stochastic process homework1

kallakuri radhakrishna

22/05/2020

example1

a

```
x <- runif(20,-0.8,1)
print(x)
```

```
##  [1]  0.10933130 -0.08717039  0.73054764  0.68541652 -0.56103550
##  [6]  0.51214425  0.72262463 -0.18390341  0.24713013  0.40824793
## [11] -0.32549207  0.28991100 -0.03110595 -0.21624835 -0.78209993
## [16]  0.26120498  0.66155322  0.82880968 -0.18431757  0.68270952
```

b

```
y <- 0
for (i in 1:20)
{
  y[i]<- 100+100*x[i]
  I <- as.numeric(x>=0.5)
  t <- sum(I)
}

y
```

```
##  [1] 110.93313  91.28296 173.05476 168.54165  43.89645 151.21443 172.26246
##  [8]  81.60966 124.71301 140.82479  67.45079 128.99110  96.88940  78.37516
## [15]  21.79001 126.12050 166.15532 182.88097  81.56824 168.27095
```

```
I
```

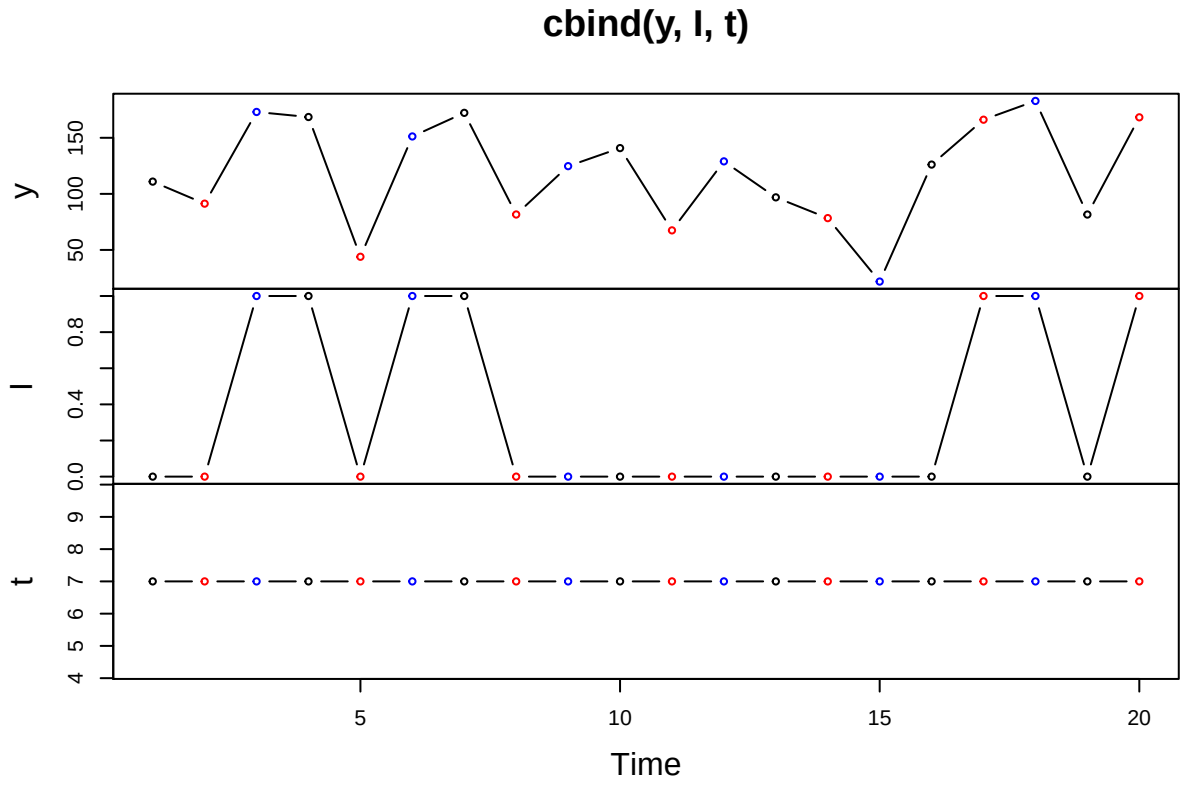
```
##  [1] 0 0 1 1 0 1 1 0 0 0 0 0 0 0 0 0 1 1 0 1
```

```
t
```

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

c

```
plot.ts(cbind(y,I,t), type="b", col=c("black","red","blue"), ylab="")
```



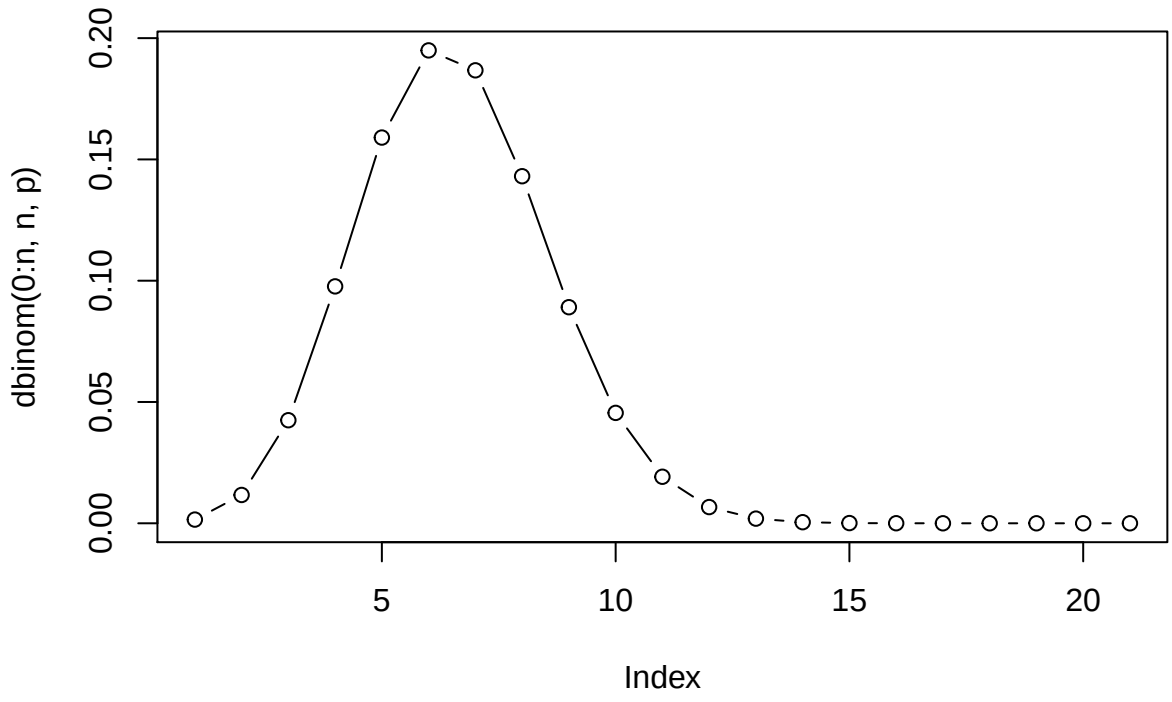
d

```
p <- 1-punif(0.5,-0.8,1)

p <- 0.277
n<- 20

Tn<- for(i in 1:20) {show(dbinom(i, n, p))}
```

```
## [1] 0.01167179
## [1] 0.04248176
## [1] 0.09765516
## [1] 0.1590104
## [1] 0.1949472
## [1] 0.1867233
## [1] 0.1430771
## [1] 0.08907685
## [1] 0.04550353
## [1] 0.01917694
## [1] 0.006679255
## [1] 0.001919246
## [1] 0.0004525003
## [1] 8.668228e-05
## [1] 1.328409e-05
## [1] 1.590462e-06
```

```
## [1] 1.433758e-07
## [1] 9.155162e-09
## [1] 3.692189e-10
## [1] 7.072865e-12
```

```
plot(dbinom(0:n, n, p), type = "b")
```




```
p <- 0.2777
In <-  sample(c(0,1), 20, replace = T, prob = c(1-p, p))
print(In)
```

```
##  [1] 1 0 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 1 0
```

e

```
# T(20)>5

p1 <- 1- pbinom(5,20,prob=0.2777)
cat("T(20)>5",p1)
```

```
## T(20)>5 0.4955403
```

```
#T(20)=T(18)

p2 <- sum(dbinom(c(18,20),20,prob=0.277))
cat("T(20)=T(18)",p2)
```

```
## T(20)=T(18) 9.162235e-09
```

f

```
x3 <- runif(300,-0.8,1)
y3 <- 0
for (i in 1:300)
{
  y3[i]<- 100+100*x3[i]

}
y3 # 300 realisations
```

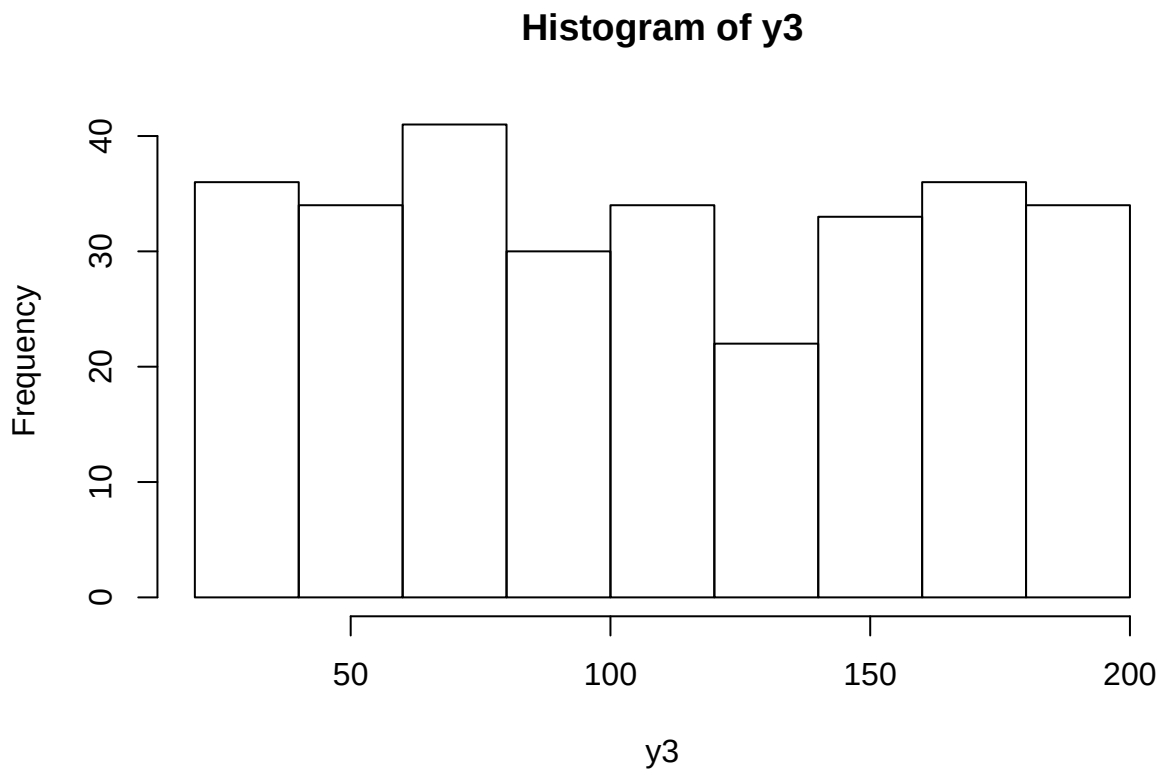
```
##   [1] 114.70850 117.61477 138.40153 123.26008  62.25036 149.98384  70.14784
##   [8]  21.82816 109.43619  82.71659 115.69082 112.90605  77.09984 116.06447
##  [15]  57.08918 142.35052  91.81612 168.19809 132.11397  80.08815 115.65911
##  [22]  38.05110  30.02285 199.96721 175.90313 182.52393 147.75894  23.39084
##  [29] 178.47960  62.34321 103.00636 190.73119  33.31379  93.38852 129.38718
##  [36] 138.33685 169.72714  93.06282  62.67329 163.21903 134.37136 148.75566
##  [43] 186.98501 114.55888 127.89575 120.05173  76.91434 168.47302  20.64068
##  [50]  34.74353  64.84875  40.06013  61.79017 143.28986  28.19248 174.98113
##  [57]  83.97775 115.03947  44.46672  31.17710 162.73702 136.26981 105.66491
##  [64] 164.13988 156.09425 142.48662  99.15727 183.84466  51.84900 159.60101
##  [71] 135.97459 169.71623  46.79545  75.48182 115.18794  59.90026 104.64715
##  [78] 177.36341  72.20846 140.17375 148.68002 109.45074 185.25369 190.48436
##  [85]  74.15597 169.52053 106.73033  29.75750 151.88509  69.53097 176.05744
##  [92] 142.91519 168.91886 108.78670  52.52962  31.13607 178.49322 102.12550
##  [99] 169.14213 102.45274 197.61430 173.88285  62.50623 172.03479  89.35727
## [106] 162.22717 187.71053 195.94115 151.07137  62.05335 155.49390  77.80484
## [113] 126.53283  79.86769  49.69400 147.32389  62.59832  78.26373 176.34759
## [120]  41.63390 163.76096 174.65954 175.86283  32.38609  22.45477 146.28098
## [127]  95.16797 102.45842  99.52790  82.32182 195.94800 185.43999  49.41921
## [134]  43.77799  40.31940 114.34338 152.25833  55.24838  95.95559  74.15801
## [141] 131.62728  43.13237 107.87356 141.42348 191.45049 168.07760  51.20745
## [148] 129.47401 184.92450 169.59160 187.99760  80.86728 198.94613 185.70747
## [155]  66.20728  52.80972 153.53227 101.10963  97.34118  75.95449  33.76247
## [162]  83.88796  30.22336 117.59297 124.39335 164.02998  68.43995  76.52731
## [169]  58.63414  69.49751 196.17341  58.46255 179.05390  89.83362 113.28379
## [176] 192.23450  35.54402 193.81467 194.16027 196.04684  34.89461 119.04246
## [183]  44.51493  40.89760 156.14105  99.66769 122.70835 125.00503 111.57940
## [190] 109.24247 102.05143 151.30657  33.20398 197.63610 146.75502  51.00296
## [197]  56.70634 152.11906  81.85898  67.43374 161.37391  44.93468  97.11547
## [204]  20.62701  21.44107  63.27342  27.88579  41.32946  65.75272  30.98768
## [211]  72.87529 190.10239  71.16364 143.79900  31.21277 165.91151 114.23971
## [218]  31.28887 129.80917 190.60750 189.96131 166.72354  96.22287 105.78206
## [225]  65.59914  53.46050 155.08000 182.77345  31.67007  30.71537 143.87991
## [232] 196.09203  82.03137  85.03721 193.39707 170.20447  41.23922  40.94323
## [239]  48.24719 163.21429 135.17504 189.55220  75.67231 120.87082  93.11247
## [246] 183.92536  27.17447  30.74341  55.47804  20.65293  76.38500  64.83741
## [253] 145.38595 183.58320  91.14251 114.31393 146.62179  99.61060 134.07625
## [260]  84.29846  35.02799 188.41123 172.97565  45.22732 178.72393 119.97930
## [267] 148.61370  48.14843  69.73420  50.44532 134.65602  80.48982 143.82203
## [274]  62.70277  85.84937  59.03619 144.49245 114.49729  22.51157 166.00615
```

```
## [281]  77.72070  75.38911 151.76985  36.37029  67.91517  98.10136  64.65931
## [288]  27.75320 122.23115  27.27438 169.16755  41.33317 187.41298  39.35593
## [295] 143.81386  61.31352 103.37186  62.92690  91.83136  22.70687
```

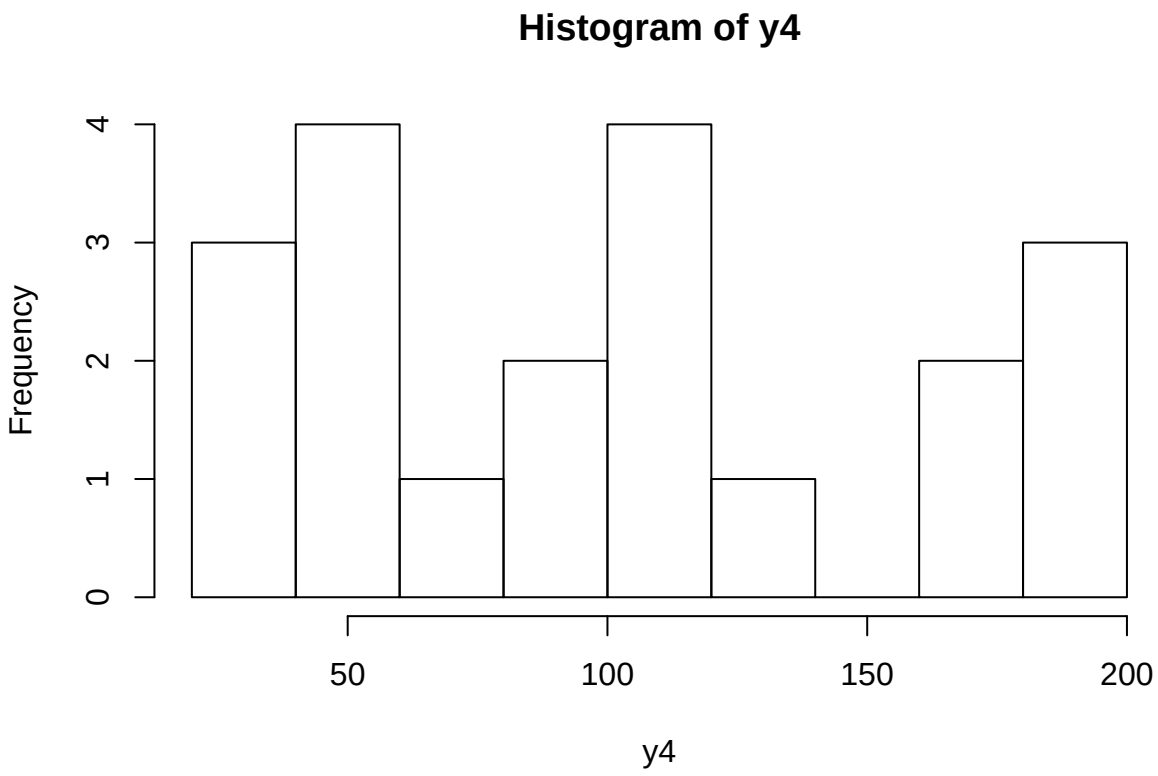
```
x4 <- runif(20,-0.8,1)
y4 <- 0
for (i in 1:20)
{
  y4[i]<- 100+100*x4[i]

}
y4 # values of Y(20) stored in a vector
```

```
##  [1]  31.79877 118.09550  80.44595 193.20379  32.24327 120.35944  50.79902
##  [8]  34.59884  57.92668 185.16651  95.04495  52.57331 110.46170 110.56100
## [15] 194.28169 165.54971  45.72907 106.23611 171.47156  66.95997
```

```
hist(y3)
```




```
hist(y4)
```



g

```
# expectation of y4 = 100*mean(x4)+100
# this is expecation of Y20
E <- 100*mean(x4)+ 100
cat("expecation of Y20",E)
```

```
## expecation of Y20 101.1753
```

```
# below are variance of Y20 whose values were stored in y4
e1 <- var(y4)
cat("variance of Y20",e1)
```

```
## variance of Y20 3137.486
```

h

```
e2 <- mean(y4)
e2
```

```
## [1] 101.1753
```

```
e3 <- sd(y4)
e3
```

```
## [1] 56.01327
```

```r
# Y(20) has mean 111.24 and sd 38.791 its probability to be less than 100 is
p3 <- pnorm(100,96.98,sd=50.52)
cat("probability that (Y(20)<= 100)",p3)
```

```
## probability that (Y(20)<= 100) 0.5238339
```

example 2

a

```r
#states
states <- c("sleeping","spinning","eating")

#transition matrix

tmat <- matrix(data=c(.7,.1,.2,
                      .2,.4,.4,
                      .3,.4,.3), byrow=TRUE, nrow=3,
                      dimnames=list(states,states))




tmat
```

```
##          sleeping spinning eating
## sleeping      0.7      0.1    0.2
## spinning      0.2      0.4    0.4
## eating        0.3      0.4    0.3
```

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

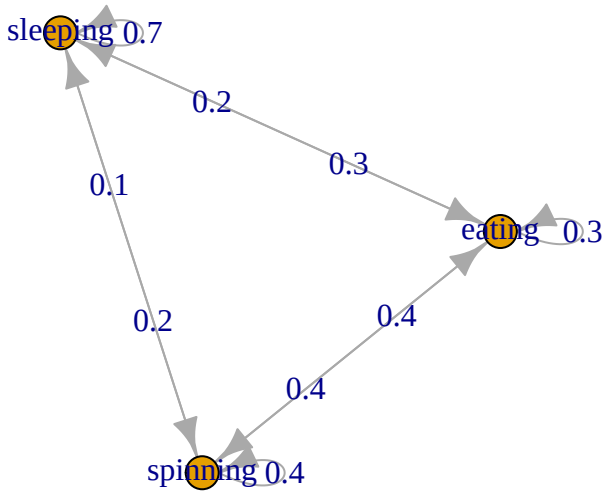
```
## Warning: package 'markovchain' was built under R version 3.6.3
```

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

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

b

```r
# transition diagram
plot(mc)
```



c

```
# simulating the hamsters activity for next 60 mins
simulation <- rmarkovchain(n=6,object=mc,t0="spinning")
simulation
```

```
## [1] "sleeping" "sleeping" "sleeping" "eating"   "spinning" "eating"
```

d

```
# transition eating -> sleeping after 60 mins
(mc^6)["eating","sleeping"]
```

```
## [1] 0.452614
```

e

```
# transition after 20/60 mins if it is currently sleeping

(mc^2)["sleeping",]
```

```
## sleeping spinning   eating
##     0.57     0.19     0.24
```

```
(mc^6)["sleeping",]
```

```
## sleeping spinning   eating
## 0.461441 0.259735 0.278824
```

```
# another method
indist <- c(1,0,0)
#after 20 mins
indist*(mc^2)
```

```
##      sleeping spinning eating
## [1,]     0.57     0.19   0.24
```

```
# after 60 mins
indist*(mc^6)
```

```
##      sleeping spinning   eating
## [1,] 0.461441 0.259735 0.278824
```

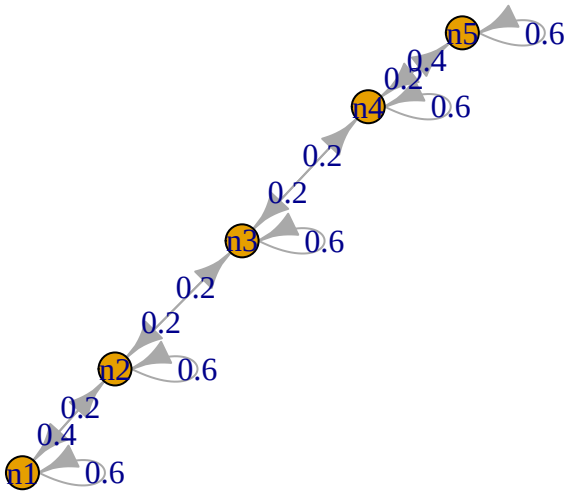
example 3

a

```
#states
states1 <- c("n1","n2","n3","n4","n5")

#transition matrix

tmat1 <- matrix(data=c(.6,.4,0,0,0,
                       .2,.6,.2,0,0,
                       0,.2,.6,.2,0,
                       0,0,.2,.6,.2,
                       0,0,0,.4,.6
                       ), byrow=TRUE, nrow=5,
                dimnames=list(states1,states1))

tmat1
```

```
##     n1  n2  n3  n4  n5
## n1 0.6 0.4 0.0 0.0 0.0
## n2 0.2 0.6 0.2 0.0 0.0
## n3 0.0 0.2 0.6 0.2 0.0
## n4 0.0 0.0 0.2 0.6 0.2
## n5 0.0 0.0 0.0 0.4 0.6
```

```
library("markovchain")
# Markov chain
mc1 <- new("markovchain", states = states1, byrow = TRUE, transitionMatrix = tmat1, name = "process")


# transition diagram
plot(mc1)
```



b

```
# 10 steps

simulation1 <- rmarkovchain(n=10,object=mc1,t0="n1")
simulation1
```

```
##  [1] "n1" "n1" "n1" "n2" "n3" "n4" "n4" "n3" "n4" "n4"
```

c

```
# transition n1 -> n1 after 5 steps
(mc1^5)["n1","n1"]
```

```
## [1] 0.27936
```

d

```
indist1 <- c(1,0,0,0,0)
#after n steps, n changes from 1 to 10
for (i in 1:10) {show((indist1*mc1^i))}
```

```
##       n1  n2 n3 n4 n5
## [1,] 0.6 0.4  0  0  0
```

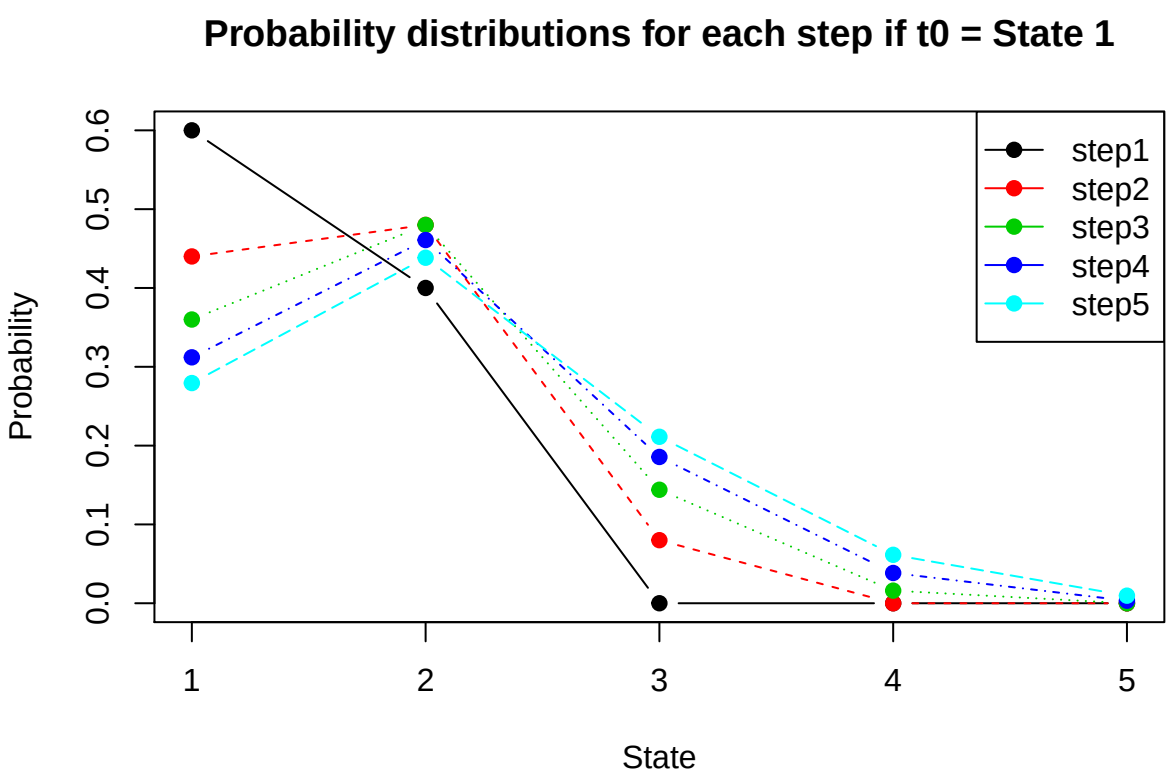
```
##        n1   n2   n3 n4 n5
## [1,] 0.44 0.48 0.08  0  0
##        n1   n2    n3    n4 n5
## [1,] 0.36 0.48 0.144 0.016  0
##         n1     n2     n3     n4     n5
## [1,] 0.312 0.4608 0.1856 0.0384 0.0032
##           n1     n2     n3      n4     n5
## [1,] 0.27936 0.4384 0.2112 0.06144 0.0096
##            n1       n2       n3       n4       n5
## [1,] 0.255296 0.417024 0.226688 0.082944 0.018048
##             n1        n2        n3        n4        n5
## [1,] 0.2365824 0.3976704 0.2360064 0.1023232 0.0274176
##             n1        n2        n3        n4        n5
## [1,] 0.2214835 0.3804365 0.2416026 0.1195622 0.0369152
##             n1        n2        n3        n4         n5
## [1,] 0.2089774 0.3651758 0.2449613 0.1348239 0.04606157
##             n1        n2        n3        n4         n5
## [1,] 0.1984216 0.3516887 0.2469767 0.1483112 0.05460173
```

e

```
prob_mat <-matrix(
     c(0.6,0.4,0,0,0, 0.44,0.48,0.08,0,0,0.36,0.48,0.144,0.016,0,0.312,0.4608,0.1856,0.0384,0.0032,0.279
prob_mat
```

```
##         [,1]   [,2]   [,3]    [,4]   [,5]
## [1,] 0.60000 0.4000 0.0000 0.00000 0.0000
## [2,] 0.44000 0.4800 0.0800 0.00000 0.0000
## [3,] 0.36000 0.4800 0.1440 0.01600 0.0000
## [4,] 0.31200 0.4608 0.1856 0.03840 0.0032
## [5,] 0.27936 0.4384 0.2112 0.06144 0.0096
```

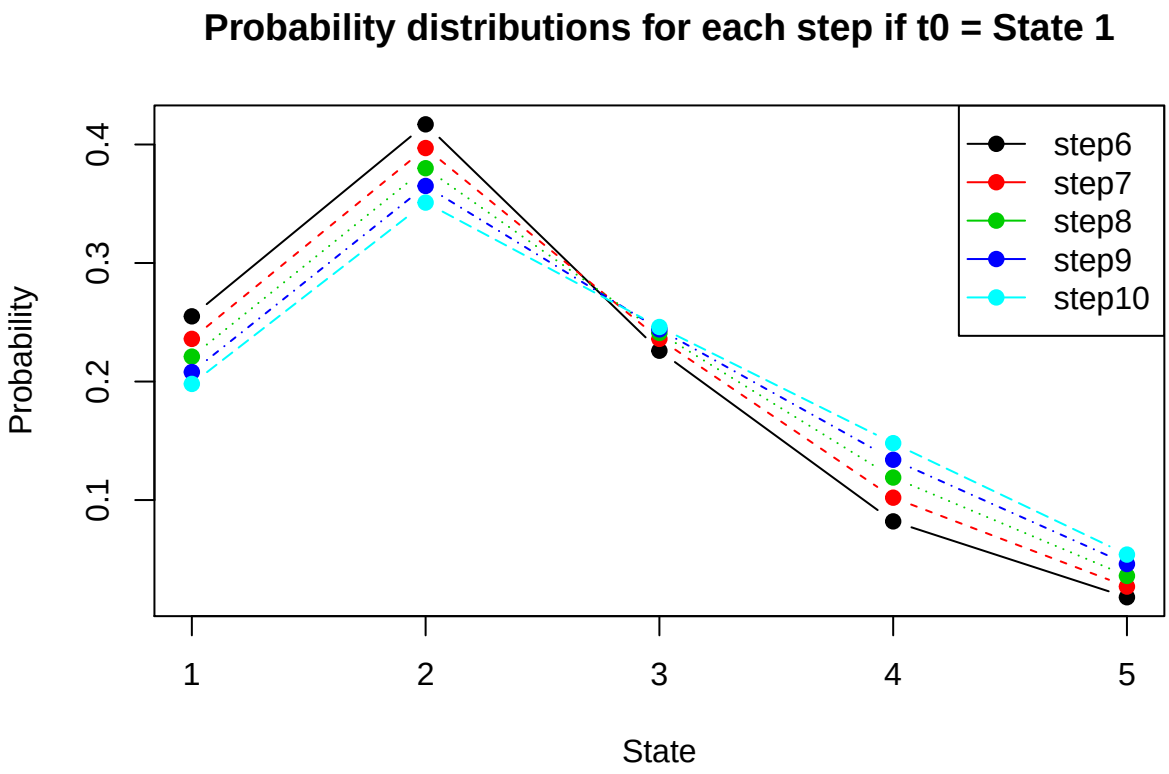
```
steps <- c("step1","step2","step3","step4","step5")
matplot(t(prob_mat[]), type = c("b"), ylab = "Probability", xlab = "State", main = "Probability distribu
legend("topright", legend = steps[1:5], col = 1:5, lty=1, cex=1, pch = 19)
```




```
prob_mat1 <- matrix(c(0.255,0.417,0.226,0.082,0.018, 0.236,0.397,0.236,0.102,0.027,0.221,0.380,0.241,0.1
prob_mat1
```

```
##       [,1]  [,2]  [,3]  [,4]  [,5]
## [1,] 0.255 0.417 0.226 0.082 0.018
## [2,] 0.236 0.397 0.236 0.102 0.027
## [3,] 0.221 0.380 0.241 0.119 0.036
## [4,] 0.208 0.365 0.244 0.134 0.046
## [5,] 0.198 0.351 0.246 0.148 0.054
```

```
steps1 <- c("step6","step7","step8","step9","step10")
matplot(t(prob_mat1[]), type = c("b"), ylab = "Probability", xlab = "State", main = "Probability distril
legend("topright", legend = steps1[1:5], col = 1:5, lty=1, cex=1, pch = 19)
```


Probability distributions for each step if t0 = State 1
Probability
0.1
0.2
0.3
0.4
1
2
3
4
5
State
step6
step7
step8
step9
step10