MSIA 400 - Assignment 3

Steven Lin

Setup

```
# Setup ####

# My PC
main = "C:/Users/Steven/Documents/Academics/3_Graduate
School/2014-2015 ~ NU/"

# Aginity main = '\\\\nas1/labuser169'

course = "MSIA_400_Analytics for Competitive Advantage"
datafolder = "Lab/Assignment_03"
setwd(file.path(main, course, datafolder))

opts_knit$set(root.dir = getwd())  # set the working
directory for all chunks
```

Problem 1

```
# Import data
filename = "markov100.txt"
mydata = read.table(filename, header = F, col.names =
paste("S", 1:100, sep = ""),
    row.names = paste("S", 1:100, sep = ""))
P = as.matrix(mydata)

# Look at data
names(mydata)
head(mydata)[, 1:6]
nrow(mydata)
# summary(mydata)

n = dim(P)[1]
```

Part a

```
# install.packages('expm')
library(expm)
```

```
## Warning: package 'expm' was built under R version 2.15.3
```

```
## Loading required package: Matrix
## Loading required package: lattice
##
## Attaching package: 'expm'
##
## The following object(s) are masked from 'package:Matrix':
##
##     expm
```

```
end = 5
a = rep(0, n)
a[1] = 1
ntrans = 10
dist = a %*% (P %^% ntrans)
dist[end]
```

```
## [1] 0.04509
```

Probability of being in State 5 after 10 transitions: **0.0451**

Part b

```
# install.packages('expm')
library(expm)
end = 10
a = rep(0, n)
a[c(1, 2, 3)] = 1/3
ntrans = 10
dist = a %*% (P %^% ntrans)
dist[end]
```

```
## [1] 0.08269
```

Suppose we are at one of States 1,2, and 3 with equal probabilities. The probability of being in State 10 after 10 transitions : **0.0827**

Part c

```
# install.packages('expm')
library(expm)

end = 1
Q = t(P) - diag(n)
Q[n, ] = rep(1, n)
rhs = rep(0, n)
rhs[n] = 1
Pi = solve(Q) %*% rhs
Pi[end]
```

```
## [1] 0.01257
```

Steady state probability of being in State 1: **0.0126**

Part d

```
# install.packages('expm')
library(expm)


# Mean first passage time m ij is the expected number of
transitions
# before we first reach state j, given we are currently in i

# Mean first passage time to state Start B: submatrix of P
obtained by
# deleting the row and column corresponding to state Sart m:
vector of
# mij, i diff Start, j = start e: vector of 1's m = e + Bm m
= (I-B)^-1e

start = 1
end = 100

B = P[-end, -end]
Q = diag(n - 1) - B
e = rep(1, n - 1)
m = solve(Q) %*% e
m[paste("S", start, sep = ""), ]
```

```
##    S1
## 254.9
```

Mean first passage time from State 1 to State 100: **254.9395**

Problem 2

```
# Import data
filename = "webtraffic.txt"
mydata = read.table(filename, header = T)
P = as.matrix(mydata)

# Look at data
names(mydata)
head(mydata)[, 1:6]
nrow(mydata)
# summary(mydata)
```

Part a

```
# sum columns (traffic from tij column) and store in traffic
matrix counts
# total traffic between state i to state j
Traffic = matrix(as.matrix(colSums(P)), ncol = 9, byrow = T)
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,]    0    0    0    0    0    0    0    0    0
```

Part b

```
Traffic[9, 1] = 1000  #why 9,1 and not 9,9?

sumRows = rowSums(Traffic)
P = Traffic

for (i in 1:9) {
    P[i, ] = P[i, ]/sumRows[i]
}
colnames(P) = paste("p", 1:9, sep = "")
rownames(P) = paste("p", 1:9, sep = "")

# the one step transition probability matrix
P
```

```
##    p1      p2      p3      p4      p5      p6      p7
p8     p9
## p1  0 0.44700 0.55300 0.00000 0.00000 0.00000 0.00000
0.00000 0.0000
## p2  0 0.03611 0.36107 0.50392 0.00000 0.00000 0.00000
0.00000 0.0989
## p3  0 0.20218 0.05206 0.62954 0.00000 0.00000 0.00000
0.00000 0.1162
## p4  0 0.00000 0.00000 0.04972 0.17853 0.35254 0.27910
0.00000 0.1401
## p5  0 0.00000 0.00000 0.00000 0.04322 0.10216 0.17682
0.24951 0.4283
## p6  0 0.00000 0.00000 0.00000 0.13987 0.04384 0.00000
0.61378 0.2025
## p7  0 0.00000 0.00000 0.00000 0.00000 0.27326 0.02035
0.53779 0.1686
## p8  0 0.00000 0.00000 0.00000 0.41195 0.00000 0.00000
0.04717 0.5409
## p9  1 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000
0.00000 0.0000
```

Part c

```
library(expm)

n = dim(P)[1]
Q = t(P) - diag(n)
Q[n, ] = rep(1, n)
rhs = rep(0, n)
rhs[n] = 1
Pi = solve(Q) %*% rhs

# steady state probability vector
Pi
```

```
##       [,1]
## p1 0.15833
## p2 0.10085
## p3 0.13078
## p4 0.14012
## p5 0.08059
## p6 0.07584
## p7 0.05446
## p8 0.10070
## p9 0.15833
```

Part d

```
avgTimePage = c(0.1, 2, 3, 5, 5, 3, 3, 2, 0)  # add zero time
p9
names(avgTimePage) = paste("p", 1:9, sep = "")

# this is like an upperbound, know that the average time
spent in system
# has to be lower than this
sum(avgTimePage)
```

```
## [1] 23.1
```

```
# p i is the fraction of time the process is in state i (in
the long-run)
# this means 0.15 of the time, the user is in page 1 the lon-
run average
# time per period is time p1 * long-run time prob p1 (steady
state
# probability state 1)

avgTimeSite2 = sum(avgTimePage * Pi)
avgTimeSite2
```

```
## [1] 2.306
```

```
# Assume start state x, expected number of step until the
first return is
# 1/p(x)
1/Pi[1, 1]
```

```
##    p1
## 6.316
```

```
# Mean first passage time m ij is the expected number of
transitions
# before we first reach state j, given we are currently in i

# Mean first passage time to state 9 B: submatrix of P
obtained by
# deleting the row and column corresponding to state 9 m:
vector of mij, i
# diff 9, j = 9 e: vector of 1's m = e + Bm m = (I-B)^-1e

start = 1
end = 9

B = P[-end, -end]
Q = diag(n - 1) - B
e = rep(1, n - 1)
m = solve(Q) %*% e
m
```

```
##     [,1]
## p1 5.316
## p2 4.402
## p3 4.247
## p4 3.392
## p5 2.430
## p6 2.749
## p7 2.940
## p8 2.100
```

```
# ave time p1 * # times in p1 + ave time p2 * # times in p2 /
(# times in
# p1 + p2)

# repeat mean first passage time but look at time instead of
transitions
# mij = pij * tj + sum(k diff j) pik(tk+mkj) mij = e + Bm,
where e= pij *
# tj + sum(k diff j) pik(tk) tj = 0 for state 9 then add ti
(othwerwise
# will double count)

start = 1
end = 9

B = P[-end, -end]
Q = diag(n - 1) - B

t = avgTimePage[-end]  # there is no time for state 9
e = P[-end, end] * avgTimePage[end] + B %*% t
m = solve(Q) %*% e
m
```

```
##      [,1]
## p1 14.463
## p2 12.224
## p3 11.657
## p4  7.736
## p5  4.355
## p6  5.450
## p7  5.792
## p8  4.144
```

```
avgTimeSite = m[paste("p", start, sep = ""), ] + t[start]
avgTimeSite
```

```
##    p1
## 14.56
```

```
# Average time on stie = Total time on site/visits total time
on site =
# page a + page bb average time on page = total time on page
# a/(pageviews-Exits)

# Other option is to sum all visits to page i, and do sum(avg
tim page i *
# # visits page i)/sum(vists)

sum(apply(Traffic, 2, sum) * avgTimePage)/sum(Traffic[, 1])
```

```
## [1] 14.56
```

The average time a visitor spend on the website: 14.563. This is using the approach of mean passage time. Alternatively, multiplying the average times by the number of visits (total time spent in system) divided by the number of visitors gives the same answer. Using your approach (multiply steady state by average times) the answer is: 2.3057. Note that this number seems to low, since we know most customers visit more than one page and the lowest average time per page is 2 min (without counting page 1).

Part e

```
# from Page2, 30% of the current outgoing traffic to State 3
would move to
# State 6
Traffic[2, 6] = Traffic[2, 3] * 0.3
Traffic[2, 3] = Traffic[2, 3] - Traffic[2, 6]

# 20% of the current outgoing traffic to State 4 would move
to State 7
Traffic[2, 7] = Traffic[2, 4] * 0.2
Traffic[2, 4] = Traffic[2, 4] - Traffic[2, 7]

Traffic
```

```
##       [,1] [,2] [,3]  [,4] [,5] [,6]  [,7] [,8] [,9]
##  [1,]    0  447  553   0.0    0    0   0.0    0    0
##  [2,]    0   23  161 256.8    0   69  64.2    0   63
##  [3,]    0  167   43 520.0    0    0   0.0    0   96
##  [4,]    0    0    0  44.0  158  312 247.0    0  124
##  [5,]    0    0    0   0.0   22   52  90.0  127  218
##  [6,]    0    0    0   0.0   67   21   0.0  294   97
##  [7,]    0    0    0   0.0    0   94   7.0  185   58
##  [8,]    0    0    0   0.0  262    0   0.0   30  344
##  [9,] 1000    0    0   0.0    0    0   0.0    0    0
```

```
sumRows = rowSums(Traffic)
P = Traffic

for (i in 1:9) {
    P[i, ] = P[i, ]/sumRows[i]
}
colnames(P) = paste("p", 1:9, sep = "")
rownames(P) = paste("p", 1:9, sep = "")

# the one step transition probability matrix
P
```

```
##    p1      p2      p3      p4      p5      p6      p7
p8     p9
## p1  0 0.44700 0.55300 0.00000 0.00000 0.00000 0.00000
0.00000 0.0000
## p2  0 0.03611 0.25275 0.40314 0.00000 0.10832 0.10078
0.00000 0.0989
## p3  0 0.20218 0.05206 0.62954 0.00000 0.00000 0.00000
0.00000 0.1162
## p4  0 0.00000 0.00000 0.04972 0.17853 0.35254 0.27910
0.00000 0.1401
## p5  0 0.00000 0.00000 0.00000 0.04322 0.10216 0.17682
0.24951 0.4283
## p6  0 0.00000 0.00000 0.00000 0.13987 0.04384 0.00000
0.61378 0.2025
## p7  0 0.00000 0.00000 0.00000 0.00000 0.27326 0.02035
0.53779 0.1686
## p8  0 0.00000 0.00000 0.00000 0.41195 0.00000 0.00000
0.04717 0.5409
## p9  1 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000
0.00000 0.0000
```

```
library(expm)

n = dim(P)[1]
Q = t(P) - diag(n)
Q[n, ] = rep(1, n)
rhs = rep(0, n)
rhs[n] = 1
Pi2 = solve(Q) %*% rhs

# steady state probability vector
Pi2
```

```
##       [,1]
## p1 0.16163
## p2 0.10034
## p3 0.12104
## p4 0.12276
## p5 0.08165
## p6 0.08251
## p7 0.06003
## p8 0.10841
## p9 0.16163
```

```
# Decide if the link helped balancing the traffic by
comparing the
# variance of Pi and Pi2.
Pi - Pi2
```

```
##          [,1]
## p1 -0.0033003
## p2  0.0005116
## p3  0.0097357
## p4  0.0173631
## p5 -0.0010571
## p6 -0.0066697
## p7 -0.0055673
## p8 -0.0077155
## p9 -0.0033003
```

```
var(Pi)
```

```
##          [,1]
## [1,] 0.001411
```

```
var(Pi2)
```

```
##         [,1]
## [1,] 0.00122
```

```
var(Pi2) < var(Pi)
```

```
##      [,1]
## [1,] TRUE
```

```
(var(Pi2) - var(Pi))/var(Pi)
```

```
##          [,1]
## [1,] -0.1354
```

```

```

The variance of Pi2 is less than the variance of Pi, so adding the links helped reduced the variance and thus balanace the traffic. More specifically, the variance was reduced by 13.5446 %.