MSiA 400 Lab Assignment 1: mjk3551

Matt Ko

Problem 1a

Construct a 9 by 9 matrix `Traffic` that counts total traffic from State i to State j, for $i, j \in \{1, \cdots, 9\}$. Note that `Traffic` has 0's in row 9 and column 1. Set `Traffic[9,1]=1000`. (This is equivalent to making each user return to the home page after they leave the website.) Display `Traffic`. Hint: `colSums()` adds all rows for each column.

```
web = as.matrix(read.delim("webtraffic.txt"))
counts = colSums(web)
Traffic = matrix(,nrow = 9,ncol = 9)
for(row in 1:9){
  for(col in 1:9){
    #print((row-1)+(col))
    #print(counts[(row-1)*9+(col)])
    Traffic[row,col]=counts[(row-1)*9+(col)]
  }
}
Traffic[9,1]=1000
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
```

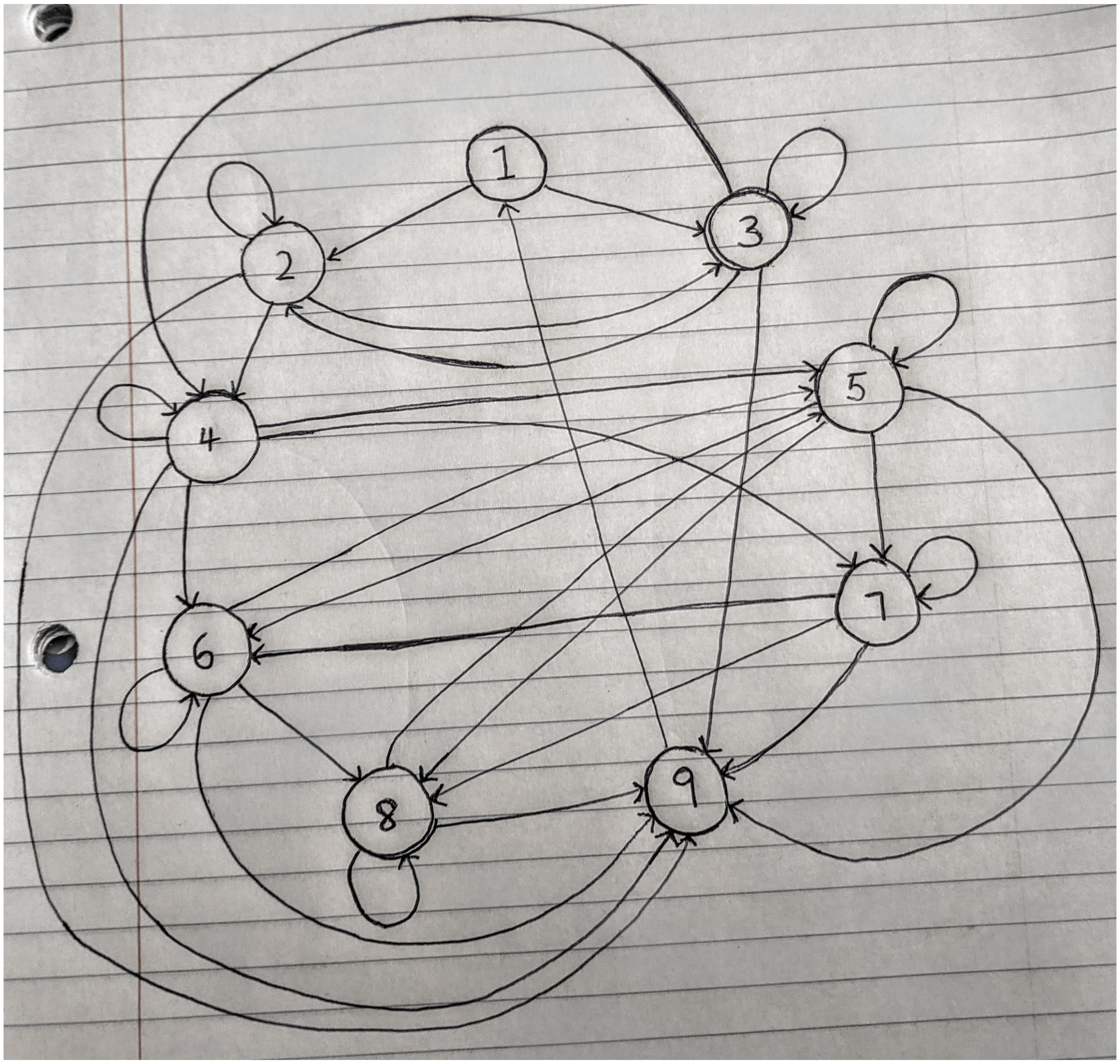


Figure 1: ‘Markov Graph’

Problem 1b

Draw a directed graph where each node represents a state, and each arrow from State i to State j has positive (non-zero) traffic (i.e., `Traffic[i,j]>0`). This may be submitted as a TikZ graph (or using your graphing program of choice) or a picture of a hand-drawn graph (provided it is legible). Is the Markov chain irreducible? Is the Markov chain ergodic? Explain.

This graph is both irreducible and ergodic. It is irreducible because all states communicate with each other. In terms of this problem, you can get to any page after entering the website through the website without entering a url separately. The graph it ergodic because all states are aperiodic, with most having traffic from itself to itself, and the greatest common factor of the number of steps needed to get back to state 1 and 9 is 1. All states are also recurrent because they will eventually return to the same state if they leave.

Problem 1c

Construct and display the one-step transition probability matrix `P` (using the Maximum Likelihood estimate, i.e., $p_{ij} = \frac{\textbf{Traffic}[i,j]}{\sum\limits_{j=1}^{9} \textbf{Traffic}[i,j]}$).

```
rowcount = rowSums(Traffic)
ProbMatrix = matrix(,nrow = 9,ncol = 9)
for(row in 1:9){
  for(col in 1:9){
    ProbMatrix[row,col] = Traffic[row,col]/rowcount[row]
  }
}
ProbMatrix
```

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

Problem 1d

What is the probability of a visitor being on Page 5 after 5 clicks?

```
a = c(1,rep(0,8))
a
```

```
## [1] 1 0 0 0 0 0 0 0 0
```

```
prob5 = a %*% ProbMatrix %*% ProbMatrix %*% ProbMatrix %*% ProbMatrix %*% ProbMatrix
prob5[5]
```

```
## [1] 0.1315178
```

~13.15% probability that the visitor is on page 5 after 5 clicks.

Problem 1e

Compute and display the steady-state probability vector `Pi`, solving the system of equations (as demonstrated in lab).

```
ProbMatrix[9,1]=1; ProbMatrix[9,9]=0 #make the network irreducible
Q=t(ProbMatrix)-diag(9)
Q[9,]=rep(1,9)
rhs=c(rep(0,8),1)
Pi=solve(Q,rhs)
Pi
```

```
## [1] 0.15832806 0.10085497 0.13077897 0.14012033 0.08058898 0.07583914 0.05446485
## [8] 0.10069664 0.15832806
```

Problem 1f

What is the average time a visitor spends on the website (until he/she first leaves)? <u>Hint:</u> Modify the mean first passage time equations, with time spent at each state.

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

```
## [1] 0.1 2.0 3.0 5.0 5.0 3.0 3.0 2.0
```

```
m=solve(Q,rhs)
m[1]
```

```
## [1] 14.563
```

14.563 minutes.

Problem 2a

Determine the number of samples required to achieve an error tolerance of 10^{-3} with 99% confidence.

```r
getsamp <- function(lambda,delta,tolerance){
  n = (1/(lambda^2))/((tolerance)^2*(1-delta))
  return(n)
}
```

$$(1/\lambda^2)/(10^{-3})^2 * (1 - 0.99) = 10^6/(0.01 * \lambda^2)$$

The expression above is the expression regarding the number of samples needed with respect to lambda.

Problem 2b

Compute the approximation (using the number of samples obtained in Problem 2a) and verify that it is within tolerance by comparing to the exact solution: $\int\limits_0^\infty e^{-\lambda x} \sin x dx = \frac{1}{1+\lambda^2}$. Numerically evaluate for each of $\lambda = 1, 2, 4$.

```r
set.seed(100)
approx <- function(lambda,n){

  X = runif(n,0,1)
  Y = -log(X)/lambda
  z = sin(Y)/lambda
  ans = sum(z)/n
  return(ans)
}

#for lambda = 1
n1 = getsamp(1,0.99,10^-3)
l1 = approx(1,n1)
l1
```

```
## [1] 0.5000047
```

```r
#for lambda = 2

n2 = getsamp(2,0.99,10^-3)
l2 = approx(2,n2)
l2
```

```
## [1] 0.1999701
```

```r
#for lambda = 4

n4 = getsamp(4,0.99,10^-3)
l4 = approx(4,n4)
l4
```

```
## [1] 0.05882791
```

```r
#exact solution comparison

exact <- function(lambda){

  ans = 1/(1+lambda^2)
  return(ans)
}

#for lambda=1
print(exact(1))
```

```
## [1] 0.5
```

```r
#for lambda=2
print(exact(2))
```

```
## [1] 0.2
```

```r
#for lambda=4
print(exact(4))
```

```
## [1] 0.05882353
```

Problem 3a

Which MCMC algorithm (Metropolis, Metropolis-Hastings, or Gibbs) is better suited for this problem?

Metropolis-Hastings is the best, because the distribution is not large and multivariate and the exponential distribution is not symmetric.

Problem 3b

Using a burn-in period of 5000 samples and keeping every 100 samples, generate 100 samples from the gamma distribution with shape $k = 2$ and scale $\theta = 2$. Use the algorithm you chose in Problem 3a and write your own sampler (as opposed to using a function from a package).

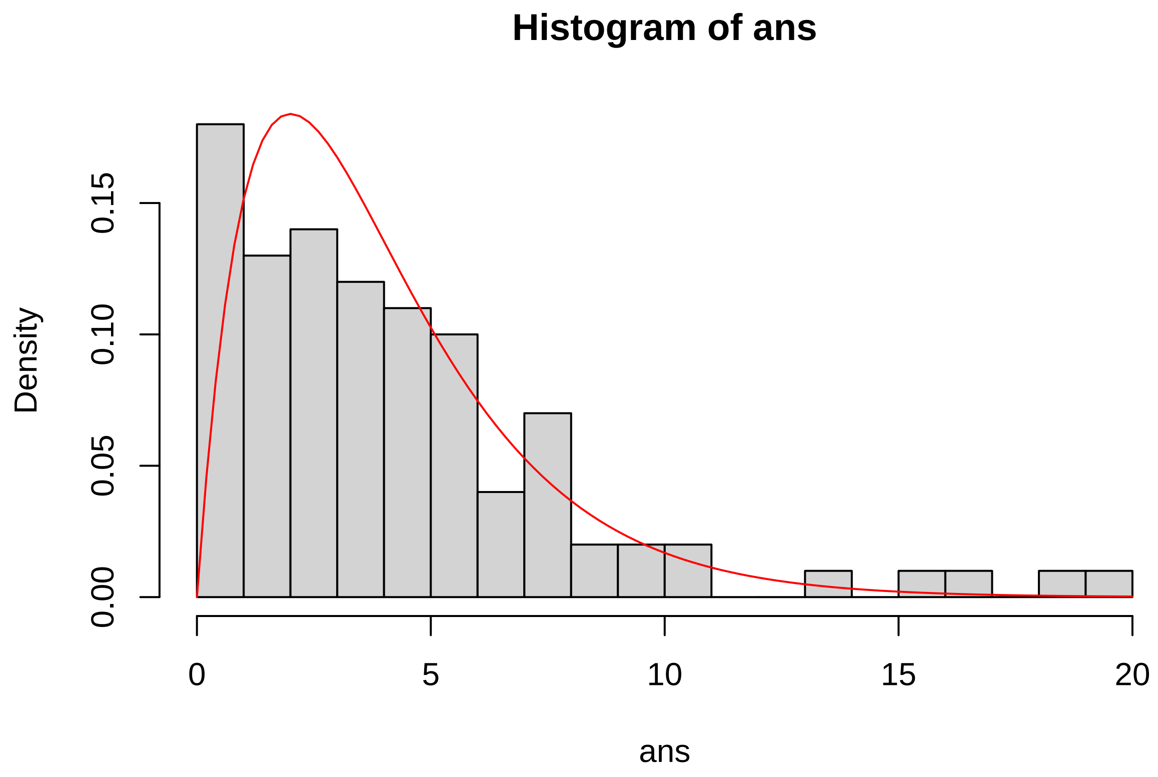
```r
x0 = 1

q_func = function(x){
  ans = rexp(1,rate = x)
  return(ans)

}

p_func = function(x,k,theta){
  ans = x^(k-1)*exp(-1*x/theta)
}

algo_func = function(t,k,thet){
  x = rep(NA,t)
  x[1]=1
  for(i in 1:t){
    xp = q_func(x[i])
    alpha = (p_func(xp,k,thet)*q_func(x[i]))/(p_func(x[i],k,thet)*q_func(xp))
    u = runif(1,0,1)
    if(u<=alpha){
      x[i+1] = xp
    }
    else{
      x[i+1] = x[i]
    }
  }
  return(x)
}

x =algo_func(15000,2,2)
test <- x[5001:15000]
ans <- test[seq(1, length(test), 100)]
ans
```

```
##   [1]  1.44980571  2.94559129  2.40139091  8.99709228  4.05532896  0.94446256
##   [7]  1.27176548  1.49109135  0.53020137  7.54805820  4.59093430  5.47755489
##  [13]  2.24837774  4.04799974  3.21674624  6.54077429  0.73705204  3.11751525
##  [19]  3.23760880  5.84839451  5.76841216  1.55950932  5.87469279  2.19984976
##  [25]  7.68163700  1.16159188  1.99415290  6.67768092  3.93755187  4.67153386
##  [31]  4.07237418  7.13902190  0.67496693  5.25851637  0.11483558  2.30851624
##  [37]  0.55951255  2.25803044  9.99448512 16.05888209  0.83877052 10.73012412
##  [43]  2.48981964  2.96144424  3.23459749  3.89393146  3.37541001 10.18570486
##  [49]  7.04082045  0.26690046  0.46150472  1.08582041  4.38834252  0.76054536
##  [55]  0.94488768  2.16650001  0.02689928  4.57458618  4.06629717  0.61169723
##  [61]  7.51060980  8.82232403  0.66300251  9.23963286  5.26151471  1.79831491
##  [67]  5.69296813  3.43638654  5.69839377  3.76638072 15.48850333  2.04461969
##  [73] 13.70415451  1.66108203  0.11544046  1.38915736  2.62956222  4.25509355
##  [79]  5.24903657  7.04826910  0.74751919  6.80808906  1.43549008  3.76979253
##  [85]  4.12064531  2.01597127  2.41339823 18.32921556  1.92963703  7.04435818
##  [91]  5.56814121  0.63006438  3.93936863  6.72873319  0.13552145  3.57316579
##  [97]  2.47045586 19.30249776  4.43229366  1.62676438
```

Problem 3c

Are the samples generated in Problem 3b sufficiently random? How can you tell?

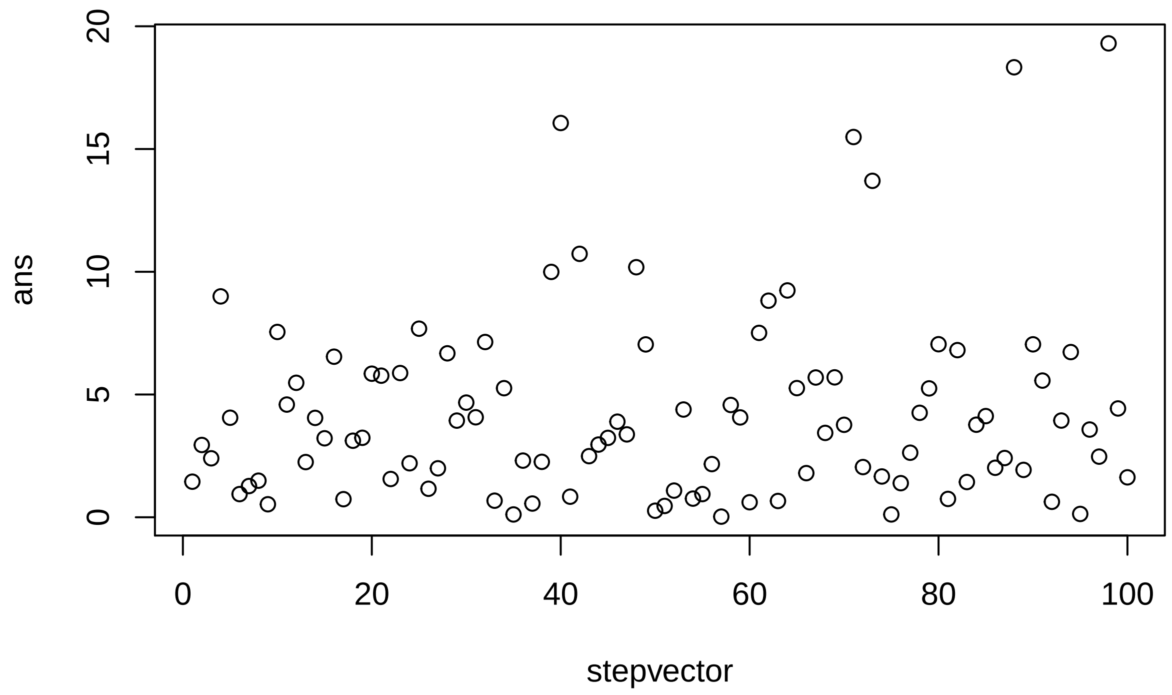
```r
hist(ans,15,prob=TRUE)
curve(dgamma(x, shape = 2, scale = 2), add=TRUE,col="red")
```



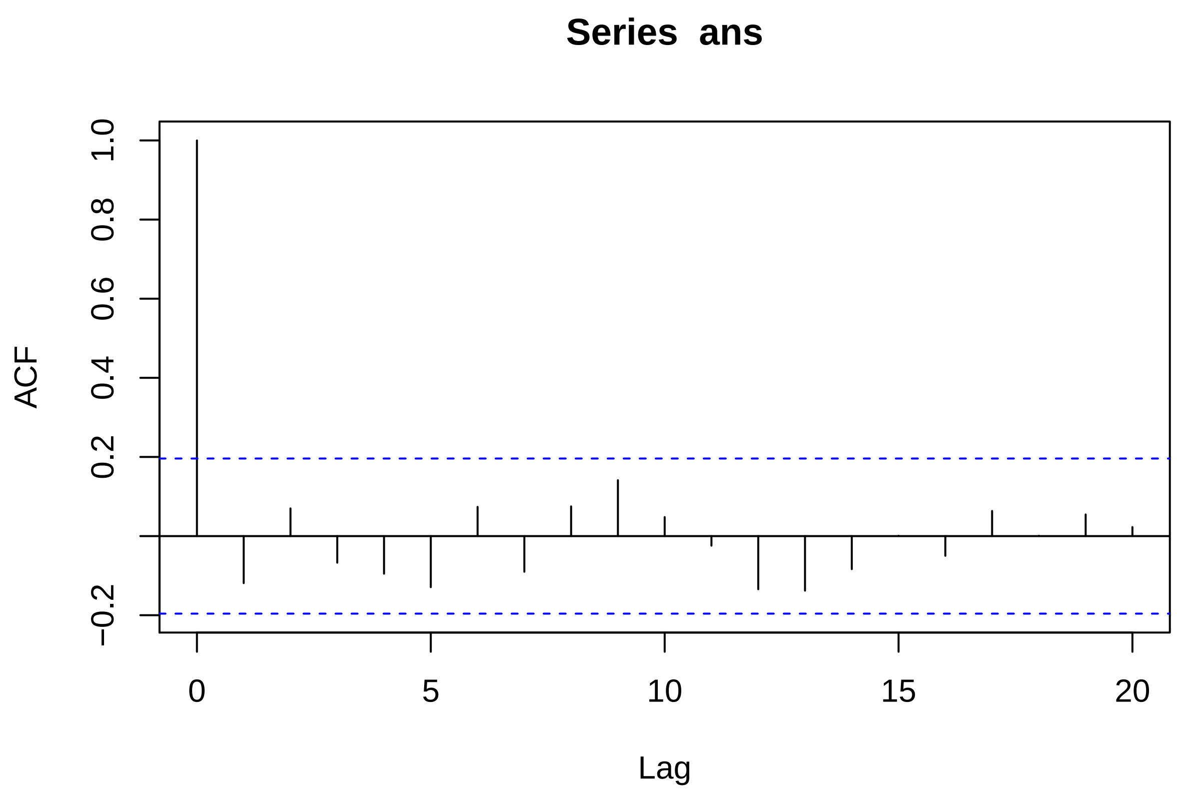
```r
stepvector = seq(1,100)
cor(ans,stepvector)
```

```
## [1] 0.1262007
```

```r
plot(stepvector,ans)
```



```r
acf(ans)
```



Overall, based on the histogram and the plot of steps versus values, the values visually seem sufficiently random. The correlation between the steps and values is also very low, it can be considered negligible. Additionally, the ACF function does not suggest that there exists autocorrelation within our samples.