Homework 5

DATA604 Simulation and Modeling

Daniel Dittenhafer

April 10, 2016

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) Given the Markov transition matrix for X.

The following code creates and prints the Markov transition matrix for X.

```
tmX <- matrix(c(0.90, 0.05, 0.03, 0.02,
                0, 0.85, 0.09, 0.06,
                0, 0.00, 0.90, 0.10,
                1, 0.00, 0.00, 0.00), nrow=4, byrow=TRUE)
kable(tmX)
```

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

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?

First we define a helper function:

```
markovChainStep <- function(initialState, tranMatrix, steps)
{
  n <- steps
  step <- initialState
  for(i in 1:n)
  {
    step <- step %*% tranMatrix
  }
  return (step)
}
```

Next we call the helper function to compute the requested probability. We step twice in the function as it is 2 steps beyond the initial probability state.

```
bStep <- markovChainStep(tmX, tmX, 2)
kable(bStep)
```

0.771	0.115875	0.085425	0.0277
0.114	0.617125	0.208575	0.0603
0.180	0.005000	0.732000	0.0830
0.830	0.087500	0.058500	0.0240

The probability that the machine is in the failed state 3 weeks after it is new is 0.0277.

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

Compute the steady state probabilities for this recurrent transition matrix.

```
steadyState <- markovChainStep(tmX, tmX, 100)
steadyState
```

```
##           [,1]      [,2]     [,3]       [,4]
## [1,] 0.4918033 0.1639344 0.295082 0.04918033
## [2,] 0.4918033 0.1639344 0.295082 0.04918033
## [3,] 0.4918033 0.1639344 0.295082 0.04918033
## [4,] 0.4918033 0.1639344 0.295082 0.04918033
```

```
steadyState[1,4]
```

```
## [1] 0.04918033
```

```
Fm <- matrix(c(1, 1, 1, 1,
               1, 1, 1, 1,
               1, 1, 1, 1,
               1, 1, 1, 1), byrow=TRUE, nrow=4)
Fm
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    1    1    1
## [2,]    1    1    1    1
## [3,]    1    1    1    1
## [4,]    1    1    1    1
```

```
Rm <- matrix(nrow=4, ncol=4)
for(i in 1:nrow(Fm))
{
  for(j in 1:ncol(Fm))
  {
    if(i == j)
    {
      Rm[i,j] <- 1 / (1 - Fm[j,j])
    }
    else
    {
      Rm[i,j] <- Fm[i,j] / (1 - Fm[j,j])
    }
  }
}

Rm
```

```
##      [,1] [,2] [,3] [,4]
## [1,]  Inf  Inf  Inf  Inf
## [2,]  Inf  Inf  Inf  Inf
## [3,]  Inf  Inf  Inf  Inf
## [4,]  Inf  Inf  Inf  Inf
```

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

```
1 / steadyState[1,4]
```

```
## [1] 20.33333
```

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

Using the steady state probabilities, we sum the probabilities of being in the working states and multiply by 52 weeks to determine on average how many weeks per year the machine is working.

```
sum(steadyState[1,1:3]) * 52
```

```
## [1] 49.44262
```

f) Each week that the machine is in the low state...

a profit of $1000 is realized; each eweek that the machine is in the medium state, a profit of $500 is realized; each week tha the machine is in the high state, a profit of $400 is realized... What is the long-run average profit per week realized by the machine?

```
PL <- c(1000, 500, 400, -700)
PL
```

```
## [1] 1000  500  400 -700
```

```
lrAvgProfit <- PL %*% steadyState[1, ]
lrAvgProfit
```

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

g) Policy change?

Define the new profit and loss values:

```
newPL <- c(1000, 500, -600, -700)
newPL
```

```
## [1] 1000  500 -600 -700
```

Define the new transition matrix:

```
newX <- matrix(c(0.90, 0.05, 0.03, 0.02,
                 0, 0.85, 0.09, 0.06,
                 1, 0.00, 0.00, 0.00,
                 1, 0.00, 0.00, 0.00), nrow=4, byrow=TRUE)
kable(newX)
```

0.9	0.05	0.03	0.02
0.0	0.85	0.09	0.06
1.0	0.00	0.00	0.00
1.0	0.00	0.00	0.00

Compute the new steady state and long-run average profit:

```
newSteadyState <- markovChainStep(newX, newX, 100)

newLrAvgProfit <- newPL %*% newSteadyState[1, ]
newLrAvgProfit
```

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

At an average profit of $ 769.3, an ≈ 111.93 increase, the new policy is worthwhile.

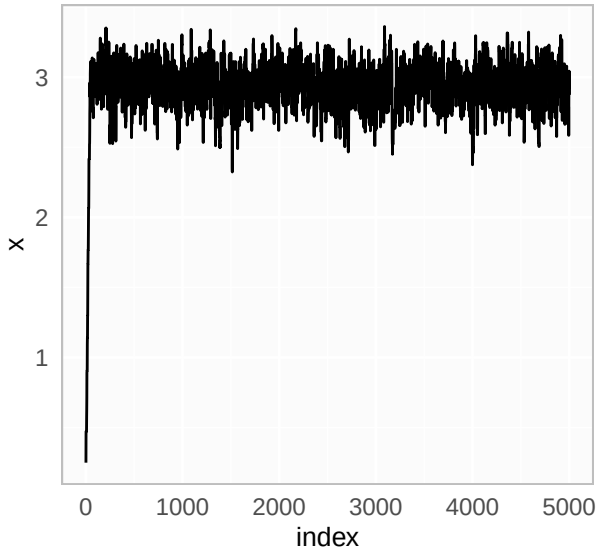
2

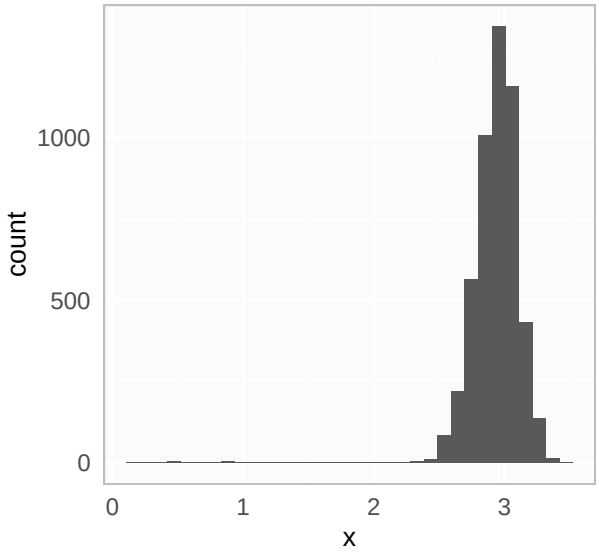
Rao (1973) presented an example on genetic linkage of 197 animals in four categories. The group sizes are (125, 18, 20, 34). Assume that the probabilities of the corresponding multinomial distribution are . . .

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

w <- 0.25
m <- 5000
burn <- 1000
#days <-  250
x <- numeric(m)

# Function to wrap the computation of the target density.
prob <- function(theta, K) {
  p <- (0.5 + (theta / 4))^K[1] * (1 - theta / 4)^K[2] * (1 - theta / 4)^K[3] * (theta/4)^K[4]
}
# Random Walk - Metropolis algorithm
u <- runif(m)
v <- runif(m, -w, w)
x[1] <- 0.25
for(i in 2:m) {
  y <- x[i-1] + v[i]
  if(u[i] <= prob(y, groupSizes) / prob(x[i-1], groupSizes)) {
    x[i] <- y
  } else {
    x[i] <- x[i-1]
  }
}
# Use mean to distill the MCMC results to a single value
xb <- x[burn + 1: m]
theta.hat <- mean(xb, na.rm = TRUE)
print(theta.hat)
```

```
## [1] 2.93708
```

The estimate of θ is 2.9370804. The following charts show the random walk Metropolis chain and resulting empirical distribution.





3

Define the provided Y vector:

```
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,
```

Define the `fullcondm` function per the implementation pseudo code.

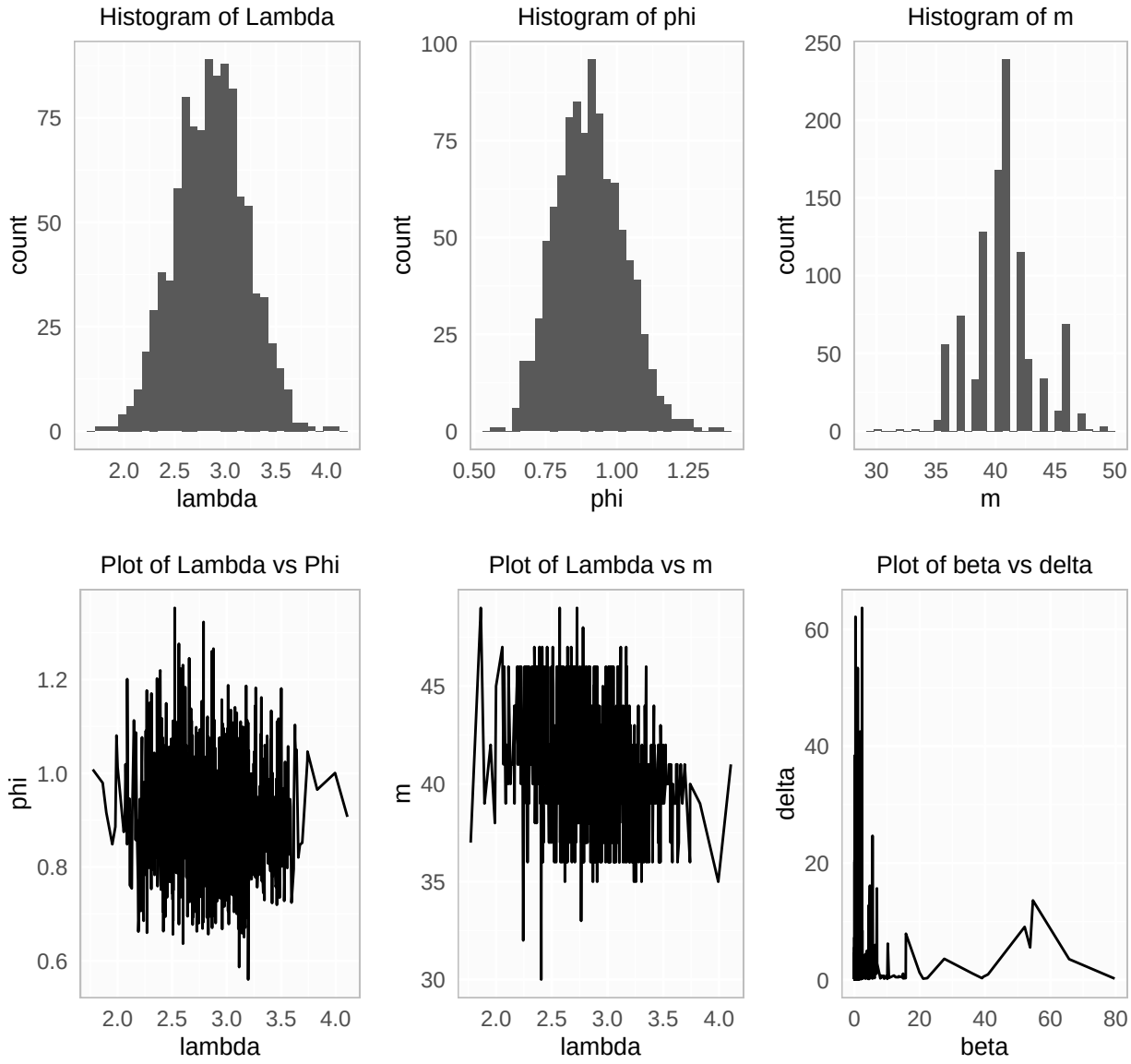
```
fullcondm <- function(m, lambda, phi, y, n, alpha0, beta0, gamma0, delta0, verbose=FALSE)
{

  lamexp <- ifelse(m > 1, sum(y[1:m]), 0)
  phiexp <- ifelse(m < n, sum(y[(m+1):n]), 0)



  t1 <- lambda^(alpha0-1+lamexp)
  t2 <- exp(-(beta0+m)*lambda)
  t3 <- phi^(gamma0-1+phiexp)
  t4 <- exp(-(delta0+n-m)*phi)

  if(verbose == TRUE)
  {
    print(paste(t1, t2, t3, t4, lamexp, phiexp, sep=" "))
  }

  return (t1*t2*t3*t4)
}
```

Define the `R` code for the Gibbs sample based onthe pseudo-code for implementation:

```
gibbsSampler3 <- function(alpha0, gamma0)
{
  n <- length(Y)
```

```
  z <- 1000
  #lambda <- phi <- B <- delta <- numeric(z)
  #L <- numeric(n)

  lambda <- 1
  phi <- 1
  m <-  sample(1:n, 1)
  B <- 1
  delta <- 1

  Mprob <- numeric(n)
  mChain <- matrix(nrow=z, ncol=5)

  # Gibbs Sampler
  for(i in 1:z) {


    sum1 <- sum(Y[1:m])
    sum2 <- sum(Y[(m+1):n])

    lambda <- rgamma(1, shape=sum1+alpha0, scale=B / (B * m + 1))
    phi <- rgamma(1, shape=sum2+gamma0, scale=delta / ( delta * (n-m) + 1))

    for(j in 1:n) {
      Mprob[j] <- fullcondm(j, lambda, phi, Y, n, alpha0, B, gamma0, delta, FALSE)
    }

    m <- sample(1:n, size=1, prob=Mprob)

    bDone <- FALSE
    while(!bDone) {
      B <- 1 / rgamma(1, shape=alpha0, scale=lambda + 1)
      if(B <= 99)
      {
        bDone <- TRUE
      }
    }

    bDone <- FALSE
    while(!bDone) {
      delta <- 1 / rgamma(1, shape=gamma0, scale=phi + 1)
      if(delta <= 99)
      {
        bDone <- TRUE
      }
    }

    mChain[i,] <- c(lambda, phi, m, B, delta)

  }

  return (mChain)
}

gibbsData <- gibbsSampler3(1, 1)
```

a) Run your code for 5000 iterations.



b)

When do you think the change point occured?

```
mMean <- mean(dfMChain$m)
mMean
```

```
## [1] 40.634
```

```
roundedM <- round(mMean)
```

Rounding to a whole year, it appears 41 years would be change point. The 95% confidence interval is computed based on the standard deviation of the vector of m's generated from the MCMC simulation as shown below:

```
n <- length(Y)
sdM <- sd(dfMChain$m)
tVal <- pt(0.975, df=n-1)
```

```
seM <- sdM / sqrt(n)

ci95 <- c(mMean - (tVal * sdM), mMean + (tVal * sdM))
ci95
```

```
## [1] 38.40706 42.86094
```

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

The following code computes the before and after change point means:

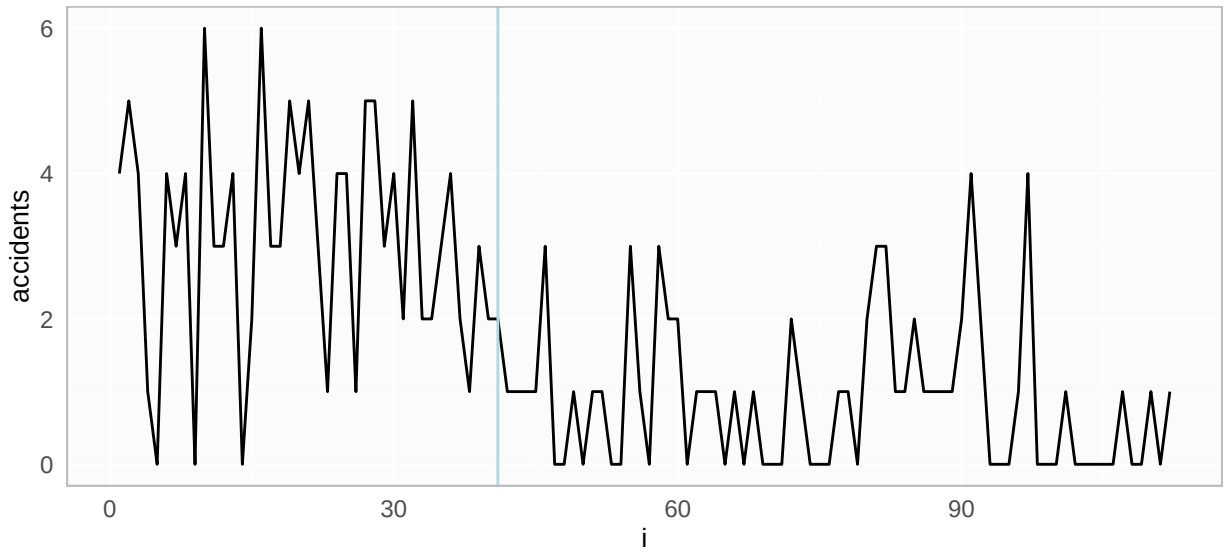
```
beforeMean <- mean(Y[1:roundedM])
beforeMean
```

```
## [1] 3.097561
```

```
afterMean <- mean(Y[(roundedM+1):n])
afterMean
```

```
## [1] 0.9014085
```

As shown in the line chart below, there does appear to be a difference in the data prior to 41 versus after 41 (the blue line = 41).



c) How is Gibbs sampling different from the Metropolis-Hastings approach?

Gibbs sampling is a special case of Metropolis-Hastings, where every sampled point is accepted.

4) Traveling Salesman Problem

First define the cost matrix:

```
C <- as.matrix(read.table("TravelingSalesmenCosts.csv", sep=","))
kable(C)
```

V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15	V16	V17
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

```
costTsp <- function(sol, C)
{
  cost <- 0
  for(i in 2:length(sol))
  {
    a <- sol[i - 1]
    b <- sol[i]
    cost <- cost + as.numeric(C[a,b])
  }

  return (cost)
}
```

```
simulatedAnnealingTsp <- function(T, z, beta, C)
{
  n <- nrow(C)
  results <- numeric(z)
  path <- matrix(nrow=z, ncol=n)

  x <- sample(1:17, 17, replace=FALSE)
  Sx <- costTsp(x, C)
  xbest <- x
  sbest <- Sx


  for( i in 1:z )
  {
    x <- sample(1:17, 17, replace=FALSE)
    ft <-  x[1:2]
    I <- ft[order(ft)]
    y <- x #c(x[1:I[1]-1], x[I[2]:-1:]

    Sy <- costTsp(y, C)
    alpha <- ifelse( Sy < Sx, 1, exp(-(Sy-Sx)/T))
    u <- runif(1, 0, 1)
    if(u < alpha)
    {
      x <- y
      Sx <- Sy
```

```
    }

    T <- beta * T

    xbest <- x
    sbest <- Sx

    results[i] <- sbest
    path[i,] <- x
  }

  return(list(costs=results, paths=path))
}

z <- 10000
T <- 1
beta <- 0.9999
res <- simulatedAnnealingTsp(T, z, beta, C)

# Minimum solution and path:
min(res$costs)
```

```
## [1] 2722
```

```
ndxMin <- which.min(res$costs)
res$paths[ndxMin,]
```

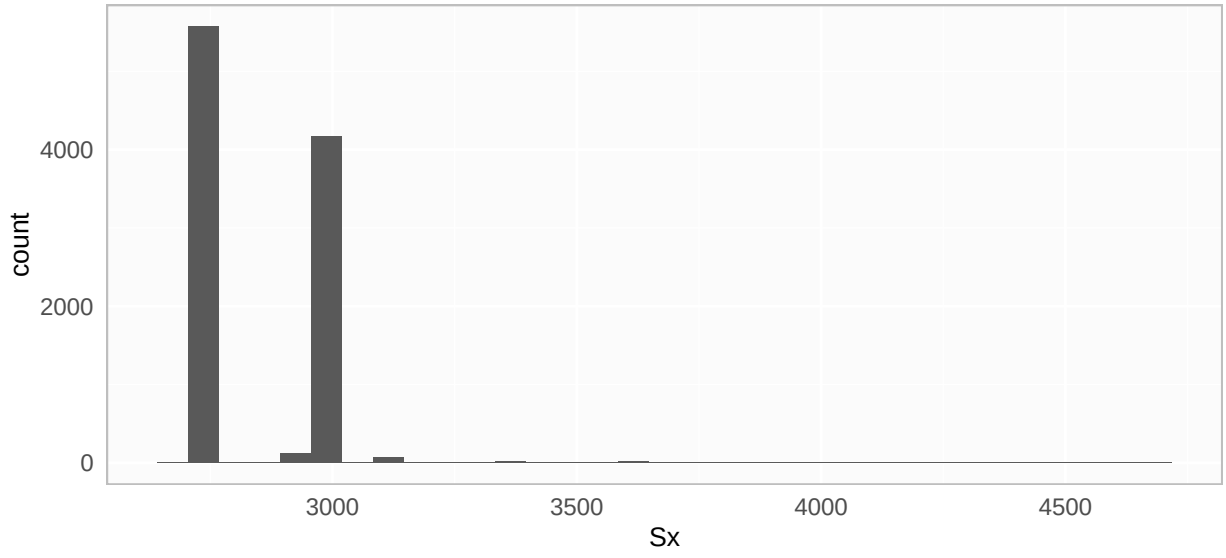
```
##  [1]  7  1  4 13  9 16 12  5 17 14 11  6  8  3 15 10  2
```

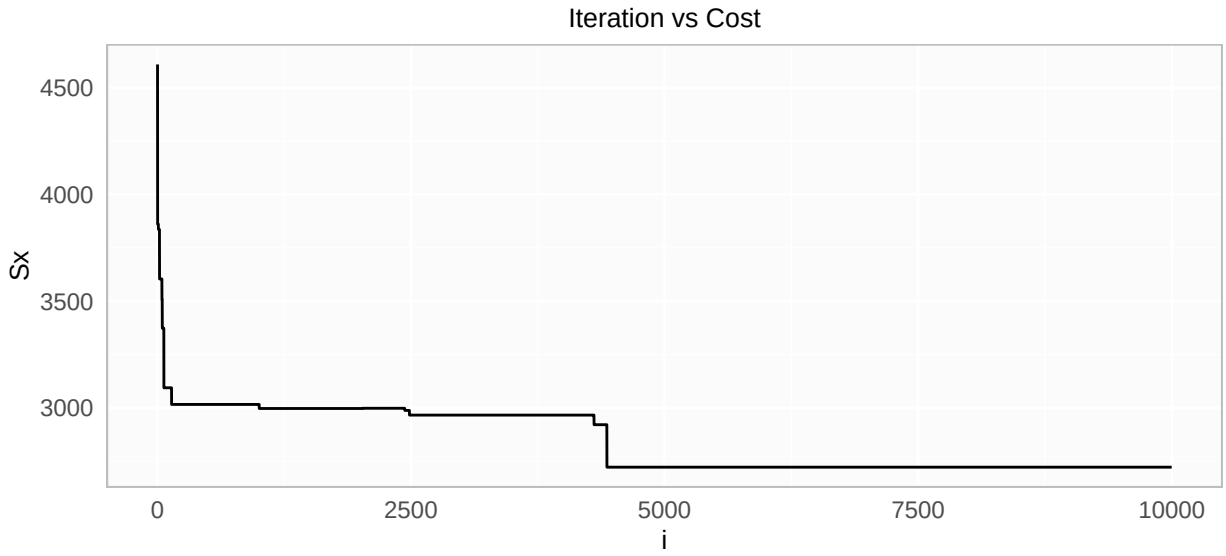
A random tour, for comparison:

```
n <- nrow(C)
r <- sample(1:n, n, replace=FALSE)
costTsp(r, C)
```

```
## [1] 4015
```

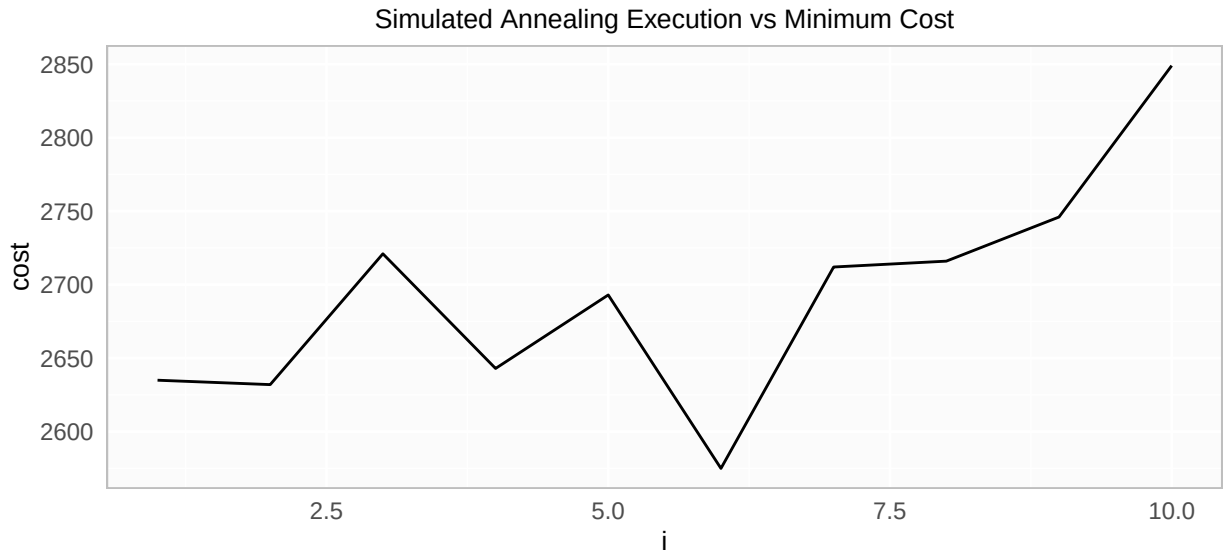
The following histogram shows the distribution of the costs through the simulated annealing process:





Running the simulated annealing process again, this time multiple times inorder to examine the behaviour of the minimum across multiple runs.

```
z2 <- 10
costList <- numeric(z2)
for(i in 1:z2)
{
  res <- simulatedAnnealingTsp(T, z, beta, C)
  costList[i] <- min(res$costs)
}

minCost <- min(costList) #2632
dfStatsCost <- data.frame(min=minCost, mean=mean(costList), max=max(costList), sd=sd(costList))
```



The minimum cost from the 10 runs is 2575. As shown in the line chart above, any given run does not find the minimum, and potentially we still haven't found the global minimum.

min	mean	max	sd
2575	2692.2	2849	76.24638

Experiment with Citations

As an aside, I wanted to experiment with using R Markdown and bibtex citations. The first citation below is created using the `citep` function, while the second uses the bibtex approach `[@Auth_Year]` based on the bib file's key for the reference.

The citation: (Mau, Newton, and Larget, 1999)

Another citation approach: (Mau et al., 1999)

References

Mau, B., Newton, M.A. and Larget, B. (1999), "Bayesian phylogenetic inference via markov chain monte carlo methods", *Biometrics*, Wiley-Blackwell, Vol. 55 No. 1, pp. 1–12.