Lab Assignment 01

Sam Swain

2022-10-30

Question 1

```
df_1 = read.table("webtraffic.txt", header = TRUE)
```

1(A)

```
df_1 = read.table("webtraffic.txt", header = TRUE)
# Create count matrix
df_1 <- colSums(df_1)
Traffic <- matrix(data = 0, nrow = 9, ncol = 9)
Traffic[9, 1] <- 1000
k = 1
for (i in 1:8) {
  for (j in 1:9) {
    Traffic[i, j] <- df_1[k]
    k = k + 1
  }
}

# Create probability matrix
Traffic_probability <- t(apply(Traffic ,1 , function(x) x/sum(x)))
Traffic
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
##  [1,]    0  447  553    0    0    0    0    0    0
##  [2,]    0   23  230  321    0    0    0    0   63
##  [3,]    0  167   43  520    0    0    0    0   96
##  [4,]    0    0    0   44  158  312  247    0  124
##  [5,]    0    0    0    0   22   52   90  127  218
##  [6,]    0    0    0    0   67   21    0  294   97
##  [7,]    0    0    0    0    0   94    7  185   58
##  [8,]    0    0    0    0  262    0    0   30  344
##  [9,] 1000    0    0    0    0    0    0    0    0
```

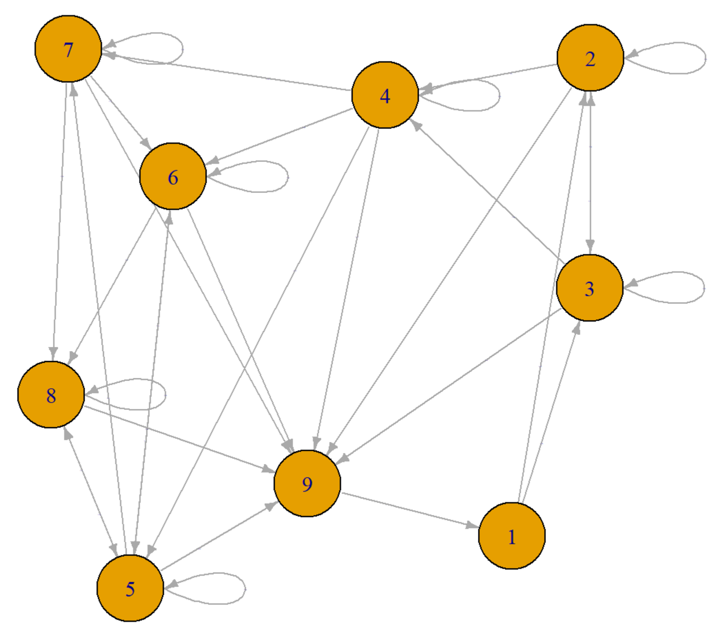
1(B)

```
set.seed(6)
library(markovchain)
```

```
## Warning: package 'markovchain' was built under R version 4.2.1
```

```
## Package:  markovchain
## Version:  0.9.0
## Date:     2022-07-01
## BugReport: https://github.com/spedygiorgio/markovchain/issues
```

```
states <- c("1", "2", "3", "4", "5", "6", "7", "8", "9")

mc <- new("markovchain", states=states, transitionMatrix=Traffic_probability)

dimnames(Traffic_probability) <- list(states, states)
mc@transitionMatrix <- Traffic_probability

par(mar=c(0.1 ,0.1 , 0.1, 0.1))

plot(mc, edge.arrow.size=0.5,
     vertex.size = 25,
     edge.label.cex = 0.0001)
```



1(C)

```
Traffic_probability
```

```
##   1          2          3          4         5          6          7          8
## 1 0 0.44700000 0.55300000 0.00000000 0.0000000 0.00000000 0.00000000 0.00000000
## 2 0 0.03610675 0.36106750 0.50392465 0.0000000 0.00000000 0.00000000 0.00000000
## 3 0 0.20217918 0.05205811 0.62953995 0.0000000 0.00000000 0.00000000 0.00000000
## 4 0 0.00000000 0.00000000 0.04971751 0.1785311 0.35254237 0.27909605 0.00000000
## 5 0 0.00000000 0.00000000 0.00000000 0.0432220 0.10216110 0.17681729 0.24950884
## 6 0 0.00000000 0.00000000 0.00000000 0.1398747 0.04384134 0.00000000 0.61377871
## 7 0 0.00000000 0.00000000 0.00000000 0.0000000 0.27325581 0.02034884 0.53779070
## 8 0 0.00000000 0.00000000 0.00000000 0.4119497 0.00000000 0.00000000 0.04716981
## 9 1 0.00000000 0.00000000 0.00000000 0.0000000 0.00000000 0.00000000 0.00000000
##           9
## 1 0.0000000
## 2 0.0989011
## 3 0.1162228
## 4 0.1401130
## 5 0.4282908
## 6 0.2025052
## 7 0.1686047
## 8 0.5408805
## 9 0.0000000
```

1(D)

The probability of a visitor being on Page 5 after 5 clicks is 0.043222

1(E)

```
Traffic_probability[9, 1] = 1; Traffic_probability[9, 9] = 0
Q=t(Traffic_probability) - diag(9)
Q[9,] = rep(1, 9)
rhs = c(rep(0, 8), 1)
Pi = solve(Q,rhs)
print(Pi)
```

```
##          1          2          3          4          5          6          7
## 0.15832806 0.10085497 0.13077897 0.14012033 0.08058898 0.07583914 0.05446485
##          8          9
## 0.10069664 0.15832806
```

1(F)

```
B=Traffic_probability[1:8,1:8]
Q=diag(8)-B
rhs=c(0.1, 2, 3, 5, 5, 3, 3, 2)
m=solve(Q,rhs)
```

The average time a visitor spends on the website until he/she first leaves is 14.563 minutes.

Question 2

2(A)

$$n \geq \frac{var[p(x)]}{(tolerance)^2\delta} = \frac{\frac{1}{\lambda^2}}{(10^{-3})^2 * 0.01} \therefore n \geq \frac{10^8}{\lambda^2}$$

2(B)

```
lambdas = c(1, 2, 4)
results = c()
results_real = c()

for (i in lambdas){

  n = 10^8/i^2
  x = runif(n, 0, 1)
  y = -log(x)/i
  g = sin(y)/i

  results = append(results, sum(g)/n)
  results_real = append(results_real, (1 / (1+i^2)))
}
```

Lambda = 1

Using MCMC, we get a value of 0.5000116. This is different from the real result, 0.5 by 1.1574817^{-5}. Therefore it is within the tolerance.

Lambda = 2

Using MCMC, we get a value of 0.2000093. This is different from the real result, 0.2 by 9.2558051^{-6}. Therefore it is within the tolerance.

Lambda = 4

Using MCMC, we get a value of 0.0588085. This is different from the real result, 0.0588235 by 1.5026272^{-5}. Therefore it is within the tolerance.

Question 3

3(A)

We can't use Metropolis Sampling because a gamma distribution is not symmetrical. Since gamma distributions aren't join distributions, we can't use Gibbs Sampling either. Therefore, we are left with Metropolis-Hastings Algorithm.

3(B)

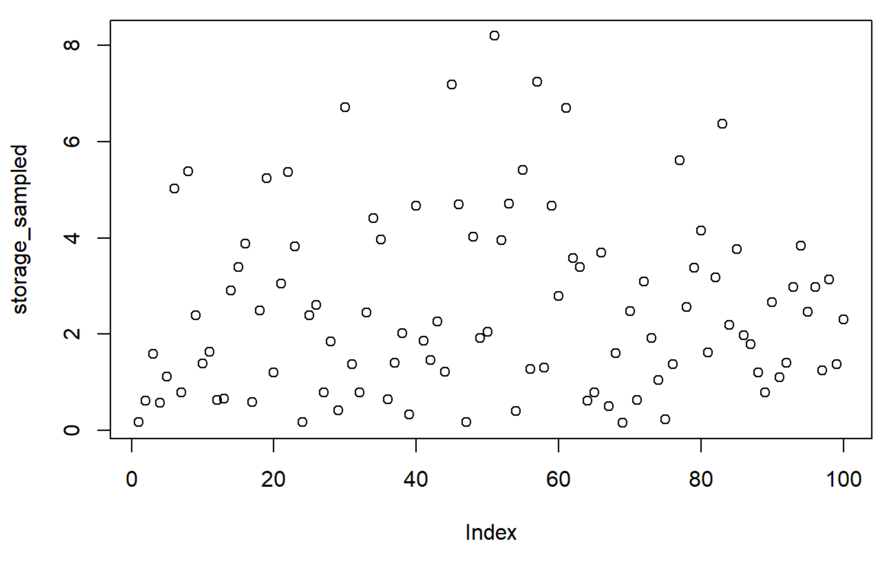
```
x = 1
n = 205000
storage = rep(0, n)

for (i in 1:n) {

  x_prime = rchisq(1, x)

  r_top = dchisq(x, x_prime) * dgamma(x = x_prime, shape = 2, scale = 2)
  r_bottom = dchisq(x, x) * dgamma(x = x, shape = 2, scale = 2)

  a = r_top/r_bottom

  u = runif(1, min = 0, max = 1)

  if (u <= a) {
    x = x_prime
  }

  storage[i] = x

}
```

```
remove_burnin_storage = storage[5001:n]
storage_sampled = remove_burnin_storage[seq(1, n, 2000)]
storage_sampled = storage_sampled[!is.na(storage_sampled)]
```

3(C)

As we can see from the scatter plot below, the data seems to be random. Looking further into a time series plot, we can see there is no apparent correlation between concurrent observations.

```
plot(storage_sampled)
```




```
plot(ts(storage_sampled))
```