HW06

Zach White

April 4, 2017

Problem 3

Part B

```
x1 = c(0,1/2,1/2,0,0)
x2 = c(1/3,0,0,1/3,0)
x4 = c(1/3,0,0,1/3,0)
x5 = c(0,1/3,1/3,0,1/3)
x6 = c(0,0,0,1/2,0)

x3 = c(0,1/3,0,0,1/2)
x7 = c(0,0,1/3,0,0)

R = cbind(x3,x7)

Q = rbind(x1,x2,x4,x5,x6)

I.Q = diag(1,5) - Q
inv.I.Q = solve(I.Q)
inv.I.Q %*% rep(1,5)
```

```
##          [,1]
## [1,] 5.666667
## [2,] 4.666667
## [3,] 4.666667
## [4,] 5.333333
## [5,] 3.666667
```

```
inv.I.Q %*% R
```

```
##             x3        x7
## [1,] 0.5833333 0.4166667
## [2,] 0.7500000 0.2500000
## [3,] 0.4166667 0.5833333
## [4,] 0.6666667 0.3333333
## [5,] 0.8333333 0.1666667
```

Problem 4

```
x1 = c(1,0,0,0,0,0,0,0,0,0)
x2 = c(-1/2,1,0,0,0,0,0,0,0,0)
x3 = c(-1/3,-1/3,1,0,0,0,0,0,0,0)
x4 = c(-1/4,-1/4,-1/4,1,0,0,0,0,0,0)
x5 = c(-1/5,-1/5,-1/5,-1/5,1,0,0,0,0,0)
```

```
x6 = c(-1/6,-1/6,-1/6,-1/6,-1/6,1,0,0,0,0)
x7 = c(-1/7,-1/7,-1/7,-1/7,-1/7,-1/7,1,0,0,0)
x8 = c(-1/8,-1/8,-1/8,-1/8,-1/8,-1/8,-1/8,1,0,0)
x9 = c(-1/9,-1/9,-1/9,-1/9,-1/9,-1/9,-1/9,-1/9,1,0)
x10 = c(-1/10,-1/10,-1/10,-1/10,-1/10,-1/10,-1/10,-1/10,-1/10,1)

I.Q = rbind(x1,x2,x3,x4,x5,x6,x7,x8,x9,x10)
solve(I.Q)
```

```
##        x1        x2   x3  x4        x5        x6    x7        x8  x9 x10
##  [1,] 1.0 0.0000000 0.00 0.0 0.0000000 0.0000000 0.000 0.0000000 0.0   0
##  [2,] 0.5 1.0000000 0.00 0.0 0.0000000 0.0000000 0.000 0.0000000 0.0   0
##  [3,] 0.5 0.3333333 1.00 0.0 0.0000000 0.0000000 0.000 0.0000000 0.0   0
##  [4,] 0.5 0.3333333 0.25 1.0 0.0000000 0.0000000 0.000 0.0000000 0.0   0
##  [5,] 0.5 0.3333333 0.25 0.2 1.0000000 0.0000000 0.000 0.0000000 0.0   0
##  [6,] 0.5 0.3333333 0.25 0.2 0.1666667 1.0000000 0.000 0.0000000 0.0   0
##  [7,] 0.5 0.3333333 0.25 0.2 0.1666667 0.1428571 1.000 0.0000000 0.0   0
##  [8,] 0.5 0.3333333 0.25 0.2 0.1666667 0.1428571 0.125 1.0000000 0.0   0
##  [9,] 0.5 0.3333333 0.25 0.2 0.1666667 0.1428571 0.125 0.1111111 1.0   0
## [10,] 0.5 0.3333333 0.25 0.2 0.1666667 0.1428571 0.125 0.1111111 0.1   1
```

Problem 5

Part A

```
x1 = c(0,1/2,0,1/2,0,0,0)
x2 = c(1/3,0,1/3,0,1/3,0,0)
x3= c(0,1/2,0,0,1/2,0,0)
x4 = c(1/3,0,0,0,1/3,0,1/3)
x5 = c(0,1/3,0,1/3,0,1/3,0)
x6 = c(0,0,1/2,0,1/2,0,0)
x7 = c(0,0,0,0,0,0,1)

P = rbind(x1,x2,x3,x4,x5,x6,x7)
P %^% 100
```

```
##            [,1]         [,2]         [,3]         [,4]         [,5]
## x1 0.0005050308 0.0008404005 0.0004683869 0.0005928308 0.0009248537
## x2 0.0005833802 0.0009707786 0.0005410516 0.0006848014 0.0010683337
## x3 0.0005995732 0.0009977247 0.0005560697 0.0007038096 0.0010979877
## x4 0.0003721073 0.0006192082 0.0003451081 0.0004367985 0.0006814334
## x5 0.0005509760 0.0009168561 0.0005109985 0.0006467636 0.0010089924
## x6 0.0006081322 0.0010119673 0.0005640076 0.0007138565 0.0011136615
## x7 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
##            [,6]      [,7]
## x1 0.0003258927 0.9963426
## x2 0.0003764510 0.9957752
## x3 0.0003869002 0.9956579
## x4 0.0002401181 0.9973052
## x5 0.0003555408 0.9960099
## x6 0.0003924233 0.9955960
## x7 0.0000000000 1.0000000
```

Part B

This is under my formulation for the problem because I wasn't aware of the last, qualifyin statement saying that the stat before the exit wasn't an absorption. I take that state as absorbing and then in my next simulation. I do it according to the way it should be.

```
n.samps = 10000
food.before = NULL
for(j in 1:n.samps){
  chain = NULL
  chain[1] = 4
  i = 1
  while(chain[i] != 7){
    if(chain[i] == 1){chain[i+1] = sample(c(2,4),1)}
    else if(chain[i] == 2){chain[i+1] = sample(c(1,3,5),1)}
    else if(chain[i] == 3){chain[i+1] = sample(c(2,6),1)}
    else if(chain[i] == 4){chain[i+1] = sample(c(1,5,7),1)}
    else if(chain[i] == 5){chain[i+1] = sample(c(4,2,6),1)}
    else if(chain[i] == 6){chain[i+1] = sample(c(3,5),1)}
    #if(chain[i] == 7){chain[i+1] = sample(c(4,8))}
    if(chain[i] != 7){
      i = i+1
    }
  }
  if(is.element(3,chain)){
    food.before[j] = 1
  }
  else{
    food.before[j] = 0
  }
}
chain
```

```
## [1] 4 7
```

```
mean(food.before)
```

```
## [1] 0.415
```

```
n.samps = 10000
food.before = NULL
for(j in 1:n.samps){
  chain = NULL
  chain[1] = 4
  i = 1
  while(chain[i] != 8){
    if(chain[i] == 1){chain[i+1] = sample(c(2,4),1)}
    else if(chain[i] == 2){chain[i+1] = sample(c(1,3,5),1)}
    else if(chain[i] == 3){chain[i+1] = sample(c(2,6),1)}
    else if(chain[i] == 4){chain[i+1] = sample(c(1,5,7),1)}
    else if(chain[i] == 5){chain[i+1] = sample(c(4,2,6),1)}
    else if(chain[i] == 6){chain[i+1] = sample(c(3,5),1)}
    else if(chain[i] == 7){chain[i+1] = sample(c(4,8),1)}
    if(chain[i] != 8){
      i = i+1
    }
```

```
  }
  if(is.element(3,chain)){
    food.before[j] = 1
  }
  else{
    food.before[j] = 0
  }
}
chain
```

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

```
mean(food.before)
```

```
## [1] 0.5859
```