stochastic exam

kallakuri radhakrishna

15/06/2020

example 1

a

```
#states
states <- c("A","B","C","D")

#transition matrix

tmat <- matrix(data=c(.8,.1,.09,.01,
                      .1,.8,.08,.02,
                      0,.1,.85,.05,
                      0,0,0,1), byrow=TRUE, nrow=4,
               dimnames=list(states,states))

tmat
```

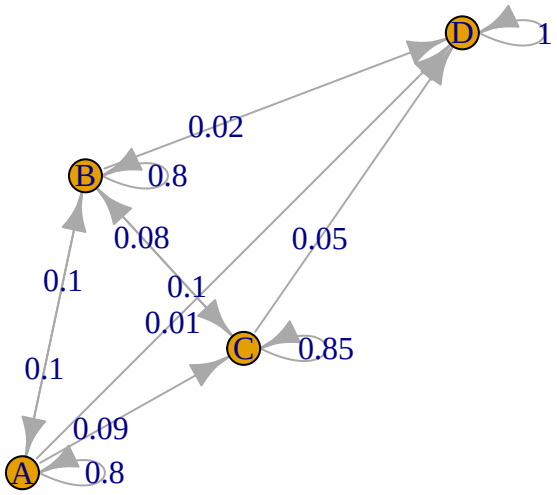
```
##     A   B    C    D
## A 0.8 0.1 0.09 0.01
## B 0.1 0.8 0.08 0.02
## C 0.0 0.1 0.85 0.05
## D 0.0 0.0 0.00 1.00
```

```
library("markovchain")
```

```
## Warning: package 'markovchain' was built under R version 3.6.3
```

```
## Package:  markovchain
## Version:  0.8.4.1
## Date:     2020-05-04
## BugReport: http://github.com/spedygiorgio/markovchain/issues
```

```
# Markov chain
mc <- new("markovchain", states = states, byrow = TRUE, transitionMatrix = tmat, name = "bankruptcy")

# transition diagram
plot(mc)
```



b

Find the probability distribution over its ratings after 3 years.

```
#initial distribution

q0 <- c(0,1,0,0)

dist <-  q0*(mc^3)
print(dist)
```

```
##            A      B       C       D
## [1,] 0.1938 0.5565 0.18689 0.06281
```

c

Is the chain absorbing? Explain. What does absorption mean in this case. For a chain to be absorbing the following conditions have to be met. it contains at least one absorbing state,(A state i is absorbing if pii = 1) every non-absorbing state leads to an absorbing state. Yes the chain is absorbing as it has atleast one absorbing state D in thi case and every other non-absorbing state leads to it. It means that once a company becomes bankrupt it will stay there forever.

```
absorbingStates(mc)
```

```
## [1] "D"
```

d

Which states communicate with the C-rating state? If both i -> j and j -> i then we say that the states communicate. By observation it is state B which communicates with state C and state A communicates with C and C connects to A via B.

So both A and B communicate with C

```
communicatingClasses(mc)
```

```
## [[1]]
## [1] "A" "B" "C"
##
## [[2]]
## [1] "D"
```

e

the probability that a company with rating B will be bankrupted in no more than 5 years is 0.316.

```
q0 <- c(0,1,0,0)

q0*mc
```

```
##        A   B    C    D
## [1,] 0.1 0.8 0.08 0.02
```

```
q0*(mc^2)
```

```
##         A     B     C     D
## [1,] 0.16 0.658 0.141 0.041
```

```
q0*(mc^3)
```

```
##           A      B       C       D
## [1,] 0.1938 0.5565 0.18689 0.06281
```

```
q0*(mc^4)
```

```
##            A        B         C         D
## [1,] 0.21069 0.483269 0.2208185 0.0852225
```

```
q0*(mc^5)
```

```
##              A         B         C         D
## [1,] 0.2168789 0.4297661 0.2453193 0.1080357
```

```
# adding all probabilities for state D
0.02 + 0.041+0.062+0.085+0.108
```

```
## [1] 0.316
```

f

```
cf <-canonicForm(mc)
cf
```

```
## bankruptcy
##  A  4 - dimensional discrete Markov Chain defined by the following states:
```

```
##  D, A, B, C
##  The transition matrix  (by rows)  is defined as follows:
##      D   A   B    C
## D 1.00 0.0 0.0 0.00
## A 0.01 0.8 0.1 0.09
## B 0.02 0.1 0.8 0.08
## C 0.05 0.0 0.1 0.85
```

```
# list all transient states
ts <- transientStates(cf)
# list the absorbing states
as <- absorbingStates(cf)
ts
```

```
## [1] "A" "B" "C"
```

```
as
```

```
## [1] "D"
```

```
# matrices Q and S
Q <- cf[ts, ts]
S <- cf[ts, as]
Q
```

```
##     A   B    C
## A 0.8 0.1 0.09
## B 0.1 0.8 0.08
## C 0.0 0.1 0.85
```

```
S
```

```
##    A    B    C
## 0.01 0.02 0.05
```

```
#calculation of fundamental matrix
fund <- solve(diag(length(ts))-Q)
fund
```

```
##      A  B    C
## A 11.0 12 13.0
## B  7.5 15 12.5
## C  5.0 10 15.0
```

g

What is expected time to bankruptcy for a company with rating B? expected time to bankruptcy for a company with rating B is 35 years.

```
# time to absorption
rowSums(fund)
```

```
##  A  B  C
## 36 35 30
```

h

expected number of years when a company with current rating B will have rating A can be seen from fundamental matrix is 7.5 years.

```
fund
```

```
##      A  B    C
## A 11.0 12 13.0
## B  7.5 15 12.5
## C  5.0 10 15.0
```

example 2

a

transition matrix 1 2 3 4 5 6 [1,] 0.00 0.50 0.5000000 0.0000000 0.0000000 0.0000000 [2,] 0.50 0.00 0.5000000 0.0000000 0.0000000 0.0000000 [3,] 0.25 0.25 0.0000000 0.2500000 0.2500000 0.0000000 [4,] 0.00 0.00 0.3333333 0.0000000 0.3333333 0.3333333 [5,] 0.00 0.00 0.3333333 0.3333333 0.0000000 0.3333333 [6,] 0.00 0.00 0.0000000 0.5000000 0.5000000 0.0000000

```
states1 <- c("1","2","3","4","5","6")
states1
```

```
## [1] "1" "2" "3" "4" "5" "6"
```

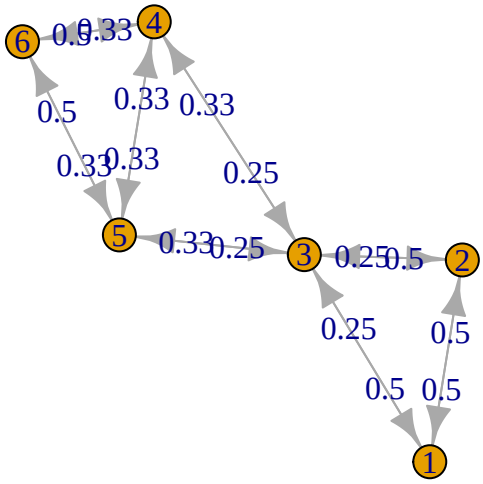
```
adjmat <- matrix(data=c(0,1/2,1/2,0,0,0,
                        1/2,0,1/2,0,0,0,
                        1/4,1/4,0,1/4,1/4,0,
                        0,0,1/3,0,1/3,1/3,
                        0,0,1/3,1/3,0,1/3,
                        0,0,0,1/2,1/2,0
                        ), byrow=TRUE, nrow=6,
                 dimnames=list(states1,states1))

# a
adjmat
```

```
##      1    2         3         4         5         6
## 1 0.00 0.50 0.5000000 0.0000000 0.0000000 0.0000000
## 2 0.50 0.00 0.5000000 0.0000000 0.0000000 0.0000000
## 3 0.25 0.25 0.0000000 0.2500000 0.2500000 0.0000000
## 4 0.00 0.00 0.3333333 0.0000000 0.3333333 0.3333333
## 5 0.00 0.00 0.3333333 0.3333333 0.0000000 0.3333333
## 6 0.00 0.00 0.0000000 0.5000000 0.5000000 0.0000000
```

```
rs <- rowSums(adjmat)
trmat <- diag(1/rs) %*% adjmat
trmat
```

```
##         1    2         3         4         5         6
## [1,] 0.00 0.50 0.5000000 0.0000000 0.0000000 0.0000000
## [2,] 0.50 0.00 0.5000000 0.0000000 0.0000000 0.0000000
## [3,] 0.25 0.25 0.0000000 0.2500000 0.2500000 0.0000000
## [4,] 0.00 0.00 0.3333333 0.0000000 0.3333333 0.3333333
## [5,] 0.00 0.00 0.3333333 0.3333333 0.0000000 0.3333333
## [6,] 0.00 0.00 0.0000000 0.5000000 0.5000000 0.0000000
```

```
rwgraph <- new("markovchain", states = states1, byrow = TRUE,
               transitionMatrix = trmat, name = "Random walk")
plot(rwgraph)
```



b

Calculate P3 where P is the transition matrix

```
P3 <- trmat^3
P3
```

```
##             1        2          3          4          5          6
## [1,] 0.000000 0.125000 0.12500000 0.00000000 0.00000000 0.00000000
## [2,] 0.125000 0.000000 0.12500000 0.00000000 0.00000000 0.00000000
## [3,] 0.015625 0.015625 0.00000000 0.01562500 0.01562500 0.00000000
## [4,] 0.000000 0.000000 0.03703704 0.00000000 0.03703704 0.03703704
## [5,] 0.000000 0.000000 0.03703704 0.03703704 0.00000000 0.03703704
## [6,] 0.000000 0.000000 0.00000000 0.12500000 0.12500000 0.00000000
```

c

yes the chain is regular, as it now has strictly positive elements which are all equal as can be seen in its 1000th power and 500th power. Each row has the limit distribution.

```
rwgraph^500
```

```
## Random walk^500
##  A  6 - dimensional discrete Markov Chain defined by the following states:
##  1, 2, 3, 4, 5, 6
##  The transition matrix  (by rows)  is defined as follows:
```

```
##       1     2    3      4      5     6
## 1 0.125 0.125 0.25 0.1875 0.1875 0.125
## 2 0.125 0.125 0.25 0.1875 0.1875 0.125
## 3 0.125 0.125 0.25 0.1875 0.1875 0.125
## 4 0.125 0.125 0.25 0.1875 0.1875 0.125
## 5 0.125 0.125 0.25 0.1875 0.1875 0.125
## 6 0.125 0.125 0.25 0.1875 0.1875 0.125
```

```
rwgraph^1000
```

```
## Random walk^1000
##  A  6 - dimensional discrete Markov Chain defined by the following states:
##  1, 2, 3, 4, 5, 6
##  The transition matrix  (by rows)  is defined as follows:
##       1     2    3      4      5     6
## 1 0.125 0.125 0.25 0.1875 0.1875 0.125
## 2 0.125 0.125 0.25 0.1875 0.1875 0.125
## 3 0.125 0.125 0.25 0.1875 0.1875 0.125
## 4 0.125 0.125 0.25 0.1875 0.1875 0.125
## 5 0.125 0.125 0.25 0.1875 0.1875 0.125
## 6 0.125 0.125 0.25 0.1875 0.1875 0.125
```

d

as we can see that steadystates function is giving an output which is numerically similar to all the rows of rwgraph1^1000 hence it is again proved that it is a regular markov chain which has a unique convergence.

```
 invariant <- steadyStates(rwgraph)
invariant
```

```
##          1     2    3      4      5     6
## [1,] 0.125 0.125 0.25 0.1875 0.1875 0.125
```

e

mean recurrence time for state 4 is 5.33 steps. we define recurrence time as the time chain will return to its own state, so it takes 5.33 steps for state 4 to reach 4.

```
rt <- 1/invariant
show(rt)
```

```
##      1 2 3        4        5 6
## [1,] 8 8 4 5.333333 5.333333 8
```

```
# alternate method
# mean recurrence times
meanRecurrenceTime(rwgraph)
```

```
##        1        2        3        4        5        6
## 8.000000 8.000000 4.000000 5.333333 5.333333 8.000000
```

f

mean first passage time 1 2 3 4 5 6 1 0.000000 5.333333 2 8 8 13 2 5.333333 0.000000 2 8 8 13 3 8.666667 8.666667 0 6 6 11 4 12.666667 12.666667 4 0 4 7 5 12.666667 12.666667 4 4 0 7 6 13.666667 13.666667 5 3 3 0

```r
# the W matrix - all rows contain pi
Wmat <- matrix(rep(invariant, 6), nrow = 6, byrow = T)
show(Wmat)
```

```
##       [,1]  [,2] [,3]   [,4]   [,5]  [,6]
## [1,] 0.125 0.125 0.25 0.1875 0.1875 0.125
## [2,] 0.125 0.125 0.25 0.1875 0.1875 0.125
## [3,] 0.125 0.125 0.25 0.1875 0.1875 0.125
## [4,] 0.125 0.125 0.25 0.1875 0.1875 0.125
## [5,] 0.125 0.125 0.25 0.1875 0.1875 0.125
## [6,] 0.125 0.125 0.25 0.1875 0.1875 0.125
```

```r
#fundamental matrix
Zmat <- solve(diag(6) - trmat + Wmat)
show(Zmat)
```

```
##         [,1]       [,2]     [,3]       [,4]       [,5]      [,6]
## 1  1.2864583  0.6197917  0.40625 -0.4453125 -0.4453125 -0.421875
## 2  0.6197917  1.2864583  0.40625 -0.4453125 -0.4453125 -0.421875
## 3  0.2031250  0.2031250  0.90625 -0.0703125 -0.0703125 -0.171875
## 4 -0.2968750 -0.2968750 -0.09375  1.0546875  0.3046875  0.328125
## 5 -0.2968750 -0.2968750 -0.09375  0.3046875  1.0546875  0.328125
## 6 -0.4218750 -0.4218750 -0.34375  0.4921875  0.4921875  1.203125
```

```r
#mean first passage matrix
# the diagonal elements
dia <- rowSums(diag(6)*Zmat)
show(dia)
```

```
##        1        2        3        4        5        6
## 1.286458 1.286458 0.906250 1.054688 1.054688 1.203125
```

```r
# the matrix of zjj
Zjj <- matrix(rep(dia, 6), nrow = 6, byrow = T)
show(Zjj)
```

```
##          [,1]     [,2]    [,3]     [,4]     [,5]     [,6]
## [1,] 1.286458 1.286458 0.90625 1.054688 1.054688 1.203125
## [2,] 1.286458 1.286458 0.90625 1.054688 1.054688 1.203125
## [3,] 1.286458 1.286458 0.90625 1.054688 1.054688 1.203125
## [4,] 1.286458 1.286458 0.90625 1.054688 1.054688 1.203125
## [5,] 1.286458 1.286458 0.90625 1.054688 1.054688 1.203125
## [6,] 1.286458 1.286458 0.90625 1.054688 1.054688 1.203125
```

```r
# the mean passage matrix
Mmat <- (Zjj-Zmat)/Wmat
show(Mmat)
```

```
##        [,1]      [,2] [,3] [,4] [,5] [,6]
## 1  0.000000  5.333333    2    8    8   13
## 2  5.333333  0.000000    2    8    8   13
## 3  8.666667  8.666667    0    6    6   11
## 4 12.666667 12.666667    4    0    4    7
## 5 12.666667 12.666667    4    4    0    7
## 6 13.666667 13.666667    5    3    3    0
```

```r
# alternate method
#mean passage times matrix
```

```
meanFirstPassageTime(rwgraph)
```

```
##           1         2 3 4 5  6
## 1  0.000000  5.333333 2 8 8 13
## 2  5.333333  0.000000 2 8 8 13
## 3  8.666667  8.666667 0 6 6 11
## 4 12.666667 12.666667 4 0 4  7
## 5 12.666667 12.666667 4 4 0  7
## 6 13.666667 13.666667 5 3 3  0
```

g

the answer is 13.66

```
trmat
```

```
##         1    2         3         4         5         6
## [1,] 0.00 0.50 0.5000000 0.0000000 0.0000000 0.0000000
## [2,] 0.50 0.00 0.5000000 0.0000000 0.0000000 0.0000000
## [3,] 0.25 0.25 0.0000000 0.2500000 0.2500000 0.0000000
## [4,] 0.00 0.00 0.3333333 0.0000000 0.3333333 0.3333333
## [5,] 0.00 0.00 0.3333333 0.3333333 0.0000000 0.3333333
## [6,] 0.00 0.00 0.0000000 0.5000000 0.5000000 0.0000000
```

```
# make state 1  absorbing
tmabs <- trmat
tmabs[1, ] <- c(1,0,0,0,0,0)

tmabs
```

```
##         1    2         3         4         5         6
## [1,] 1.00 0.00 0.0000000 0.0000000 0.0000000 0.0000000
## [2,] 0.50 0.00 0.5000000 0.0000000 0.0000000 0.0000000
## [3,] 0.25 0.25 0.0000000 0.2500000 0.2500000 0.0000000
## [4,] 0.00 0.00 0.3333333 0.0000000 0.3333333 0.3333333
## [5,] 0.00 0.00 0.3333333 0.3333333 0.0000000 0.3333333
## [6,] 0.00 0.00 0.0000000 0.5000000 0.5000000 0.0000000
```

```
mcabs <- new("markovchain", states = states1, byrow = TRUE, transitionMatrix = tmabs, name = "absorbing

cf1 <-canonicForm(mcabs)
cf1
```

```
## absorbing
##  A  6 - dimensional discrete Markov Chain defined by the following states:
##  1, 2, 3, 4, 5, 6
##  The transition matrix  (by rows)  is defined as follows:
##      1    2         3         4         5         6
## 1 1.00 0.00 0.0000000 0.0000000 0.0000000 0.0000000
## 2 0.50 0.00 0.5000000 0.0000000 0.0000000 0.0000000
## 3 0.25 0.25 0.0000000 0.2500000 0.2500000 0.0000000
## 4 0.00 0.00 0.3333333 0.0000000 0.3333333 0.3333333
## 5 0.00 0.00 0.3333333 0.3333333 0.0000000 0.3333333
## 6 0.00 0.00 0.0000000 0.5000000 0.5000000 0.0000000
```

```r
# list all transient states
ts1 <- transientStates(cf1)
# list the absorbing states
as1 <- absorbingStates(cf1)
ts1
```

```
## [1] "2" "3" "4" "5" "6"
```

```r
as1
```

```
## [1] "1"
```

```r
# matrices Q and S
Q1 <- cf1[ts1, ts1]
S1 <- cf1[ts1, as1]
Q1
```

```
##      2         3         4         5         6
## 2 0.00 0.5000000 0.0000000 0.0000000 0.0000000
## 3 0.25 0.0000000 0.2500000 0.2500000 0.0000000
## 4 0.00 0.3333333 0.0000000 0.3333333 0.3333333
## 5 0.00 0.3333333 0.3333333 0.0000000 0.3333333
## 6 0.00 0.0000000 0.5000000 0.5000000 0.0000000
```

```r
S1
```

```
##    2    3    4    5    6
## 0.50 0.25 0.00 0.00 0.00
```

```r
#calculation of fundamental matrix
fund1 <- solve(diag(length(ts1))-Q1)
fund1
```

```
##           2        3     4     5         6
## 2 1.3333333 1.333333 1.000 1.000 0.6666667
## 3 0.6666667 2.666667 2.000 2.000 1.3333333
## 4 0.6666667 2.666667 3.875 3.125 2.3333333
## 5 0.6666667 2.666667 3.125 3.875 2.3333333
## 6 0.6666667 2.666667 3.500 3.500 3.3333333
```

```r
rowSums(fund1)
```

```
##         2         3         4         5         6
##  5.333333  8.666667 12.666667 12.666667 13.666667
```

example 3

a

arrival rate is 20 and service rate is 12. number of servers s <- 3 incoming rate lam <- 20 service rate mu <- 12

b

we are asking that the first customer will arive after 10 mins The arrival rate lambda = 20 is given in hour unit. Thus, we are asking about probability of (T >= 1/6) the answer is 0.035

```r
p <- 1-pexp(1/6, rate = 20)
p
```

```
## [1] 0.03567399
```

c

The arrival epoch S2 of the next two customers is distributed with Erlang distribution:

```r
pgamma(1/6, shape = 2, rate = 20)
```

```
## [1] 0.8454127
```

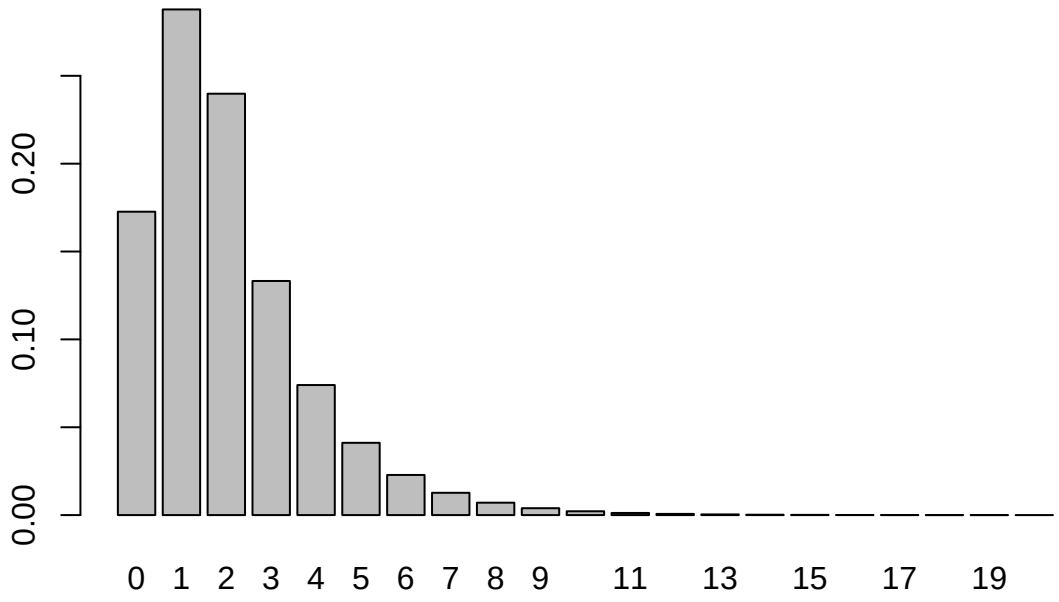
d

limit distribution 1.726619e-01 2.877698e-01 2.398082e-01 1.332268e-01 7.401486e-02 4.111937e-02 2.284409e-02 [8] 1.269116e-02 7.050646e-03 3.917026e-03 2.176125e-03 1.208959e-03 6.716436e-04 3.731353e-04 [15] 2.072974e-04 1.151652e-04 6.398068e-05 3.554482e-05 1.974712e-05 1.097062e-05 6.094791e-06

```r
s <- 3
# incoming rate
lam <- 20
# service rate
mu <- 12
k <- 0:20
# traffic intensity
rho <- lam/mu
# indicator for k <= s
kls <- as.numeric(k <= s)
# indicator for k > s
kgs <- as.numeric(k > s)
# limit distribution
# pi = pp*pi0
pp <- kls*rho^k/factorial(k) + kgs*rho^k/(factorial(s)*s^(k-s))

pi0 <- 1/(sum(pp[1:s]) + rho^s/factorial(s)*s/(s-rho))

show(pi0)
```

```
## [1] 0.1726619
```

```r
# the limit distribution
pi <- pp*pi0
barplot(pi, names = k)
```




```
xi <- rho/s
# average number of customers waiting
Lq <- rho^(s+1)/(factorial(s)*s)/(1-xi)^2*pi0
# average time in the queue
Wq <- Lq/lam
# average time in the system
W <- Wq + 1/mu
# expected number of customers in the system
L <- lam*W

# the limit distribution
pi <- pp*pi0
pi
```

```
##  [1] 1.726619e-01 2.877698e-01 2.398082e-01 1.332268e-01 7.401486e-02
##  [6] 4.111937e-02 2.284409e-02 1.269116e-02 7.050646e-03 3.917026e-03
## [11] 2.176125e-03 1.208959e-03 6.716436e-04 3.731353e-04 2.072974e-04
## [16] 1.151652e-04 6.398068e-05 3.554482e-05 1.974712e-05 1.097062e-05
## [21] 6.094791e-06
```

e

average number of customers waiting is 0.3747

```
Lq <- rho^(s+1)/(factorial(s)*s)/(1-xi)^2*pi0
Lq
```

```
## [1] 0.3747002
```

f

average time in the queue

average time in queue in mins is 0.018*60 is 1.08 mins.

```
Wq <- Lq/lam
Wq
```

```
## [1] 0.01873501
```