Homework set 2 - due 02/04/2022

Math 495 – Renato Feres

Simulating a Markov chain in R

This is a quick tutorial with examples for simulating sample sequences of Markov chains using R. The basic information about R that you need here is provided by the R tutorial from HW 1. Make sure that the code makes sense to you. You may need to use some of it in the homework itself.

- **Generating sample sequences of a finite state Markov chain.** The following is a simple program for generating sample sequences of a Markov chain. It consists of a function that takes in three arguments: N (number of steps), π_0 (the initial probability distribution), P (the transition matrix), and produces a vector of length N: $(X_1,\ldots,X_N)$ where each X_i lies in the state space $\{1,2,\ldots,s\}$. Note that π_0 must be a vector of length s and P must be an s-by-s matrix. Here is the program:

```
#The set of states is {1,2,...,s}
#where s is the length of pi0 and P is s-by-s.
#pi0 is the probability distribution of the initial step
#and P is the transition matrix.
Markov=function(N,pi0,P) {
  #Number of states
  s=length(pi0)
  X=matrix(0,nrow=1,ncol=N)           #Initialize matrix to record chain values.
  a=sample(c(1:s),1,replace=TRUE,pi0) #Sample from 1,...,s with distribution pi0.
  X[1]=a                              #This sets the initial value of the chain.
  for (i in 2:N) {
    a=sample(c(1:s),1,replace=TRUE,P[a,]) #Next value of chain has distribution P[a,].
    X[i]=a
  }
  X #Output of the function Markov is the sequence (chain) of length N.
}
```

To test that it works as expected, let us do the following experiment. Let $\pi_0 = (1,0)$ and

$$P = \begin{pmatrix} 0.8 & 0.2 \\ 0.3 & 0.7 \end{pmatrix}.$$

We generate $N = 10^5$ steps of the sequence: `X=Markov(100000,pi0,P)` and compute the relative frequency of occurrences of state 1 (the set of states being $\{1,2\}$.) It is not difficult to compute the exact stationary distribution: $\pi = (0.6, 0.4)$. Let us compute the approximate stationary probability of 1 using the Markov chain simulation:

```
#Define the probability distribution of X0
pi0=c(1,0)
#Define the transition probabilities matrix
P=matrix(0,nrow=2,ncol=2)
P[1,]=c(0.8,0.2)
P[2,]=c(0.3,0.7)
#Number of steps
N=10^5
> X=Markov(N,pi0,P)
> sum(X==1)/N
[1] 0.60168
```

The result is sufficiently close to 0.6 to make me believe that the program has no bugs.

Here is a variant of the above program. We wish to find the number of steps, beginning at state i, till the chain with transition matrix P reaches a different state j. We count step 0 but not the step at which the chain is at j.

```
#This program gives the number of steps in one
#run of the Markov chain with transition probabilities
#Matrix P, starting at i, till it reaches j. We count
#the 0th step but not the step when the chain is at j.
#Be careful: if the probability of eventually reaching j from
#i is 0, the program will not stop! (We say in such case that i and j do
#not communicate.)
Number_of_steps=function(i,j,P) {
  T=0      #Initialize time (or number of steps) as 0.
  u=dim(P) #dim(P) is (s,s) if s is the number of states.
  s=u[1]
  X=i.     #i is the initial state.
  while (X!=j){
    X=sample(c(1:s),1,replace=TRUE,P[X,])
    T=T+1
  }
  return(T)
}
```

Suppose we wish to find the expected number of steps it takes to get to state 2 starting at 1 for the two-state chain with transition matrix P given above. We can do the following:

```
P=matrix(0,2,2)
P[1,]=c(0.8,0.2)
P[2,]=c(0.3,0.7)
N=1000
T=matrix(0,1,N) #T will record the number of steps for each trial of the experiment.
```

```
for (n in 1:N){
  T[n]=Number_of_steps(1,2,P)
}
sum(T)/N
```

Note that we are approximating the expected time by the mean number of steps of N sample values. This can be justified by the law of large numbers. One run of this program, for $N = 1000$, gave me the value 5.07. We will see later (soon!) in this course that the exact number is 5.

Let us do one more experiment based on the chain of the first example above. Here it will be convenient to label the states as 0 or 1 (instead of 1 or 2), so the state set is $S = \{0, 1\}$. We are interested in the frequency over the long run of patterns of 4 consecutive symbols: 0010, 1001, etc. Each pattern corresponds to an integer expressed in base 2. So, for example, $0110 \mapsto 6$, $1101 \mapsto 13$, and so forth. In general, the correspondence between patterns and integers is given by

$$a_0 a_1 a_2 a_3 \mapsto a_0 2^0 + a_1 2^1 + a_2 w^2 + a_3 2^3.$$

This naturally defines another Markov chain with 16 states and state transitions

$$a_i a_{i+1} a_{i+2} a_{i+3} \mapsto a_{i+1} a_{i+2} a_{i+3} a_{i+4}.$$

```
#Define the probability distribution of X0
pi0=c(1,0)
#Define the transition probabilities matrix
P=matrix(0,nrow=2,ncol=2)
P[1,]=c(0.8,0.2)
P[2,]=c(0.3,0.7)
#Number of steps
N=10^5
X=Markov(N,pi0,P)
sum(X==1)/N
#Subtract 1 from each entry of X so that states are 0 and 1
X=X-1
#Initialize an array Y. It will have integer entries from 0 to 15 that label the
#16 patterns of four digits from {0,1}.
#Set initially Y to be all zeros:
Y=matrix(0,1,N-3)
#Now for each index i associate the integer from 0 to 15 corresponding
#to the binary string of 4 symbols given by X[i] ... X[i+3]
for (i in 1:(N-3)) {
  Y[i]=X[i]*1+X[i+1]*2+X[i+2]*4+X[i+3]*8
}
#We can get an idea of the relative frequencies of the various binary p
#patterns by plotting a histogram:
hist(Y,freq=FALSE,breaks=seq(from=-0.5,to=15.5,by=1))
#We can also get the relative frequencies of the 16 patterns directly:
rel_freq = matrix(0,1,16)
for (k in 0:15) {
```

```
  rel_freq[k+1]=sum(Y==k)/(N-3)
}
#The same vector of relative frequencies could have been obtained
#from the histogram as follows:
h=hist(Y,freq=FALSE,breaks=seq(from=-0.5,to=15.5,by=1))
rel_freq_from_hist = h$density
```

The histogram is shown next.

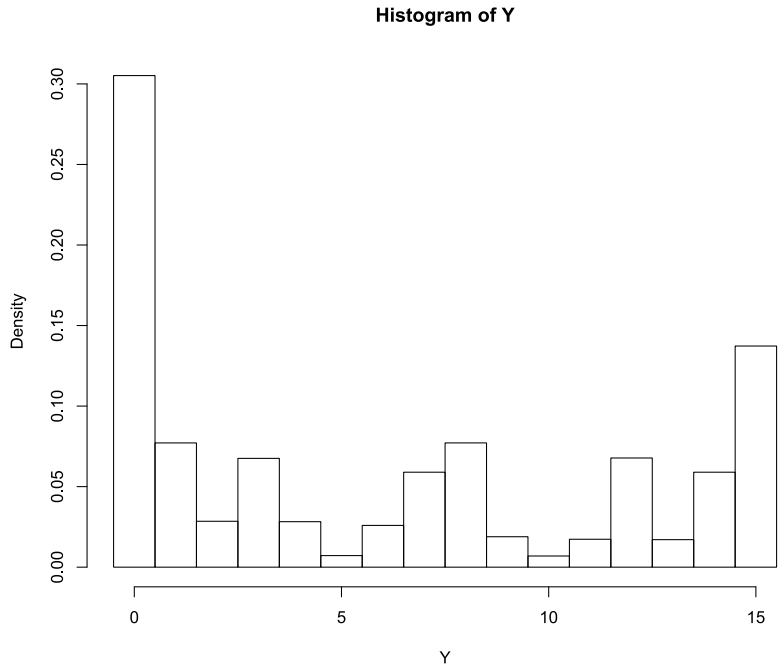

Figure 1: Histogram of relative frequencies of patterns of 4 symbols for a two-state Markov chain.

Problems

As a general advice for all our HW assignments, work on all the exercises, even those you will not be asked to turn in for grading. Exams may contain any of these (except the parts that are specifically about R). The ones you'll be asked to turn in are: Exercises 3 and 4.

1. (Textbook, Exercise 2.2, Page 70). Let $X_0, X_1, \ldots$ be a Markov chain with states $\{1,2,3\}$ and transition matrix

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{pmatrix} 0 & 1/2 & 1/2 \\ 1 & 0 & 0 \\ 1/3 & 1/3 & 1/3 \end{pmatrix} \end{array}$$

   (later on, I'll generally omit the labels on the margin of rows and columns as indicated in this matrix) and initial distribution $\alpha = (1/2, 0, 1/2)$. Find the following:

   (a) $P(X_2 = 1 | X_1 = 3)$

   (b) $P(X_1 = 3, X_2 = 1)$

   (c) $P(X_1 = 3 | X_2 = 1)$

   (d) $P(X_9 = 1 | X_1 = 3, X_4 = 1, X_7 = 2)$

2. (Textbook, Exercise 2.5, Page 71). Consider a random walk on $\{0, 1, \ldots, k\}$, which moves left and right with respective probabilities q and p. If the walk is at 0 it transitions to 1 on the next step. If the walk is at k it transitions to $k-1$ on the next step. This is called *random walk with reflecting boundaries*. Assume that $k = 3$, $q = 1/4$, $p = 3/4$, and the initial distribution is uniform. You may use R (or any other program) to obtain the required powers of the transition matrix.

   (a) Exhibit the transition matrix.

   (b) Find $P(X_7 = 1 | X_0 = 3, X_2 = 2, X_4 = 2)$

   (c) Find $P(X_3 = 1, X_5 = 3)$

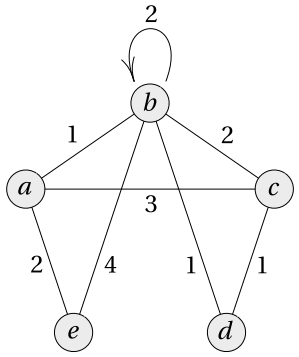

Figure 2: Graph of Exercise 3.

3. (Textbook, Exercise 2.8, Page 71). Give the Markov transition matrix for the random walk on the weighted graph in Figure 2. (Keep the entries as fractions rather than in decimal form.) I suggest redrawing the graph as a

Markov transition graph (like those of Figure 2.5, page 51 of the textbook). Note that each line in the above graph corresponds to two arrows (in opposite directions) in the transition graph.

4. (Snakes and Ladders, simplified. See the full size on page 121 of the textbook. This problem is to be done by computer.) I assume for simplicity that you are the only player in this board game. Start at square 1 and move at each step to one, two, or three squares ahead according to the outcome of spinning a roulette. If you land at the bottom of a ladder, immediate climb it. If you land on the head of a snake, slide down to the tip of the snake's tail. The game ends at reaching square 12. The only way to reach 12 from squares 10 and 11 is by drawing 2 or 1, respectively; if the roulette gives a higher number than necessary, stay where you are. We wish to run a stochastic simulation of this game by using the Markov chain program. (See the examples in the tutorial part of this assignment. You should be able to modify them for this problem.)

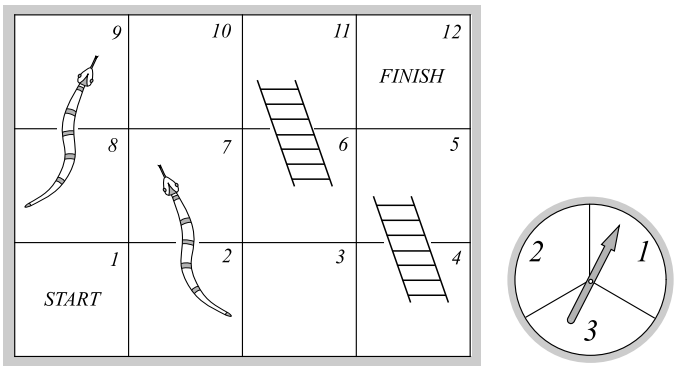

Figure 3: The snakes-and-ladders game.

(a) What is the expected number of time steps to go from 1 to 12? What is its standard deviation?

(b) Draw a histogram of the distribution of the number of steps. (Be careful with the breaks!)

Before writing your program, convince yourself that my transition probabilities diagram given below correctly interprets the rules of the game. The simulation consists of running the game till finish a large number of times and obtaining the statistics asked in the problem.

Remarks about this problem: First note that our original program `Markov` needs some changes. In that program we have a deterministic stopping time N, but now this time is random and we cannot tell in advance what it is. The following modifications should take care of this problem. The input variable `TerminalStates` is an array giving the states at which the chain should stop. In this example we have the single element array `c(8)` since 8 is the index of state 12. (The mathematical chain simply goes on forever inside this terminal set; for the snakes and ladders example, a sample path of the chain could be

$$1\ 3\ 5\ 8\ 10\ 10\ 11\ 12\ 12\ 12\ \cdots$$

but in the program we wish to interrupt it at the first occurrence of 12.)

```
################################
#We modify the first Markov program to allow
#for a set of terminal states, at which the
```

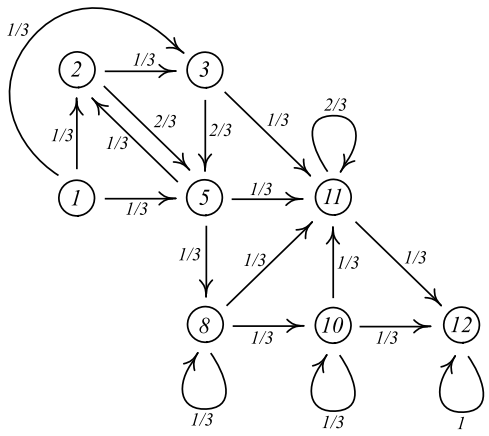

Figure 4: Markov transitions diagram for the snakes-and-ladders game.

```
#chain must stop. Let TerminalStates be the array of
#such states.
Markov_stopped=function(pi0,P,TerminalStates) {
  #Number of states
  s=length(pi0)
  X=matrix(0,1,1000)
  a=sample(c(1:s),1,replace=TRUE,pi0)
  X[1]=a
  i=1
  while (X[i] %in% TerminalStates == FALSE) {
    a=sample(c(1:s),1,replace=TRUE,P[a,])
    i=i+1
    X[i]=a
  }
  U=X[1:i]
  return(U)
}
################################
```

Note that 4, 6, 7, 9 are square numbers but not states. For example, landing at square 4 immediately takes you to square 5. So the set of states is $\{1,2,3,5,8,10,11,12\}$. Because the states and their indices do not coincide, it could be helpful to translate the output of the previous program (a string of indices) into a string of actual states. This is a minor point, but the following script does the translation. See if you can make sense of the syntax. (Never assume that my coding solutions are the best or the most efficient! You may find a better way to do the same thing.)

```
################################
#This function associates to each of
#1, 2, 3, 4, 5, 6, 7, 8 the corresponding states:
#1, 2, 3, 5, 8, 10, 11, 12
Substitute_state = function(x) {
```

```
  a1 = which(x==4)
  a2 = which(x==5)
  a3 = which(x==6)
  a4 = which(x==7)
  a5 = which(x==8)
  u  = x
  u[a1] = 5
  u[a2] = 8
  u[a3] = 10
  u[a4] = 11
  u[a5] = 12
  return(u)
}
################################
```

To test the program, here's what one trial run would look like (you'll need to define

```
pi0=c(1, 0, 0, 0, 0, 0, 0, 0)
```

and P; the latter is the matrix of transition probabilities.)

```
> Substitute_state(Markov_stopped(pi0,P,c(8)))
[1]  1  3  5  8 10 10 11 12
```

Having a program to simulated the Markov chain with a stopping time, we can now answer the questions posed. For example, for part (a), run the chain from state 1 till it stops, a large number times (say, $n = 10000$). Collect the number of steps in each run into a vector U then find `mean(U)` and `sd(U)`.