Markov Chain Monte Carlo

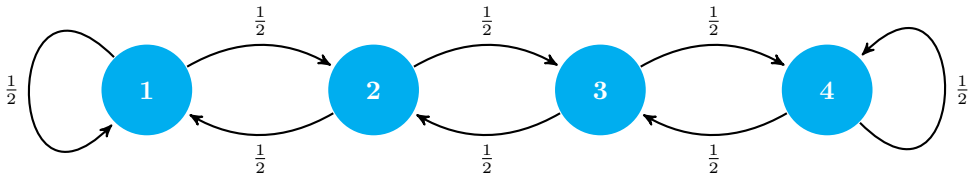
Prof. Richey and Prof. Wright

April 27, 2018

Introduction

We will build a Markov chain on a set of $n = 4$ states with a given steady-state distribution. Imagine the states lined up horizontally in a row, as in the diagram below. Our goal is achieve a steady-state distribution defined by the frequencies $f(i) = i$, for $i = 1, 2, 3, 4$.

To start, define a simple proposal transition function Q as given by the arrows in the diagram:



We can do this

Use the code from last time to build the Markov chain transition Matrix T with steady-state vector p_{ss} proportional to $[f(1), f(2), f(3), f(4)]$. First, make a proposal transition matrix Q, and then modify it to obtain T.

Agents again

Imagine an agent A located at any state j. Our method for constructing T provides a way to put A into motion.

At state j, use Q to select a site i to *consider* moving to. In other words, select i with probability $Q_{i,j}$.

- If $p_{ss}[j] \leq p_{ss}[i]$, then transition $j \to i$.
- If $p_{ss}[j] > p_{ss}[i]$, then transition $j \to i$ **with probability** ρ, where ρ is defined by

$$\rho = \frac{p_{ss}[i]}{p_{ss}[j]}.$$

Computational note

Here a couple thoughts on how to do some of this computationally.

Selecting the Propoal Transition: You could, of course, use the proposal transition matrix Q to do this. However, this approach doesn't scale well. Imagine building the $n \times n$ matrix Q when the number of states is $n = 10000$ or so.

Instead, let's take advantage of the simple transition rule. Suppose the agent is in state j.

- If the state j is "interior", i.e., $1 < j < n$, then, with equal probability the agent transitions to a new state $i = j + \Delta$ where $\Delta = \pm 1$.
- If the agent is in state j at the border, i.e., $j = 1$ or $j = n$, then the agent either stays put or moves to the only adjacent state with equal probability. Thus, $i = j + \Delta$ where:
  - If $j = 1$, then $\Delta \in \{0, 1\}$.
  - If $j = n$, then $\Delta \in \{-1, 0\}$.

Hence, to select the proposal transition, we just need to select a value of Δ for a set of two values defined by the state. Specifically, given a current state, `currState`, you can create the proposed next state, `propState`, via:

```
n <- 4
currState <- 1
if(1 < currState && currState < n){
  Delta <- c(-1,1)
}else if(currState == 1){
  Delta <- c(0,1)
}else{
  Delta <- c(-1,0)
}
propState <- currState + sample(Delta, 1)
```

Do something with probability ρ: This is simple. If you have a probability $\rho \in [0, 1]$, then you can decide (True or False) to do something with this probability pretty easily using the R `sample` function

```
rho <- .55
sample(c(TRUE,FALSE), 1, prob=c(rho,1-rho))
```

```
## [1] TRUE
```

For example, to do something repeatedly with probability ρ:

```
for(i in 1:10){
  print(sample(c(TRUE,FALSE), 1, prob=c(rho,1-rho)))
}
```

```
## [1] TRUE
## [1] FALSE
## [1] TRUE
## [1] TRUE
## [1] TRUE
## [1] TRUE
## [1] FALSE
## [1] TRUE
## [1] TRUE
## [1] TRUE
```

Simple example.

Suppose $a = 1$ and $b = 1$. With probability $\rho = a/(a + b)$, let $a = a + 1$. Otherwise, let $b = b + 1$. What happens to the ratio a/b if you do this, say, 10000 times?

```
a <- 1
b <- 1
M <- 10000
for(m in 1:M){
  rho <- a/(a+b)
  doIt <- sample(c(TRUE,FALSE), 1, prob=c(rho,1-rho))
```

```
  if(doIt){
    a <- a + 1
  }else{
    b <- b + 1
  }
}
c(a, b, a/(a+b))
```

```
## [1] 2815.0000000 7187.0000000    0.2814437
```

Interesting. What would happen if you repeat this experiment a large number of times a keep track of the various values of the final ratio $a/(a+b)$?

There are other ways to make a decision based on a probability ρ using the built-in random number generators in R. For example, the following code uses the `runif` (random uniform) function.

```
rho <- .55
runif(1) < rho
```

```
## [1] FALSE
```

This is simpler than what we had previously.

Back to Markov chains

Exercise: With our simple 4-state Markov chain, start an agent A in any state (state 1 is fine). Use the transition rules defined above to send A off on a long journey. Keep track of where the agent goes, say in a vector called `statesVisited` . Does the agent visit the states according to the steady state frequencies?

Starter code: Setup the scenario.

```
n <- 4
f <- function(i){ i }
currState <- 1
```

Now we implement a move for the current state to a new state. First the proposed state.

```
currState <- 1
if(1 < currState && currState < n){
  Delta <- c(-1,1)
}else if(currState == 1){
  Delta <- c(0,1)
}else{
  Delta <- c(-1,0)
}
propState <- currState + sample(Delta, 1)
```

Now the decision based on the probabilities.

```
currProb <- f(currState)
propProb <- f(propState)
if(currProb <= propProb){
  currState <- propState
}else{
   rho <- propProb/currProb
   doMove <- sample(c(TRUE,FALSE), 1, prob=c(rho,1-rho))
   #doMove <- runif(1)<rho #alternate method
   if(doMove){
```

```
    currState <- propState
  }else{
     # nothing to do
  }
}
currState
```

```
## [1] 2
```

This chunk of code moves a currState to a (potentially the same) new currState. The plan is to repeat this process a large number of times and keep track of the currStates that are visited.

```
x <- 2
```

Make the process a function that starts with a state an moves to a new state.

```
doMove <- function(currState){
  ## Get the proposed state
  if(1 < currState && currState < n){
    Delta <- c(-1,1)
  }else if(currState == 1){
    Delta <- c(0,1)
  }else{
     Delta <- c(-1,0)
  }
  #print(Delta )
  propState <- currState + sample(Delta, 1)

  ## get the probabilities
  currProb <- f(currState)
  propProb <- f(propState)
  ## make the decision to move
  if(currProb <= propProb){
    # easy case
    currState <- propState
  }else{
   # decide based on the ratio
   rho <- propProb/currProb
   makeMove <- sample(c(TRUE,FALSE), 1, prob=c(rho,1-rho))
   if(makeMove){
     currState <- propState
    }
  }
  currState
}
```

```
doMove(1)
```

```
## [1] 2
```

Let it run for a while...

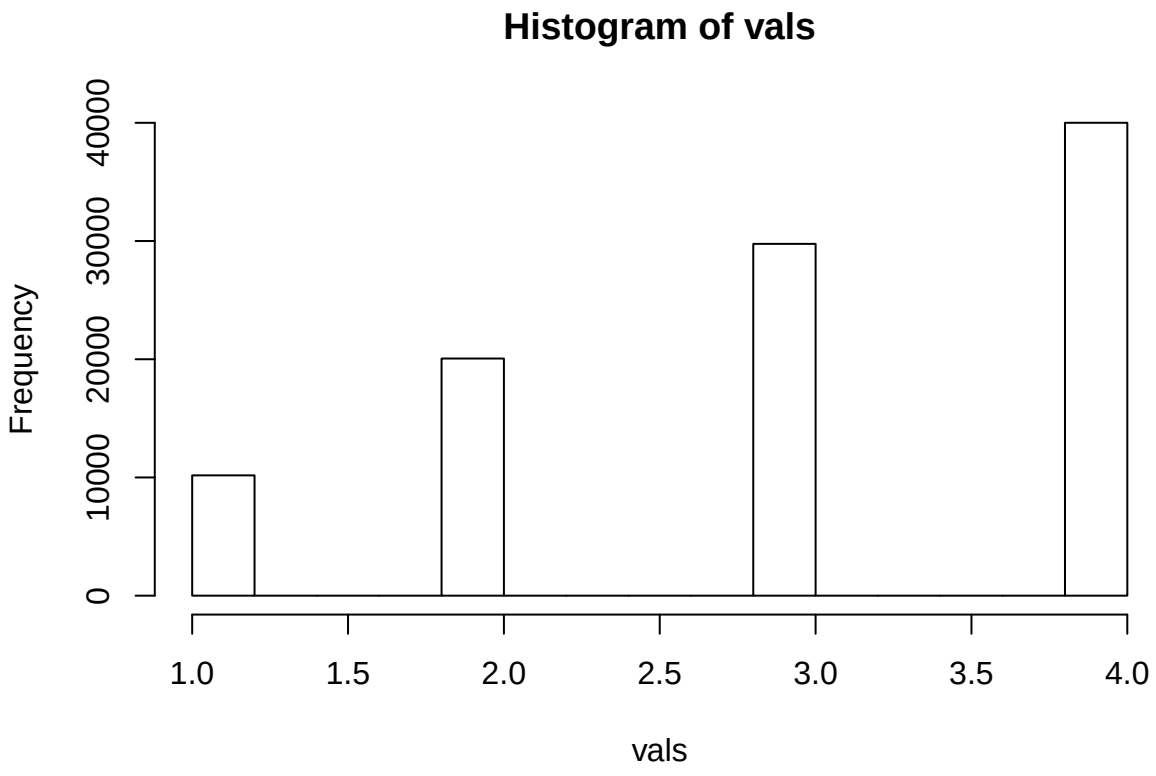
```
M <- 100000
vals <- matrix(nrow=M)
currState <- 1
for(m in 1:M){
  vals[m] <- currState
  currState <- doMove(currState)
```

```
}
table(vals)/M
```

```
## vals
##       1       2       3       4
## 0.10176 0.20062 0.29759 0.40003
```

This looks good, the values should be roughly in the same proportion as $(f(1), f*2), f(3), f(4) = (1, 2, 3, 4)$. Also, take a look at a histogram of the states visited (use the R command `hist(statesVisited)`). Since there are only 4 states, this is a pretty boring histogram.

```
hist(vals)
```



Assignment

Now imagine the interval $[0, 1]$ divided into $N + 1$ evenly spaced states:

$$0, \frac{1}{N}, \frac{2}{N}, \ldots, \frac{N}{N} = 1$$

Suppose that N is quite large, e.g., $N = 1000$ is plausible. For the desired steady-state distribution, we will use the very simple function $f(x) = x$. Note, once again this just gives the relative frequencies, not the probabilities.

Define a proposal transition in a manner similar to what we did above, namely if j is not 0 or 1, then

$$Q[j \to i] = \begin{cases} \frac{1}{2} & i = j \pm \frac{1}{N}, \\ 0 & \text{otherwise.} \end{cases}$$

If $j = 0$, then $Q[0 \to 0] = Q\left[0 \to \frac{1}{N}\right] = \frac{1}{2}$, and 0 otherwise.

If $j = 1$, then $Q[1 \to 1] = Q\left[1 \to \frac{N-1}{N}\right] = \frac{1}{2}$, and 0 otherwise.

You could use this set up to create the $N \times N$ transition matrix T. For large values of N, this is formidable and might use too much memory in the computer. Instead, we will use an agent to travel around the states.

Problem 1

Set this up and run an agent-based simulation. Use at least $N = 1000$ (you will probability want to use a smaller value to get things working). Let the agent run around the states for a while (100,000 or more steps). Keep track of where the agent goes. Do you see the steady-state distribution emerge in the samples? What does a histogram (use `hist()`) look like?

Problem 2

Change the interval to $(-1, 1)$ and chop it up into N evenly spaced points (states). To define the probability weights, use

$$f(x) = e^{-20x^2}$$

Generate at least 100,000 steps of an agent trail and keep track of the results. What is the steady-state distribution? What does the histogram look like?

Handing in your work

Submit your solutions (code and histograms) to the MCMC assignment on Moodle. Your work should be an HTML or PDF file produced using R Markdown.

Please delete the examples and discussion prior to the assignments from the file that you turn in. This is a minor assignment—it is not necessary to provide a lot of discussion about your answers.

This assignment is due on **Wednesday, May 2**.