Markov Chains

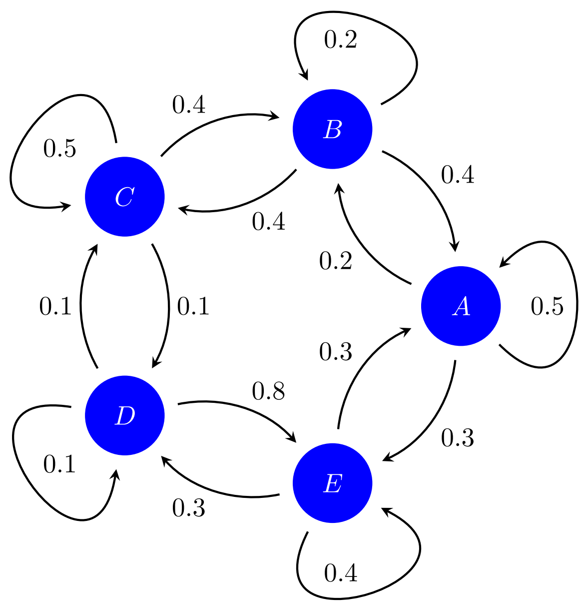
Prof. Wright

April 16, 2018

Exercises

1. Five-State Markov Chain

Consider the following state graph.



Build the 5×5 transition matrix for this state graph. Once you have it,

- Find the steady-state distribution.
- Repeat the agents simulation as above. That is, show that, starting with a reasonably large number of agents, that after a few hundred simulations, that the flow of agents in and out of each state are roughly balanced.

```
# define the transition matrix
P <- matrix(c(0.5,0.2,0,0,0.3, 0.4,0.2,0.4,0,0, 0,0.4,0.5,0.1,0, 0,0,0.1,0.1,0.8,
              0.3,0,0,0.3,0.4), nrow=5)
P
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]  0.5  0.4  0.0  0.0  0.3
## [2,]  0.2  0.2  0.4  0.0  0.0
## [3,]  0.0  0.4  0.5  0.1  0.0
## [4,]  0.0  0.0  0.1  0.1  0.3
## [5,]  0.3  0.0  0.0  0.8  0.4
```

```r
# find the steady-state distribution from an eigenvector
eigenstuff <- eigen(P)
eigenstuff
```

```
## eigen() decomposition
## $values
## [1]  1.0000000  0.7301571  0.4563345 -0.3149830 -0.1715085
##
## $vectors
##           [,1]       [,2]       [,3]         [,4]        [,5]
## [1,] 0.6142319  0.1457051  0.8275856  0.341479720 -0.44827326
## [2,] 0.2961475 -0.3986652  0.1193310 -0.130255228  0.74469549
## [3,] 0.2851791 -0.6012404 -0.3373212 -0.003041781 -0.46751521
## [4,] 0.2413054  0.2108634 -0.3300310  0.545810907  0.16062256
## [5,] 0.6288565  0.6433371 -0.2795644 -0.753993619  0.01047042
```

```r
v <- eigenstuff$vectors[,1]
v <- v/sum(v)
v
```

```
## [1] 0.2973451 0.1433628 0.1380531 0.1168142 0.3044248
```

```r
# define a function to do the agents simulation
agent_sim <- function(numAgents, numSteps){
  # we sill start with a uniform distribuiton of agents
  agents = sample(1:5, numAgents, rep=TRUE)
  numEachState <- tabulate(agents)
  numEachState

  fromState <- matrix(0, nrow=5, ncol=5)  # 5x5 matrix, initially all zeros
  for(m in 1:numSteps){
    # move all agents from each state and track where then end up
    for(i in 1:5){
      # find out where agents go from state i
      nextState <- sample(1:5, numEachState[i], rep=TRUE, prob=P[,i])
      fromState[i,] <- tabulate(nextState, nbins=5) # store counts in row i of fromState
    }
  # add up each column of fromState to find new distribution of agents
  numEachState <- colSums(fromState)
  }
  numEachState # return the final distribution of agents
}

N <- 100000
steadystate <- agent_sim(N, 100)/N
steadystate
```

```
## [1] 0.29537 0.14296 0.14008 0.11700 0.30459
```

Notice that the steady-state distribution of agents is nearly the same as the eigenvector corresponding to eigenvalue 1.

2. Simulating a Game

Construct a transtion matrix `TransProb` for a game that involves moving through 8 positions, arranged in a circle. These positions are the states of a Markov chain; we will denote the states as $S_1, S_2, \ldots, S_8$. S_1 in the first position, S_2 is the second position, and so on.

In the game, suppose that you roll a dice that tells you how many positions to move. After you reach S_8, the next position is S_1, and you go around again.

$$S1 \to S2 \to \cdots \to S8 \to S1 \to \cdots$$

Consider two versions of the game:

- **Version 1:** You move from one state to the next rolling a fair *3-sided die*. Hence if you are in S_3 and roll a 2, you move to S_5.
  If you are in S_7 and roll a 3 you move to S_2.

```
# define a function to move the last k elements of a vector to the beginning of the vector
cycleVec <- function(vec, k){
  front <- tail(vec, k)
  back <- head(vec, length(vec) - k)
  c(front, back)
}

# now build the transition matrix
P <- matrix(0, nrow=8, ncol=8)
probs <- c(1,1,1,rep(0,5))/3
for(i in 1:8){
  P[,i] <- cycleVec(probs, i)
}

P
```

```
##           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
## [1,] 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.3333333 0.3333333
## [2,] 0.3333333 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.3333333
## [3,] 0.3333333 0.3333333 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## [4,] 0.3333333 0.3333333 0.3333333 0.0000000 0.0000000 0.0000000 0.0000000
## [5,] 0.0000000 0.3333333 0.3333333 0.3333333 0.0000000 0.0000000 0.0000000
## [6,] 0.0000000 0.0000000 0.3333333 0.3333333 0.3333333 0.0000000 0.0000000
## [7,] 0.0000000 0.0000000 0.0000000 0.3333333 0.3333333 0.3333333 0.0000000
## [8,] 0.0000000 0.0000000 0.0000000 0.0000000 0.3333333 0.3333333 0.3333333
##           [,8]
## [1,] 0.3333333
## [2,] 0.3333333
## [3,] 0.3333333
## [4,] 0.0000000
## [5,] 0.0000000
## [6,] 0.0000000
## [7,] 0.0000000
## [8,] 0.0000000
```

```
# find the steady-state distribution from an eigenvector
eigenstuff <- eigen(P)
eigenstuff
```

```
## eigen() decomposition
## $values
## [1]  1.0000000+0.0000000i  0.0000000+0.8047379i  0.0000000-0.8047379i
## [4] -0.3333333+0.0000000i -0.3333333+0.0000000i -0.3333333+0.0000000i
## [7]  0.0000000+0.1380712i  0.0000000-0.1380712i
##
## $vectors
##               [,1]                  [,2]                  [,3]
## [1,] -0.3535534+0i  0.3535534+0.0000000i  0.3535534+0.0000000i
## [2,] -0.3535534+0i  0.2500000-0.2500000i  0.2500000+0.2500000i
## [3,] -0.3535534+0i  0.0000000-0.3535534i  0.0000000+0.3535534i
## [4,] -0.3535534+0i -0.2500000-0.2500000i -0.2500000+0.2500000i
## [5,] -0.3535534+0i -0.3535534+0.0000000i -0.3535534+0.0000000i
## [6,] -0.3535534+0i -0.2500000+0.2500000i -0.2500000-0.2500000i
## [7,] -0.3535534+0i  0.0000000+0.3535534i  0.0000000-0.3535534i
## [8,] -0.3535534+0i  0.2500000+0.2500000i  0.2500000-0.2500000i
##                 [,4]           [,5]          [,6]                  [,7]
## [1,] -0.008001267+0i  0.40889001+0i -0.6123724+0i  0.0000000-0.3535534i
## [2,]  0.096720699+0i -0.56072589+0i  0.2041241+0i -0.2500000+0.2500000i
## [3,] -0.537637283+0i  0.13451986+0i  0.2041241+0i  0.3535534+0.0000000i
## [4,]  0.448917852+0i  0.01731602+0i  0.2041241+0i -0.2500000-0.2500000i
## [5,] -0.008001267+0i  0.40889001+0i -0.6123724+0i  0.0000000+0.3535534i
## [6,]  0.096720699+0i -0.56072589+0i  0.2041241+0i  0.2500000-0.2500000i
## [7,] -0.537637283+0i  0.13451986+0i  0.2041241+0i -0.3535534+0.0000000i
## [8,]  0.448917852+0i  0.01731602+0i  0.2041241+0i  0.2500000+0.2500000i
##                       [,8]
## [1,]  0.0000000+0.3535534i
## [2,] -0.2500000-0.2500000i
## [3,]  0.3535534+0.0000000i
## [4,] -0.2500000+0.2500000i
## [5,]  0.0000000-0.3535534i
## [6,]  0.2500000+0.2500000i
## [7,] -0.3535534-0.0000000i
## [8,]  0.2500000-0.2500000i
```

```r
v <- eigenstuff$vectors[,1]
v <- v/sum(v)
as.numeric(v)
```

```
## [1] 0.125 0.125 0.125 0.125 0.125 0.125 0.125 0.125
```

- **Version 2:** Same as Version 1, but if you land in S_5, then your next move is to S_1*.

```r
# build the transition matrix
# start with the previous matrix P, but change column 5
P2 <- P
P2[,5] <- c(1,rep(0,7))
P2
```

```
##           [,1]      [,2]      [,3]      [,4] [,5]      [,6]      [,7]
## [1,] 0.0000000 0.0000000 0.0000000 0.0000000    1 0.3333333 0.3333333
## [2,] 0.3333333 0.0000000 0.0000000 0.0000000    0 0.0000000 0.3333333
## [3,] 0.3333333 0.3333333 0.0000000 0.0000000    0 0.0000000 0.0000000
## [4,] 0.3333333 0.3333333 0.3333333 0.0000000    0 0.0000000 0.0000000
## [5,] 0.0000000 0.3333333 0.3333333 0.3333333    0 0.0000000 0.0000000
## [6,] 0.0000000 0.0000000 0.3333333 0.3333333    0 0.0000000 0.0000000
```

```
## [7,] 0.0000000 0.0000000 0.0000000 0.3333333    0 0.3333333 0.0000000
## [8,] 0.0000000 0.0000000 0.0000000 0.0000000    0 0.3333333 0.3333333
##           [,8]
## [1,] 0.3333333
## [2,] 0.3333333
## [3,] 0.3333333
## [4,] 0.0000000
## [5,] 0.0000000
## [6,] 0.0000000
## [7,] 0.0000000
## [8,] 0.0000000
```

```
# find the steady-state distribution from an eigenvector
eigenstuff <- eigen(P2)
eigenstuff
```

```
## eigen() decomposition
## $values
## [1]  1.00000000+0.0000000i -0.19854707+0.7358353i -0.19854707-0.7358353i
## [4] -0.33333333+0.0000000i  0.00000000+0.3333333i  0.00000000-0.3333333i
## [7] -0.33333333+0.0000000i  0.06376081+0.0000000i
##
## $vectors
##              [,1]                   [,2]                   [,3]
## [1,] 0.5748586+0i -0.5992625+0.0000000i -0.5992625+0.0000000i
## [2,] 0.3193659+0i  0.0858604+0.3765916i  0.0858604-0.3765916i
## [3,] 0.3513025+0i  0.2679435+0.2278770i  0.2679435-0.2278770i
## [4,] 0.4151756+0i  0.2832077+0.0347761i  0.2832077-0.0347761i
## [5,] 0.3619480+0i  0.1973473-0.3418155i  0.1973473+0.3418155i
## [6,] 0.2554927+0i  0.0481113-0.2626531i  0.0481113+0.2626531i
## [7,] 0.2235561+0i -0.1339718-0.1139385i -0.1339718+0.1139385i
## [8,] 0.1596829+0i -0.1492359+0.0791624i -0.1492359-0.0791624i
##                  [,4]                   [,5]                   [,6]
## [1,]  1.048449e-16+0i -0.5590170+0.0000000i -0.5590170+0.0000000i
## [2,] -1.516691e-01+0i  0.3913119+0.0559017i  0.3913119-0.0559017i
## [3,] -4.066044e-01+0i  0.1118034-0.2236068i  0.1118034+0.2236068i
## [4,]  5.582735e-01+0i -0.1677051+0.0559017i -0.1677051-0.0559017i
## [5,] -6.367975e-17+0i -0.1118034-0.3354102i -0.1118034+0.3354102i
## [6,] -1.516691e-01+0i -0.1677051+0.0559017i -0.1677051-0.0559017i
## [7,] -4.066044e-01+0i  0.1118034+0.3354102i  0.1118034-0.3354102i
## [8,]  5.582735e-01+0i  0.3913119+0.0559017i  0.3913119-0.0559017i
##                  [,7]          [,8]
## [1,]  3.568009e-16+0i  0.3810663+0i
## [2,]  5.664628e-01+0i  0.1091959+0i
## [3,] -3.798745e-01+0i -0.4213663+0i
## [4,] -1.865883e-01+0i  0.3601791+0i
## [5,] -5.093675e-17+0i  0.2509831+0i
## [6,]  5.664628e-01+0i -0.3198791+0i
## [7,] -3.798745e-01+0i  0.2106832+0i
## [8,] -1.865883e-01+0i -0.5708622+0i
```

```
v <- eigenstuff$vectors[,1]
v <- v/sum(v)
as.numeric(v)
```

```
## [1] 0.216 0.120 0.132 0.156 0.136 0.096 0.084 0.060
```

Construct transition matrices for both games. For each, explore the steady-state behavior. In other words, what is the probability of being in a particular state, say S_3, after a large number of rolls? Is it different for the the different versions?