605-HW10-MarkovChains-RandomWalks

Michael Y.

November 3, 2019

Contents

```
### load some libraries
library(kableExtra)
library(expm)      # needed for matrix power operator %^%
```

HW10 - Markov Chains, Random Walks

Smith is in jail and has 1 dollar; he can get out on bail if he has 8 dollars.
A guard agrees to make a series of bets with him.

If Smith bets A dollars,

- he wins A dollars with probability .4 ,
- and loses A dollars with probability .6 .

Find the probability that he wins 8 dollars before losing all of his money if:

(a) he bets 1 dollar each time (***timid strategy***).

(b) he bets, each time, as much as possible but not more than necessary to bring his fortune up to 8 dollars (***bold strategy***).

(c) Which strategy gives Smith the better chance of getting out of jail?

(a) timid strategy

The problem can be solved in multiple ways. Here are three:

(a1) Gambler's Ruin:

This is the classic "Gambler's Ruin" problem, for which the solution is

$P_i(N) = \frac{1-(\frac{q}{p})^i}{1-(\frac{q}{p})^N}$, if $p \neq q$.

Here, $i = 1$ and $N = 8$, so we get:

```
p=0.4
q=1-p
i=1
N=8
numerator = 1-(q/p)^i
denominator = 1-(q/p)^N
result = numerator / denominator
result
```

```
## [1] 0.0203013481
```

The probability of success under the "timid" strategy is 0.0203 .

(a2) Absorbing Markov Chains:

```
#### Probabilities of win or lose
p=0.4
q=1-p

#### timid Markov transition probabilities
nodes=0:8
timidmarkov=matrix(
  c(
  1,0,0,0,0,0,0,0,0,
  q,0,p,0,0,0,0,0,0,
  0,q,0,p,0,0,0,0,0,
  0,0,q,0,p,0,0,0,0,
  0,0,0,q,0,p,0,0,0,
  0,0,0,0,q,0,p,0,0,
  0,0,0,0,0,q,0,p,0,
  0,0,0,0,0,0,q,0,p,
  0,0,0,0,0,0,0,0,1
),nrow = 9, ncol = 9, byrow = T, dimnames = list(nodes,nodes)
)

#### timid markov matrix
timidmarkov
```

```
##     0   1   2   3   4   5   6   7   8
## 0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
## 1 0.6 0.0 0.4 0.0 0.0 0.0 0.0 0.0 0.0
## 2 0.0 0.6 0.0 0.4 0.0 0.0 0.0 0.0 0.0
## 3 0.0 0.0 0.6 0.0 0.4 0.0 0.0 0.0 0.0
## 4 0.0 0.0 0.0 0.6 0.0 0.4 0.0 0.0 0.0
## 5 0.0 0.0 0.0 0.0 0.6 0.0 0.4 0.0 0.0
## 6 0.0 0.0 0.0 0.0 0.0 0.6 0.0 0.4 0.0
## 7 0.0 0.0 0.0 0.0 0.0 0.0 0.6 0.0 0.4
## 8 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0
```

```
#### Canonical order, P
canonicalorder=c("1","2","3","4","5","6","7","0","8")
Ptimid=timidmarkov[canonicalorder,canonicalorder]
Ptimid
```

```
##     1   2   3   4   5   6   7   0   8
## 1 0.0 0.4 0.0 0.0 0.0 0.0 0.0 0.6 0.0
## 2 0.6 0.0 0.4 0.0 0.0 0.0 0.0 0.0 0.0
## 3 0.0 0.6 0.0 0.4 0.0 0.0 0.0 0.0 0.0
## 4 0.0 0.0 0.6 0.0 0.4 0.0 0.0 0.0 0.0
## 5 0.0 0.0 0.0 0.6 0.0 0.4 0.0 0.0 0.0
## 6 0.0 0.0 0.0 0.0 0.6 0.0 0.4 0.0 0.0
## 7 0.0 0.0 0.0 0.0 0.0 0.6 0.0 0.0 0.4
## 0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0
## 8 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0
```

```
#### Canonical transitions, Q
Qtimid = Ptimid[1:7,1:7]
Qtimid
```

```
##     1   2   3   4   5   6   7
## 1 0.0 0.4 0.0 0.0 0.0 0.0 0.0
## 2 0.6 0.0 0.4 0.0 0.0 0.0 0.0
## 3 0.0 0.6 0.0 0.4 0.0 0.0 0.0
## 4 0.0 0.0 0.6 0.0 0.4 0.0 0.0
## 5 0.0 0.0 0.0 0.6 0.0 0.4 0.0
## 6 0.0 0.0 0.0 0.0 0.6 0.0 0.4
## 7 0.0 0.0 0.0 0.0 0.0 0.6 0.0
```

```
#### Transient-to-absorbing, R
Rtimid = Ptimid[1:7,8:9]
Rtimid
```

```
##     0   8
## 1 0.6 0.0
## 2 0.0 0.0
## 3 0.0 0.0
## 4 0.0 0.0
## 5 0.0 0.0
## 6 0.0 0.0
## 7 0.0 0.4
```

```
#### Identity matrix, I
I = diag(7)
I
```

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

```
#### Identity minus Q
ImQtimid = I-Qtimid
ImQtimid
```

```
##      1    2    3    4    5    6    7
## 1  1.0 -0.4  0.0  0.0  0.0  0.0  0.0
## 2 -0.6  1.0 -0.4  0.0  0.0  0.0  0.0
## 3  0.0 -0.6  1.0 -0.4  0.0  0.0  0.0
## 4  0.0  0.0 -0.6  1.0 -0.4  0.0  0.0
## 5  0.0  0.0  0.0 -0.6  1.0 -0.4  0.0
## 6  0.0  0.0  0.0  0.0 -0.6  1.0 -0.4
## 7  0.0  0.0  0.0  0.0  0.0 -0.6  1.0
```

```
#### N, the Fundamental Matrix
Ntimid=solve(ImQtimid)
Ntimid
```

```
##             1           2           3           4           5           6           7
## 1 1.632831086 1.054718477 0.669310071 0.412371134 0.241078509 0.126883426 0.0507533703
## 2 1.582077716 2.636796193 1.673275178 1.030927835 0.602696273 0.317208565 0.1268834259
## 3 1.505947661 2.509912768 3.179222839 1.958762887 1.145122918 0.602696273 0.2410785091
## 4 1.391752577 2.319587629 2.938144330 3.350515464 1.958762887 1.030927835 0.4123711340
## 5 1.220459952 2.034099921 2.576526566 2.938144330 3.179222839 1.673275178 0.6693100714
## 6 0.963521015 1.605868358 2.034099921 2.319587629 2.509912768 2.636796193 1.0547184774
## 7 0.578112609 0.963521015 1.220459952 1.391752577 1.505947661 1.582077716 1.6328310864
```

```
#### Expected time to absorption, basaed upon starting value
c = rep(1,7)
Nctimid = Ntimid %*% c
Nctimid
```

```
##          [,1]
## 1  4.18794607
## 2  7.96986519
## 3 11.14274385
## 4 13.40206186
## 5 14.29103886
## 6 13.12450436
## 7  8.87470262
```

```
#### Absorption probabilities, B
Btimid = Ntimid %*% Rtimid
Btimid
```

```
##             0           8
## 1 0.979698652 0.0203013481
## 2 0.949246630 0.0507533703
## 3 0.903568596 0.0964314036
## 4 0.835051546 0.1649484536
## 5 0.732275971 0.2677240285
## 6 0.578112609 0.4218873910
## 7 0.346867565 0.6531324346
```

```
#### Probability of success, starting from 1 dollar:
p1timid = Btimid["1","8"]
p1timid
```

```
## [1] 0.0203013481
```

Again, the probability is 0.0203 .

(a3) Repeated matrix multiplications:

```
p=0.4
q=1-p
nodes=0:8
timidmarkov=matrix(
  c(
  1,0,0,0,0,0,0,0,0,
  q,0,p,0,0,0,0,0,0,
  0,q,0,p,0,0,0,0,0,
  0,0,q,0,p,0,0,0,0,
  0,0,0,q,0,p,0,0,0,
  0,0,0,0,q,0,p,0,0,
  0,0,0,0,0,q,0,p,0,
  0,0,0,0,0,0,q,0,p,
  0,0,0,0,0,0,0,0,1
),nrow = 9, ncol = 9, byrow = T, dimnames = list(nodes,nodes)
)
#### timid markov matrix
timidmarkov
```

```
##     0   1   2   3   4   5   6   7   8
## 0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
## 1 0.6 0.0 0.4 0.0 0.0 0.0 0.0 0.0 0.0
## 2 0.0 0.6 0.0 0.4 0.0 0.0 0.0 0.0 0.0
## 3 0.0 0.0 0.6 0.0 0.4 0.0 0.0 0.0 0.0
## 4 0.0 0.0 0.0 0.6 0.0 0.4 0.0 0.0 0.0
## 5 0.0 0.0 0.0 0.0 0.6 0.0 0.4 0.0 0.0
## 6 0.0 0.0 0.0 0.0 0.0 0.6 0.0 0.4 0.0
## 7 0.0 0.0 0.0 0.0 0.0 0.0 0.6 0.0 0.4
## 8 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0
```

```
timidresult    = vector()
timidtransient = vector()
timiddiff      = vector()
epsilon = 1e-8

#### Loop until absorption converges (transient states empty)
for (i in 1:300) {
  timidpower = timidmarkov %^% i                    # raise markov matrix to power i
  timidresult[i]=timidpower["1","8"]                # get the probability of 1 --> 8 after i steps
  timidtransient[i] = sum(colSums(timidpower)[2:7]) # add up probabilities of transient states
  if (i>1)
    timiddiff[i]=timidresult[i]-timidresult[i-1]    # change of probability vs. prior step
  else timiddiff[i]=999                             # just for first step

#  print(paste(i,
#              round(timidtransient[i],12),
#              round(timidresult[i],   9),
#              round(timiddiff[i],        12)
#              )
#       )
```

```
    if (timidtransient[i] < epsilon) {
    print(paste("Absorbed at power ", i))
    break
  }
}
```

```
## [1] "Absorbed at power  205"
```

```
power = 1:i
# results for each step
tempresult = cbind(power, timidtransient, timidresult, timiddiff)

head(tempresult,15) %>%
  kable() %>%
kable_styling(c("striped","bordered"))
```

power	timidtransient	timidresult	timiddiff
1	5.60000000	0.000000000	999.000000000
2	5.08000000	0.000000000	0.000000000
3	4.70400000	0.000000000	0.000000000
4	4.29920000	0.000000000	0.000000000
5	3.96352000	0.000000000	0.000000000
6	3.61401600	0.000000000	0.000000000
7	3.30762240	0.001638400	0.001638400
8	3.00874752	0.001638400	0.000000000
9	2.74010112	0.003997696	0.002359296
10	2.48865178	0.003997696	0.000000000
11	2.25972191	0.006451364	0.002453668
12	2.05046061	0.006451364	0.000000000
13	1.85859960	0.008716288	0.002264924
14	1.68557157	0.008716288	0.000000000
15	1.52630023	0.010694926	0.001978638

```
tail(tempresult,15) %>%
  kable() %>%
kable_styling(c("striped","bordered"))
```

	power	timidtransient	timidresult	timiddiff
[191,]	191	0.000000037	0.020301348	0
[192,]	192	0.000000034	0.020301348	0
[193,]	193	0.000000031	0.020301348	0
[194,]	194	0.000000028	0.020301348	0
[195,]	195	0.000000025	0.020301348	0
[196,]	196	0.000000023	0.020301348	0
[197,]	197	0.000000021	0.020301348	0
[198,]	198	0.000000019	0.020301348	0
[199,]	199	0.000000017	0.020301348	0
[200,]	200	0.000000015	0.020301348	0
[201,]	201	0.000000014	0.020301348	0
[202,]	202	0.000000013	0.020301348	0
[203,]	203	0.000000011	0.020301348	0
[204,]	204	0.000000010	0.020301348	0
[205,]	205	0.000000009	0.020301348	0

```
# timid markov raised to power i
print(paste("i: ",i))
```

```
## [1] "i:  205"
```

```
round(timidpower,8)
```

```
##            0 1 2 3 4 5 6 7          8
## 0 1.00000000 0 0 0 0 0 0 0 0.00000000
## 1 0.97969865 0 0 0 0 0 0 0 0.02030135
## 2 0.94924663 0 0 0 0 0 0 0 0.05075337
## 3 0.90356860 0 0 0 0 0 0 0 0.09643140
## 4 0.83505154 0 0 0 0 0 0 0 0.16494845
## 5 0.73227597 0 0 0 0 0 0 0 0.26772403
## 6 0.57811261 0 0 0 0 0 0 0 0.42188739
## 7 0.34686756 0 0 0 0 0 0 0 0.65313243
## 8 0.00000000 0 0 0 0 0 0 0 1.00000000
```

```
# Timid result
TimidProbability = round(timidresult[i],9)
print(paste("Timid result: ",TimidProbability))
```

```
## [1] "Timid result:  0.020301348"
```

Yet again, the probability of winning with the timid strategy is 0.0203 .

(b) bold strategy

Here are three different ways to obtain the answer:

(b1) only three plays

Under the Bold strategy, the prisoner bets his entire bankroll each time (as it's not possible to reach any value in $\{5, 6, 7\}$ when starting from 1 dollar.)

Thus, to win the game, he must win three times:

- Bet 1: 1 dollar increases to 2 dollars
- Bet 2: 2 dollars increases to 4 dollars
- Bet 3: 4 dollars increases to 8 dollars

A loss on any of the above bets will "wipe out" his bankroll, forcing him to stop playing.

Because his chance of winning on each bet is 0.4, his chance of winning three times is $(0.4)^3 = 0.064$.

Thus, his probability of winning under this strategy is 6.4 percent.

(b2) Absorbing Markov Chains:

```
# Probabilities of win or lose
p=0.4
q=1-p

# bold Markov transition probabilities
nodes=0:8
boldmarkov=matrix(
  c(
1,0,0,0,0,0,0,0,0,   # have zero - game lost
q,0,p,0,0,0,0,0,0,   # have one -- bet one -- move to zero or two
q,0,0,0,p,0,0,0,0,   # have two -- bet two -- move to zero or four
q,0,0,0,0,0,p,0,0,   # have three -- bet THREE -- move to ZERO or SIX
q,0,0,0,0,0,0,0,p,   # have four -- bet four -- move to zero or eight
0,0,q,0,0,0,0,0,p,   # have five -- bet three -- move to two or eight
0,0,0,0,q,0,0,0,p,   # have six -- bet two -- move to four or eight
0,0,0,0,0,0,q,0,p,   # have seven -- bet one -- move to six or eight
0,0,0,0,0,0,0,0,1),  # have eight -- game won
nrow = 9, ncol = 9, byrow = T, dimnames = list(nodes,nodes)
)
# Canonical order, P
canonicalorder=c("1","2","3","4","5","6","7","0","8")
Pbold=boldmarkov[canonicalorder,canonicalorder]
Pbold
```

```
##   1   2 3   4 5   6 7   0   8
## 1 0 0.4 0 0.0 0 0.0 0 0.6 0.0
## 2 0 0.0 0 0.4 0 0.0 0 0.6 0.0
## 3 0 0.0 0 0.0 0 0.4 0 0.6 0.0
## 4 0 0.0 0 0.0 0 0.0 0 0.6 0.4
## 5 0 0.6 0 0.0 0 0.0 0 0.0 0.4
## 6 0 0.0 0 0.6 0 0.0 0 0.0 0.4
## 7 0 0.0 0 0.0 0 0.6 0 0.0 0.4
## 0 0 0.0 0 0.0 0 0.0 0 1.0 0.0
## 8 0 0.0 0 0.0 0 0.0 0 0.0 1.0
```

```
# Canonical transitions, Q
Qbold = Pbold[1:7,1:7]
Qbold
```

```
##   1   2 3   4 5   6 7
## 1 0 0.4 0 0.0 0 0.0 0
## 2 0 0.0 0 0.4 0 0.0 0
## 3 0 0.0 0 0.0 0 0.4 0
## 4 0 0.0 0 0.0 0 0.0 0
## 5 0 0.6 0 0.0 0 0.0 0
## 6 0 0.0 0 0.6 0 0.0 0
## 7 0 0.0 0 0.0 0 0.6 0
```

```
# Transient-to-absorbing, R
Rbold = Pbold[1:7,8:9]
Rbold
```

```
##     0   8
## 1 0.6 0.0
## 2 0.6 0.0
## 3 0.6 0.0
## 4 0.6 0.4
## 5 0.0 0.4
## 6 0.0 0.4
## 7 0.0 0.4
```

```
# Identity matrix, I
I = diag(7)
I
```

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

```
# Identity minus Q
ImQbold = I-Qbold
ImQbold
```

```
##   1    2 3    4 5    6 7
## 1 1 -0.4 0  0.0 0  0.0 0
## 2 0  1.0 0 -0.4 0  0.0 0
## 3 0  0.0 1  0.0 0 -0.4 0
## 4 0  0.0 0  1.0 0  0.0 0
## 5 0 -0.6 0  0.0 1  0.0 0
## 6 0  0.0 0 -0.6 0  1.0 0
## 7 0  0.0 0  0.0 0 -0.6 1
```

```
# N, the Fundamental Matrix
Nbold=solve(ImQbold)
Nbold
```

```
##   1   2 3    4 5   6 7
## 1 1 0.4 0 0.16 0 0.0 0
## 2 0 1.0 0 0.40 0 0.0 0
## 3 0 0.0 1 0.24 0 0.4 0
## 4 0 0.0 0 1.00 0 0.0 0
## 5 0 0.6 0 0.24 1 0.0 0
## 6 0 0.0 0 0.60 0 1.0 0
## 7 0 0.0 0 0.36 0 0.6 1
```

```r
# Expected time to absorption
c = rep(1,7)
Ncbold = Nbold %*% c
Ncbold
```

```
##   [,1]
## 1 1.56
## 2 1.40
## 3 1.64
## 4 1.00
## 5 1.84
## 6 1.60
## 7 1.96
```

```r
# Absorption probabilities, B
Bbold = Nbold %*% Rbold
Bbold
```

```
##       0     8
## 1 0.936 0.064
## 2 0.840 0.160
## 3 0.744 0.256
## 4 0.600 0.400
## 5 0.504 0.496
## 6 0.360 0.640
## 7 0.216 0.784
```

```r
# Probability of success, starting from 1 dollar:
p1bold = Bbold["1","8"]
p1bold
```

```
## [1] 0.064
```

Again, the probability is 0.064 .

(b3) Repeated matrix multiplications:

```
p=0.4
q=1-p
nodes=0:8
boldmarkov
```

```
##     0 1   2 3   4 5   6 7   8
## 0 1.0 0 0.0 0 0.0 0 0.0 0 0.0
## 1 0.6 0 0.4 0 0.0 0 0.0 0 0.0
## 2 0.6 0 0.0 0 0.4 0 0.0 0 0.0
## 3 0.6 0 0.0 0 0.0 0 0.4 0 0.0
## 4 0.6 0 0.0 0 0.0 0 0.0 0 0.4
## 5 0.0 0 0.6 0 0.0 0 0.0 0 0.4
## 6 0.0 0 0.0 0 0.6 0 0.0 0 0.4
## 7 0.0 0 0.0 0 0.0 0 0.6 0 0.4
## 8 0.0 0 0.0 0 0.0 0 0.0 0 1.0
```

```
boldmarkov
```

```
##     0 1   2 3   4 5   6 7   8
## 0 1.0 0 0.0 0 0.0 0 0.0 0 0.0
## 1 0.6 0 0.4 0 0.0 0 0.0 0 0.0
## 2 0.6 0 0.0 0 0.4 0 0.0 0 0.0
## 3 0.6 0 0.0 0 0.0 0 0.4 0 0.0
## 4 0.6 0 0.0 0 0.0 0 0.0 0 0.4
## 5 0.0 0 0.6 0 0.0 0 0.0 0 0.4
## 6 0.0 0 0.0 0 0.6 0 0.0 0 0.4
## 7 0.0 0 0.0 0 0.0 0 0.6 0 0.4
## 8 0.0 0 0.0 0 0.0 0 0.0 0 1.0
```

```
boldresult = vector()
boldtransient = vector()
epsilon = 1e-8
for (i in 1:10) {
  boldpower = boldmarkov %^% i

  # boldmarkov raised to power i
  print(paste("i: ",i))
  print(round(boldpower,8))

  boldresult[i]=boldpower["1","8"]
  boldtransient[i] = sum(colSums(boldpower)[2:7])
  if (i>1)
    bolddiff=boldresult[i]-boldresult[i-1]
  else bolddiff=999

  print(paste(i,round(boldtransient[i],10),round(boldresult[i],   8),round(bolddiff,10)))

  if (boldtransient[i] < epsilon) {
    print(paste("Absorbed at power ", i))
    break
```

```
  }

}
```

```
## [1] "i:  1"
##     0 1   2 3   4 5   6 7   8
## 0 1.0 0 0.0 0 0.0 0 0.0 0 0.0
## 1 0.6 0 0.4 0 0.0 0 0.0 0 0.0
## 2 0.6 0 0.0 0 0.4 0 0.0 0 0.0
## 3 0.6 0 0.0 0 0.0 0 0.4 0 0.0
## 4 0.6 0 0.0 0 0.0 0 0.0 0 0.4
## 5 0.0 0 0.6 0 0.0 0 0.0 0 0.4
## 6 0.0 0 0.0 0 0.6 0 0.0 0 0.4
## 7 0.0 0 0.0 0 0.0 0 0.6 0 0.4
## 8 0.0 0 0.0 0 0.0 0 0.0 0 1.0
## [1] "1 3 0 999"
## [1] "i:  2"
##      0 1 2 3    4 5 6 7    8
## 0 1.00 0 0 0 0.00 0 0 0 0.00
## 1 0.84 0 0 0 0.16 0 0 0 0.00
## 2 0.84 0 0 0 0.00 0 0 0 0.16
## 3 0.60 0 0 0 0.24 0 0 0 0.16
## 4 0.60 0 0 0 0.00 0 0 0 0.40
## 5 0.36 0 0 0 0.24 0 0 0 0.40
## 6 0.36 0 0 0 0.00 0 0 0 0.64
## 7 0.00 0 0 0 0.36 0 0 0 0.64
## 8 0.00 0 0 0 0.00 0 0 0 1.00
## [1] "2 1 0 0"
## [1] "i:  3"
##       0 1 2 3 4 5 6 7     8
## 0 1.000 0 0 0 0 0 0 0 0.000
## 1 0.936 0 0 0 0 0 0 0 0.064
## 2 0.840 0 0 0 0 0 0 0 0.160
## 3 0.744 0 0 0 0 0 0 0 0.256
## 4 0.600 0 0 0 0 0 0 0 0.400
## 5 0.504 0 0 0 0 0 0 0 0.496
## 6 0.360 0 0 0 0 0 0 0 0.640
## 7 0.216 0 0 0 0 0 0 0 0.784
## 8 0.000 0 0 0 0 0 0 0 1.000
## [1] "3 0 0.064 0.064"
## [1] "Absorbed at power  3"
```

```
# Converged power matrix
boldpower
```

```
##       0 1 2 3 4 5 6 7     8
## 0 1.000 0 0 0 0 0 0 0 0.000
## 1 0.936 0 0 0 0 0 0 0 0.064
## 2 0.840 0 0 0 0 0 0 0 0.160
## 3 0.744 0 0 0 0 0 0 0 0.256
## 4 0.600 0 0 0 0 0 0 0 0.400
## 5 0.504 0 0 0 0 0 0 0 0.496
## 6 0.360 0 0 0 0 0 0 0 0.640
```

```
## 7 0.216 0 0 0 0 0 0 0 0.784
## 8 0.000 0 0 0 0 0 0 0 1.000
```

```
# Sums of columns
print("Column sums: ")
```

```
## [1] "Column sums: "
```

```
print(round(colSums(boldpower),8))
```

```
##   0   1   2   3   4   5   6   7   8
## 5.2 0.0 0.0 0.0 0.0 0.0 0.0 0.0 3.8
```

```
# bold result
print(paste("bold result: ",round(boldresult[i],8)))
```

```
## [1] "bold result:  0.064"
```

The probability of winning with the bold strategy is 0.064 .

(c) best strategy

The Bold strategy provides a higher probability (0.064) of winning than the timid strategy (0.0203) .