*% vf)
}

sum(pf)
```

```
## [1] 2.321404
```

This means on average, if you start a machine in the low state, it will spend 2.32 weeks in the failed state.

f. Each week that the machine is in the low state, a profit of 1000 is realized. Each week that the machine is in a medium state, a profit of 500 is realized. each week that the machine is in a 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?

For this we'll need to figure out the long run matrix. I could do this by finding the stationary distribution π such that $\pi = \pi P$, but since I'm in R I'll just run my matpower function for a few iterations to get a steady state:

```
Psteady <- matpower(P,1000)

Xf <- X0 %*% Psteady
```

Given any X0 where the probabilities add up to 1, we'll end up with the steady state distribution.

Now all we need to do is multiply the profits by the probabilities to get the long run average profiit per week:

```
profit <- matrix(c(1000,500,400,-700),nrow=4)

ExpectedProfit <- Xf %*% profit

ExpectedProfit
```

```
##          [,1]
## [1,] 657.377
```

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?

Lets find out what the new expected profit would be in this case. First, I'll have to create a new P matrix. In this matrix, there's still a chance that the machine will be in a failed state (coming from the low or the medium state), but there is 0 probability of the matrix entering failed state from a high state. At high state, the machine will be brought back to low with probability of 1.

This means the third row of the matrix will change. Lets create our new P matrix, and determine it's steady state distribution, and then multiply it by a new matrix of profits to get our answer:

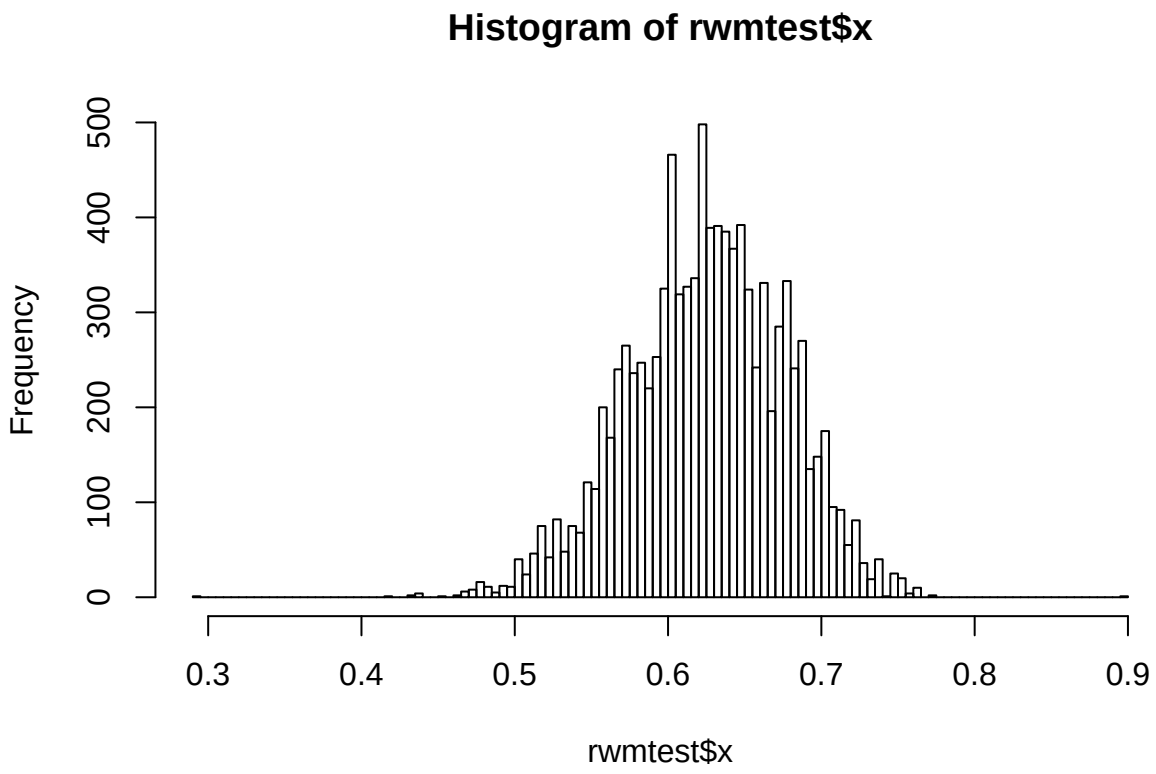
```
Pnew <- matrix(c(0.9,0,1,1,0.05,0.85,0,0,
              0.03,0.09,0,0,0.02,0.06,0,0),nrow=4)

Pnewsteady <- matpower(Pnew,1000)

Xnewf <- X0 %*% Pnewsteady

profitnew <- matrix(c(1000,500,-600,-700),nrow=4)

ExpectedProfitNew <- Xnewf %*% profitnew

ExpectedProfitNew
```

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

It does make sense to start repairing machines when they enter the high state, because it means that many more machines will be caught before they enter the failed state, and will be fixed more cheaply.

Question 2

```
rwm <- function(x0,N){
  x <- numeric(N)
  x[1] <- x0
  u <- runif(N)
  count <- 0
  k <- c(125,18,20)
  for(i in 2:N){
    y <- runif(1)
    if(
      u[i] <=
      (((2+y)^k[1] * (1-y)^sum(k[2],k[3]) * y^(197-sum(k)))/
      ((2+x[i-1])^k[1] * (1-x[i-1])^sum(k[2],k[3]) * x[i-1]^(197-sum(k))))
    ){
      x[i] <- y
    }else{
      x[i] <- x[i-1]
      count <- count+1
    }
  }
  return(list(x=x,count=count))
}

rwmtest <- rwm(0.9,10000)

hist(rwmtest$x,breaks=100)
```




```
mean(rwmtest$x)
```

```
## [1] 0.6251907
```

Question 3

```
##### INCOMPLETE #####


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)

ny <- length(Y)
n <- 1000

# Order of stats = lambda, phi, m, beta, delta

stats <- matrix(numeric(5*n),nrow=n)
Mprob <- numeric(n)
stats[1,3] <- Mprob[1] <- sample(1:(ny-1), 1)
sum1 <- sum(Y[1:Mprob[1]])
```

```
sum2 <- sum(Y[Mprob[1]+1:length(Y)])

alpha <- 0.5
gam <- 0.5

stats[1,1] <- 1
stats[1,2] <- 1
stats[1,4] <- 1/rgamma(alpha,stats[1,1]+1)
stats[1,5] <- 1/rgamma(gam,stats[1,2]+1)



for(i in 1:n){
  sum1 <- sum(Y[1:m])
  sum2 <- sum(Y[m+1:length(Y)])
  Mprob[i,4] <- rigamma()
  Mprob[i,1] <- rgamma(1, shape = sum1+alpha, b)
}

##### INCOMPLETE #######
```

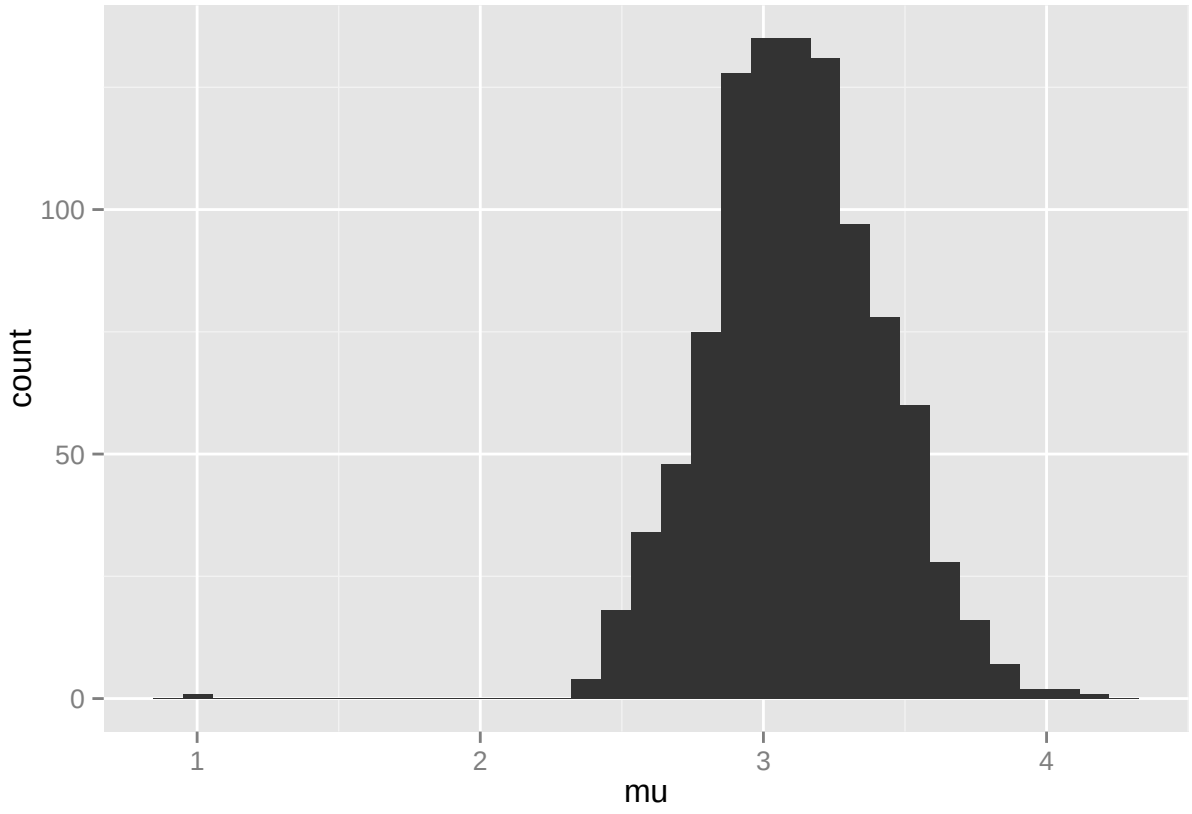
Questions are answered from the code given in statistical computing in R on page 275, edited in a minor way to allow vectors of b1 and b2, which are the book's versions of beta and delta.

I didn't understand how you were implementing alpha and gamma. The code would have been similar, I unfortunately ran out of time to wrap my head around the specifics you were going for in your example.

```
library(boot)
data(coal)
year <- floor(coal)
y <- table(year)
#plot(y)

y <- floor(coal[[1]])
y <- tabulate(y)
y <- y[1851:length(y)]


n <- length(y) #length of the data
m <- 1000 #length of the chain
mu <- lambda <- k <- numeric(m)
L <- numeric(n)
k[1] <- sample(1:n, 1)
mu[1] <- 1
lambda[1] <- 1
b1 <- 1
b2 <- 1

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

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

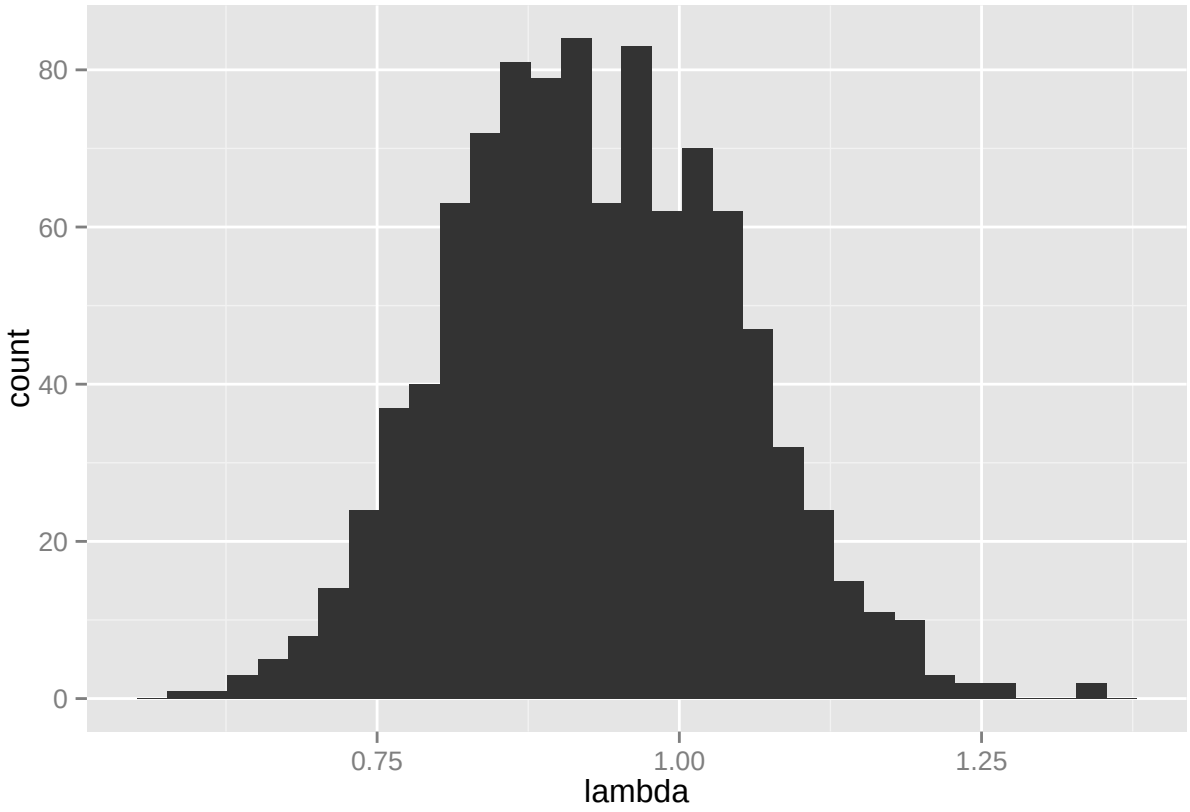
```
  #generate mu
  r <- .5 + sum(y[1:kt])
  mu[i] <- rgamma(1, shape = r, rate = kt + b1[i-1])
  #generate lambda
  if (kt + 1 > n) r <- .5 + sum(y) else
    r <- .5 + sum(y[(kt+1):n])
  lambda[i] <- rgamma(1, shape = r, rate = n - kt + b2[i-1])
  #generate b1 and b2
  b1 <- c(b1,rgamma(1, shape = .5, rate = mu[i]+1))
  b2 <- c(b2,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 distribution L on 1:n
  k[i] <- sample(1:n, prob=L, size=1)
}

library(ggplot2)

# Histogram of lambda:

qplot(mu)
```

```
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
```

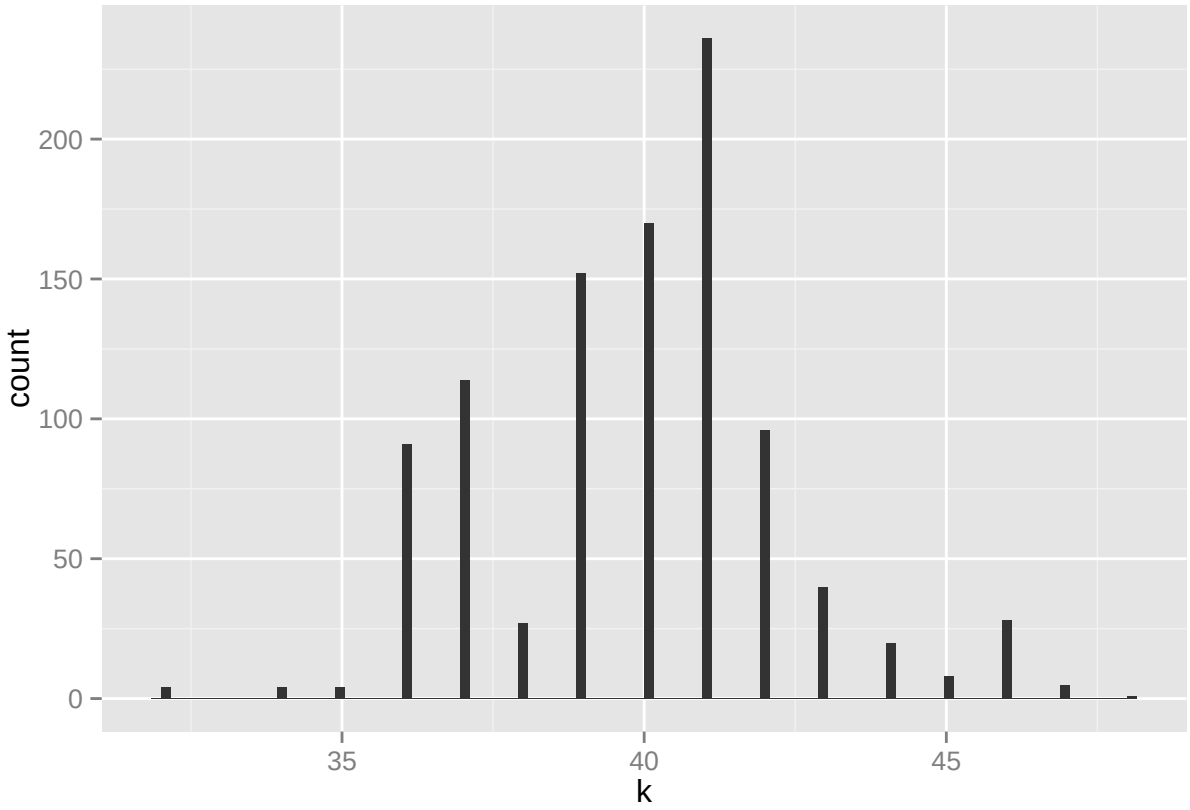



```
# Histogram of phi:

qplot(lambda)
```

```
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
```



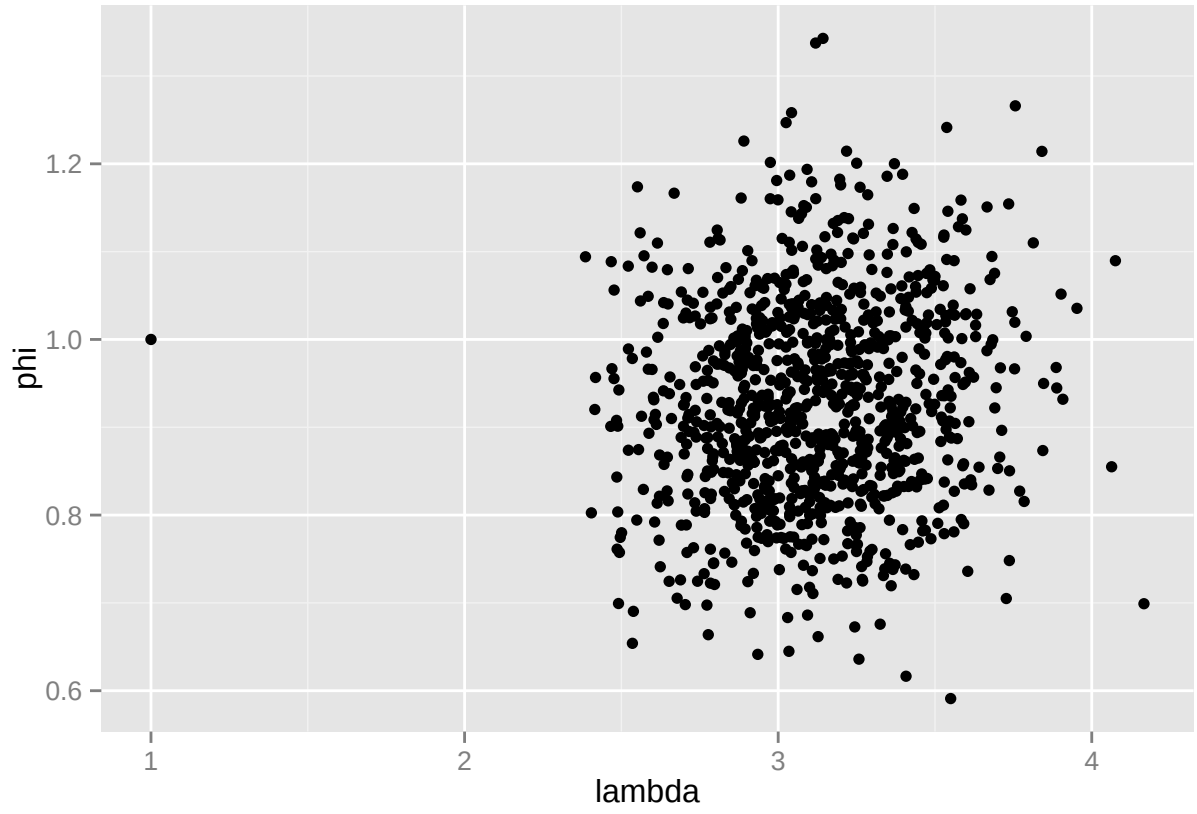

```
# Histogram of M:

qplot(k, binwidth=(range(k)[2]-range(k)[1])/100)
```



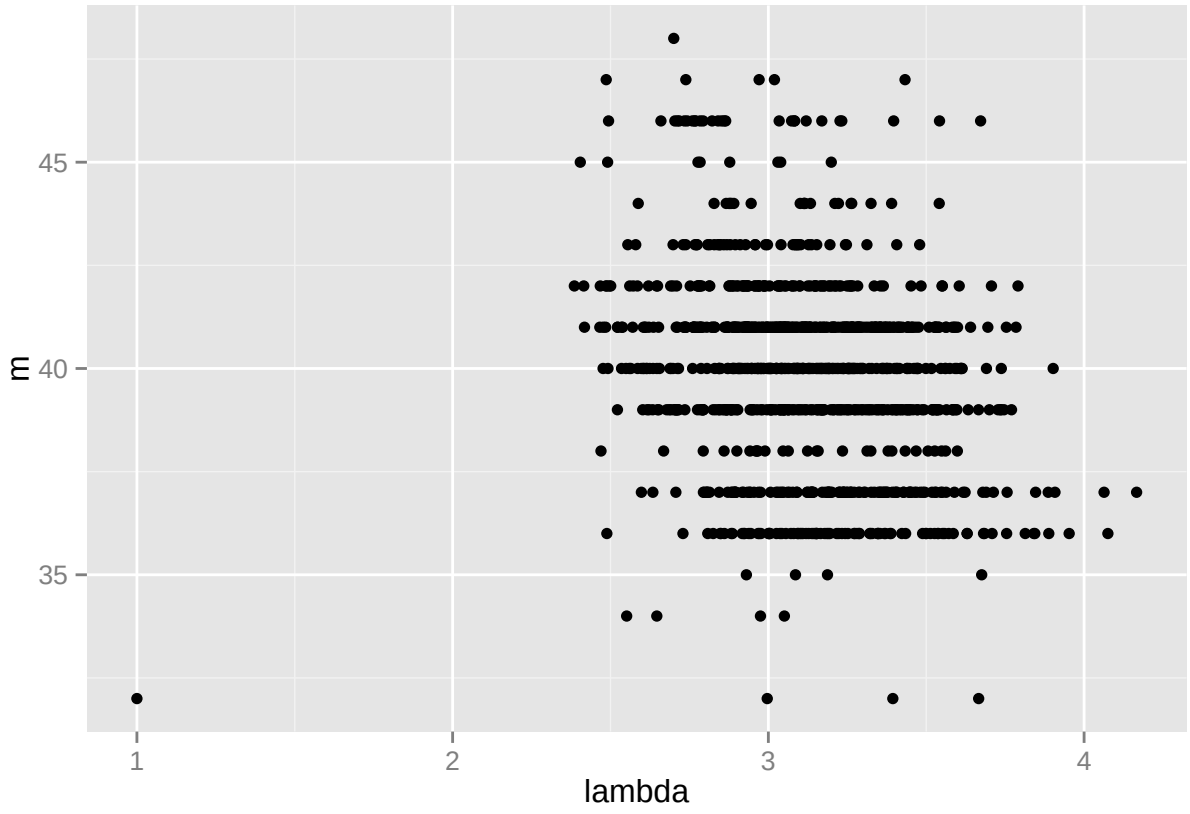

```
mean(k)
```

```
## [1] 39.891
```

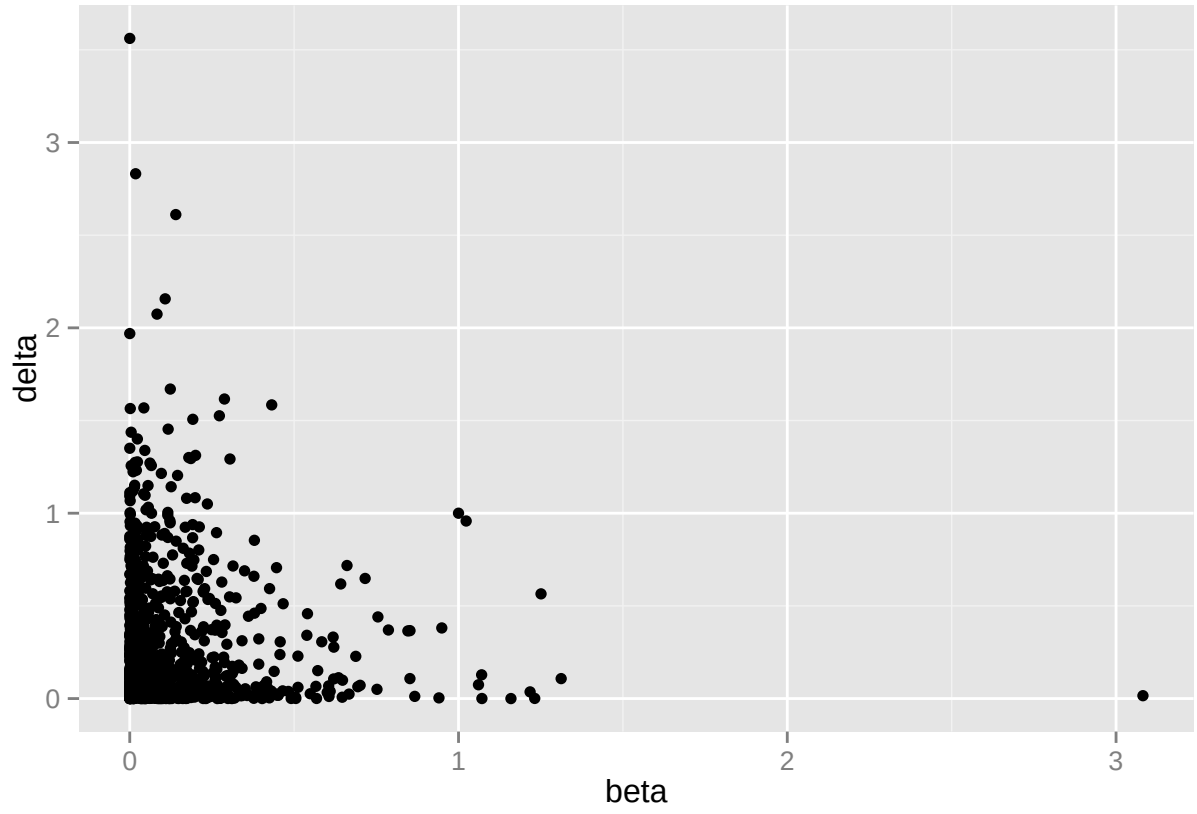
```
# Plot of lambda vs phi

ggplot(data.frame(lambda=mu, phi=lambda), aes(x=lambda,y=phi)) + geom_point()
```




```
# Plot of lambda vs m

ggplot(data.frame(lambda=mu, m=k), aes(x=lambda,y=m)) + geom_point()
```




```
# Plot of beta vs delta

ggplot(data.frame(beta = b1, delta = b2),aes(x=beta, y=delta)) + geom_point()
```



b.

When do you think the change occurred, based on your result?

I would guess the change occurred in year 40. Since the first year is 1851, this would mean the change occurred in 1891.

What were the average rates of coal mining accidents before and after the change?

```
averagebefore <- mean(y[1:40])
averageafter <- mean(y[41:length(y)])

averagebefore
```

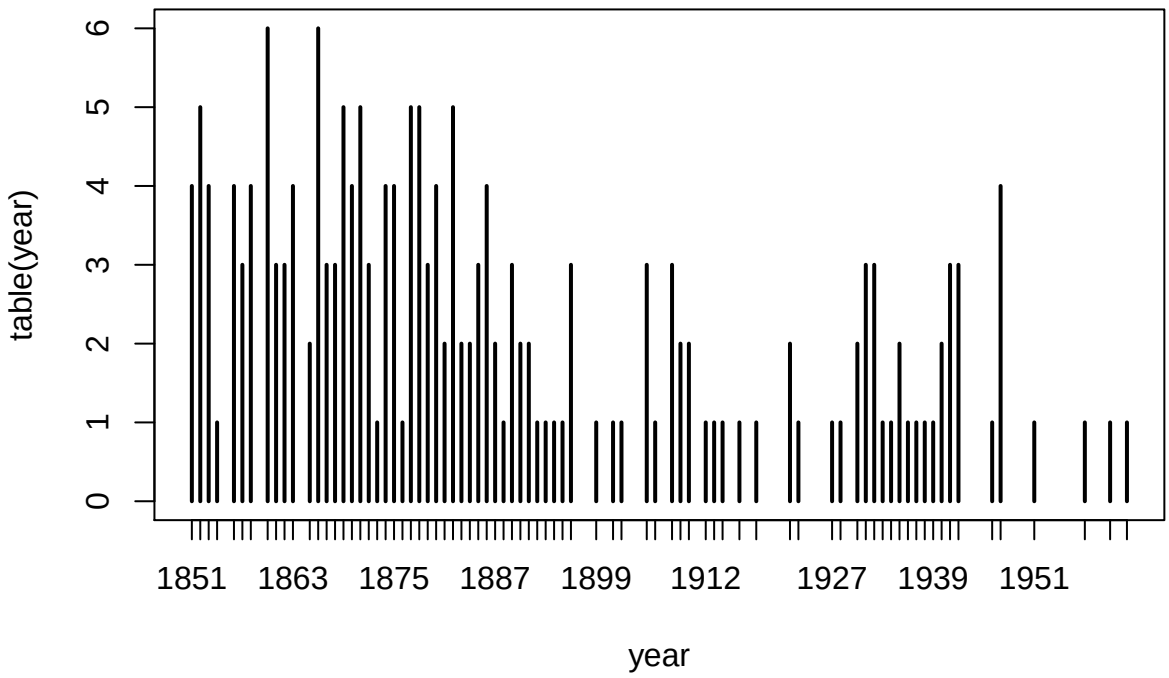
```
## [1] 3.125
```

```
averageafter
```

```
## [1] 0.9166667
```

Are the results consistant with the time series of the observations? Why or why not?

```
plot(table(year))
```



This does look plausable when looking at the plot of the data.

c. How is Gibbs sampling diffferent from the Metropolis-Hastings approach?

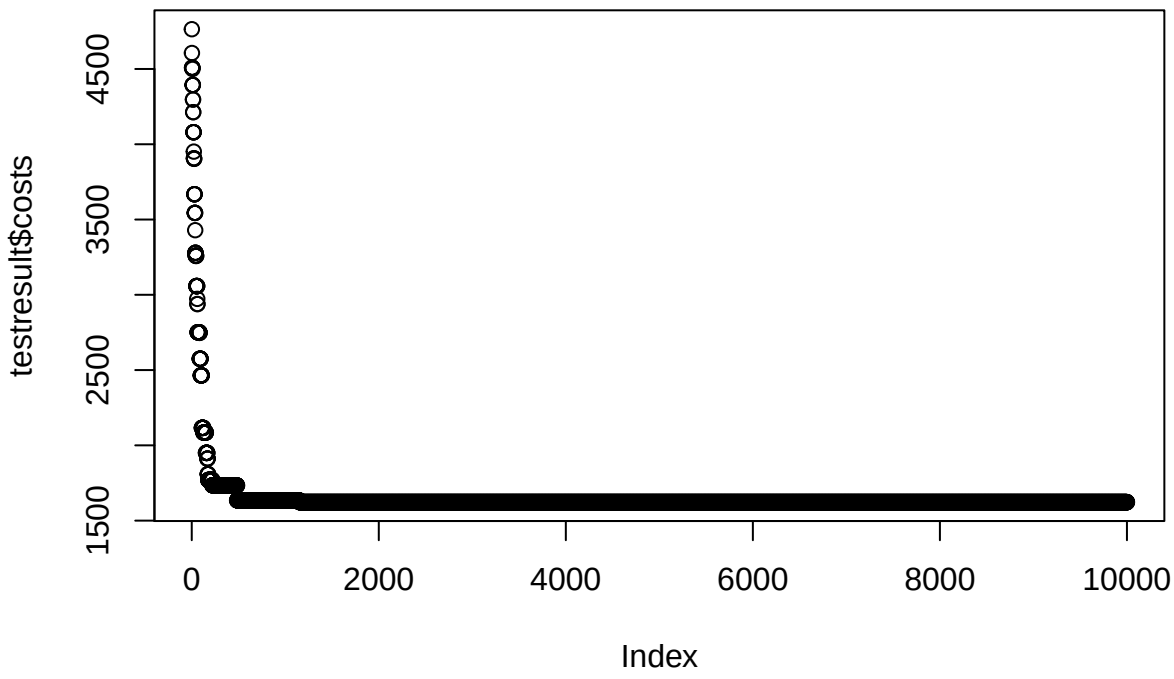
Gibbs sampling is a specific application of the Metropolis-Hastings algorithm, and works best for multiple parameter questions such as these. The metropolis algorithm works when we can randomly walk through a single parameter space. With this, we needed to walk through the parameter space of several variables concurrently, in order to find out what was happening with the change date.

Question 4

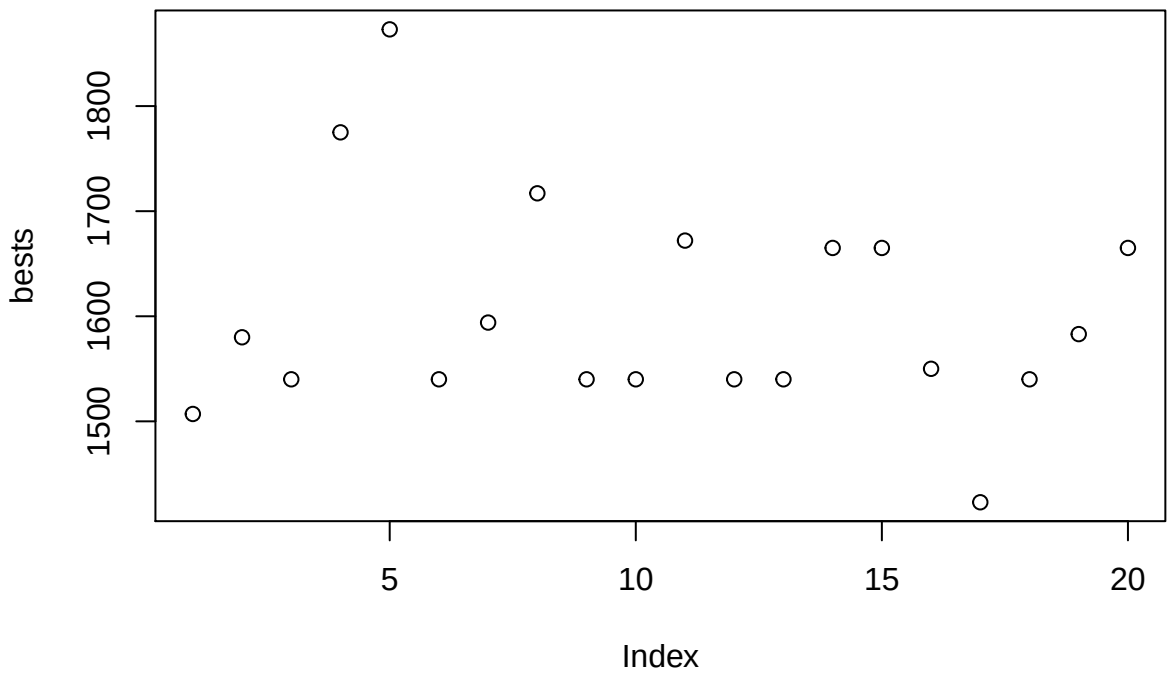
```
C <- matrix(c(0,633,257,91,412,150,80,134,259,505,353,324,70,211,268,246,121,
              633,0,390,661,227,488,572,530,555,289,282,638,567,466,420,745,518,
              257,390,0,228,169,112,196,154,372,262,110,437,191,74,53,472,142,
              91,661,228,0,383,120,77,105,175,476,324,240,27,182,239,237,84,
              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,
              46,745,472,237,528,364,332,349,202,685,542,157,289,426,483,0,336,
              121,518,142,84,297,35,29,36,236,390,238,301,55,96,153,336,0),
            nrow=17)

costfun <- function(x,C){
  cost <- 0
  for(i in 1:(length(x)-1)){
    cost <- cost + C[x[i],x[i+1]]
  }
  return(cost)
}

anneal <- function(T0, beta, N){
  temp <- T0
  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,
              46,745,472,237,528,364,332,349,202,685,542,157,289,426,483,0,336,
              121,518,142,84,297,35,29,36,236,390,238,301,55,96,153,336,0),
            nrow=17)
  n <- dim(C)[1]
  x <- sample(1:17,17)
  sx <- costfun(x,C)
  xbest <- x
  sbest <- sx
  results <- numeric(N)

  for(i in 1:N){
    x1 <- sample(1:17,17)
    I <- sort(c(x1[1],x1[2]))
    if(I[2] == 17){
      y <- c(x[1:I[1]-1],x[I[2]:I[1]])
    }else{
      y <- c(x[1:I[1]-1],x[I[2]:I[1]],x[(I[2]+1):length(x)])
    }
    sx <- costfun(x,C)
    sy <- costfun(y,C)
    if(sy < sx){
      alpha <- 1
```

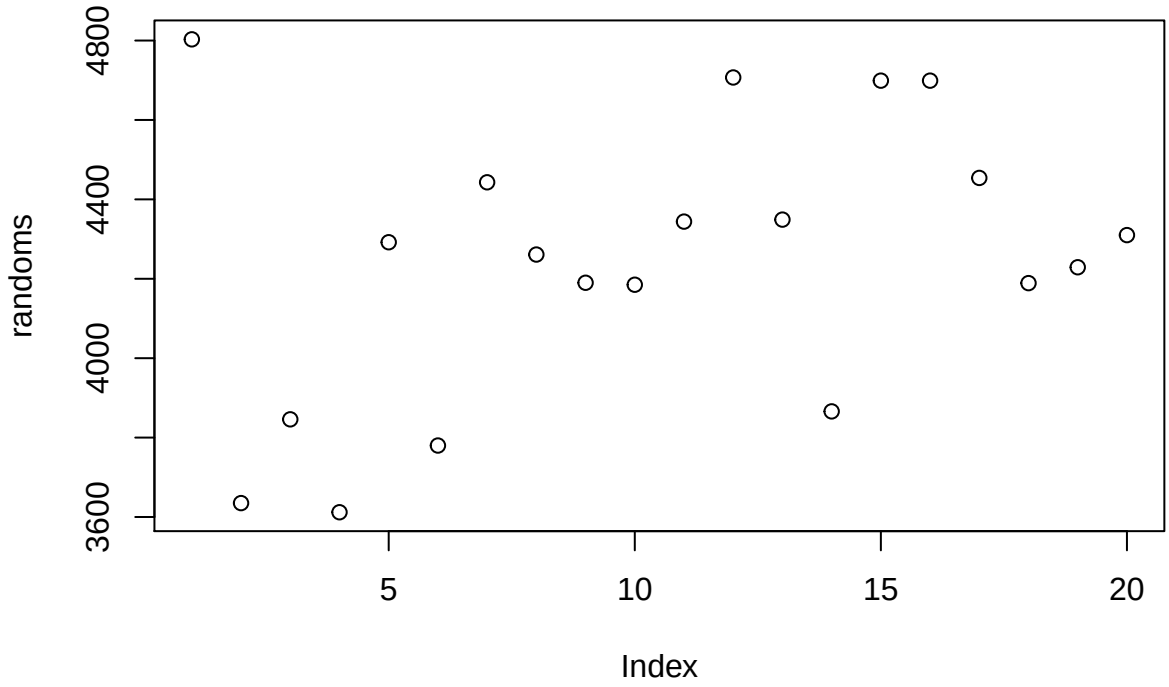
```
    }else{
      alpha <- exp(-(sy-sx)/temp)
    }
    U <- runif(1)
    if(U < alpha){
      x <- y
      sx <- sy
    }
    temp <- beta * temp
    xbest <- x
    sbest <- sx
    results[i] <- sbest
  }

  return(list(x=xbest, costs = results))
}

T0 <- 1
beta <- 0.9999
n <- 10000
x0 <- sample(1:17,17)

testresult <- anneal(T0, beta, n)
plot(testresult$costs)
```



```
bests <- c()
for(i in 1:20){
  bests <- c(bests,anneal(T0, beta, n)$costs[n])
}

plot(bests)
```




```
randoms <- c()
for(i in 1:20){
  randoms <- c(randoms,costfun(sample(1:17,17),C))
}

plot(randoms)
```



There does appear to be some variation in the best routes, but they do seem to be a remarkable improvement over random paths.