
5

MARKOV CHAIN MONTE CARLO

An algorithm must be seen to be believed.

—Donald Knuth

5.1 INTRODUCTION

How to simulate from complex and high-dimensional probability distributions is a fundamental problem in science, statistics, and numerous applied fields. Markov chain Monte Carlo (MCMC) is a remarkable methodology, which utilizes Markov sequences to effectively simulate from what would otherwise be intractable distributions. MCMC has been described as a “revolution” in applied mathematics, a “paradigm shift” for the field of statistics, and one of the “ten most important algorithms” of the 20th century. See Diaconis (2009), Robert and Casella (2011), and Dongarra and Sullivan (2000).

Given a probability distribution π , the goal of MCMC is to simulate a random variable X whose distribution is π . The distribution may be continuous or discrete, although we assume at the beginning that π is discrete. Often, one wants to estimate an expectation or other function of a joint distribution from a high-dimensional space.

The MCMC algorithm constructs an ergodic Markov chain whose limiting distribution is the desired π . One then runs the chain long enough for the chain to converge,

or nearly converge, to its limiting distribution, and outputs the final element or elements of the Markov sequence as a sample from π .

MCMC relies on the fact that the limiting properties of ergodic Markov chains have some similarities to independent and identically distributed sequences. In particular, the strong law of large numbers holds.

Law of Large Numbers

The law of large numbers is one of the fundamental limit theorems of probability. If Y_1, Y_2, \dots is an i.i.d. sequence with common mean $\mu < \infty$, then the strong law of large numbers says that, with probability 1,

$$\lim_{n \rightarrow \infty} \frac{Y_1 + \dots + Y_n}{n} = \mu.$$

Equivalently, let Y be a random variable with the same distribution as the Y_i and assume that r is a bounded, real-valued function. Then, $r(Y_1), r(Y_2), \dots$ is also an i.i.d. sequence with finite mean, and, with probability 1,

$$\lim_{n \rightarrow \infty} \frac{r(Y_1) + \dots + r(Y_n)}{n} = E(r(Y)).$$

Remarkably, the i.i.d. assumption of the strong law can be significantly weakened.

Strong Law of Large Numbers for Markov Chains

Theorem 5.1. *Assume that X_0, X_1, \dots is an ergodic Markov chain with stationary distribution π . Let r be a bounded, real-valued function. Let X be a random variable with distribution π . Then, with probability 1,*

$$\lim_{n \rightarrow \infty} \frac{r(X_1) + \dots + r(X_n)}{n} = E(r(X)),$$

where $E(r(X)) = \sum_j r(j)\pi_j$.

Although Markov chains are not independent sequences, the theorem is a consequence of the fact that, for ergodic chains, successive excursions between visits to the same state are independent. A proof of the strong law of large numbers for Markov chains may be found in Norris (1998).

Given an ergodic Markov chain with stationary distribution π , let A be a nonempty subset of the state space. Write $\pi_A = \sum_{j \in A} \pi_j$. From Theorem 5.1, we can interpret π_A

as the long-term average number of visits of an ergodic Markov chain to A . Define the indicator variable

$$I_A(x) = \begin{cases} 1, & \text{if } x \in A, \\ 0, & \text{if } x \notin A. \end{cases}$$

Then, $\sum_{k=0}^{n-1} I_A(X_k)$ is the number of visits to A in the first n steps of the chain. Let X be a random variable with distribution π . With probability 1,

$$\lim_{n \rightarrow \infty} \frac{1}{n} \sum_{k=0}^{n-1} I_A(X_k) = E(I_A(X)) = P(X \in A) = \pi_A.$$

One setting where the strong law for Markov chains arises is when there is a reward, or cost, function associated with the states of the chain.

■ **Example 5.1** Bob's daily lunch choices at the cafeteria are described by a Markov chain with transition matrix

$$P = \begin{array}{c} \begin{array}{c} \text{Yogurt} \\ \text{Salad} \\ \text{Hamburger} \\ \text{Pizza} \end{array} \begin{array}{c} \text{Yogurt} \\ \text{Salad} \\ \text{Hamburger} \\ \text{Pizza} \end{array} \begin{pmatrix} 0 & 0 & 1/2 & 1/2 \\ 1/4 & 1/4 & 1/4 & 1/4 \\ 1/4 & 0 & 1/4 & 1/2 \\ 1/4 & 0 & 1/4 & 1/2 \end{pmatrix} \end{array}.$$

Yogurt costs \$3.00, hamburgers cost \$7.00, and salad and pizza cost \$4.00 each. Over the long term, how much, on average, does Bob spend for lunch?

Solution Let

$$r(x) = \begin{cases} 3, & \text{if } x = \text{yogurt}, \\ 4, & \text{if } x = \text{salad or pizza}, \\ 7, & \text{if } x = \text{hamburger}. \end{cases}$$

The lunch chain is ergodic with stationary distribution

Yogurt	Salad	Hamburger	Pizza
7/65	2/13	18/65	6/13

With probability 1, Bob's average lunch cost converges to

$$\sum_x r(x)\pi_x = 3 \left(\frac{7}{65} \right) + 4 \left(\frac{2}{13} + \frac{6}{13} \right) + 7 \left(\frac{18}{65} \right) = \$4.72 \text{ per day.}$$

■

Armed with the strong law of large numbers for Markov chains, we now describe the details of MCMC. As is often the case, the knowing is in the doing, so we start with an expanded example.

Binary Sequences with No Adjacent 1s

The following is a toy problem for which an exact analysis is possible. Consider sequences of length m consisting of 0s and 1s. Call a sequence *good* if it has no adjacent 1s. What is the expected number of 1s in a good sequence if all good sequences are equally likely?

For $m = 4$, there are $2^4 = 16$ binary sequences of 0s and 1s. The eight good sequences are

$$(0000), (1000), (0100), (0010), (0001), (1010), (1001), (0101),$$

and the desired expectation is

$$\frac{1}{8}(0 + 1 + 1 + 1 + 1 + 2 + 2 + 2) = \frac{10}{8} = 1.25.$$

For general m , the expected number of 1s is $\mu = \sum_k k\pi_k$, where π_k is the probability that a good sequence of length m has exactly k 1s. While π_k can be derived by a combinatorial argument (see Exercise 5.4), there is no simple closed form expression for μ . So we approach the problem using simulation.

If x is a sequence of 0s and 1s of length m , let $r(x)$ be the number of 1s in the sequence. Assume that we are able to generate a uniformly random sequence of length m with no adjacent 1s. Doing this repeatedly and independently, generating good sequences Y_1, Y_2, \dots, Y_n , a Monte Carlo estimate of the desired expectation is

$$\mu \approx \frac{r(Y_1) + r(Y_2) + \dots + r(Y_n)}{n},$$

for large n . This is by the (regular) law of large numbers.

Thus, the problem of estimating μ reduces itself to the problem of simulating a good sequence.

Here is one way to generate such a sequence, which does not use Markov chains, based on the *rejection method*. Generate a sequence of 0s and 1s with no constraints. This is easy to do by just flipping m coins, where heads represents 1 and tails represents 0. If the resulting sequence is good, then keep it. If the sequence has adjacent 1s then reject it, and try again.

The approach works in principle, but not in practice. Even for moderate m , most binary sequences will have some adjacent 1s, and thus be rejected. For instance, for $m = 100$, there are $2^{100} \approx 10^{30}$ binary sequences, of which *only* 10^{21} are good. The probability that a binary sequence is good is about $10^{21}/10^{30} = 10^{-9}$. Thus, the rejection algorithm will typically see about one billion sequences before one good sequence appears. This method is prohibitively slow.

Here is an alternate approach using Markov chains. The idea is to construct an ergodic Markov chain X_0, X_1, \dots whose state space is the set of good sequences and whose limiting distribution is uniform on the set of good sequences. The Markov chain is then generated and, as in the i.i.d. case, we take

$$\mu \approx \frac{r(X_1) + r(X_2) + \dots + r(X_n)}{n},$$

for large n , as an MCMC estimate for the desired expectation.

The Markov chain is constructed as a random walk on a graph whose vertices are good sequences. The random walk proceeds as follows. From a given good sequence, pick one of its m components uniformly at random.

If the component is 1, then switch it to 0. Since the number of 1s is reduced, the resulting sequence is good. The walk moves to the new sequence.

If the component is 0, then switch it to 1 only if the resulting sequence is good. In that case, the walk moves to the new sequence. However, if the switch to 1 would result in a bad sequence, then stay put. The walk stays at the current state.

The walk can be described as a random walk on a weighted graph with weights assigned as follows. Two good sequences that differ in only one component are joined by an edge of weight 1. Also, each sequence possibly has a loop. The weight of the loop is such that the sum of the weights of all the edges incident to the sequence is equal to m .

Since all vertices have the same total weight, the stationary distribution is uniform. See Figure 5.1 for the weighted graph construction for $m = 4$.

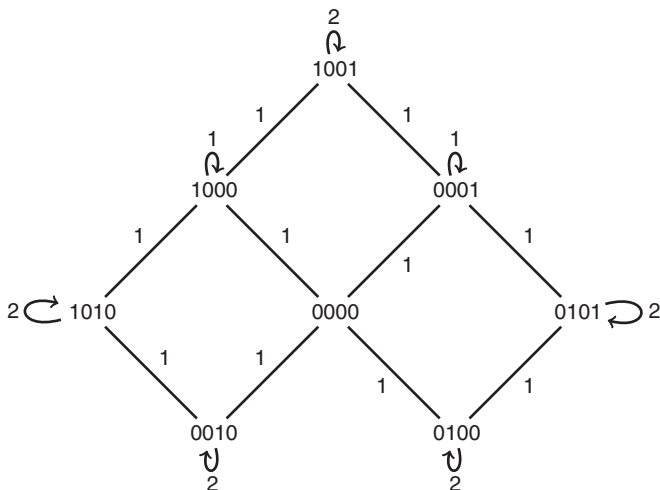


Figure 5.1 Weighted graph on sequences with no adjacent 1s.

The MCMC random walk algorithm was implemented for sequences of length $m = 100$. The chain was run for 100,000 steps (taking 2 seconds on a laptop

computer), starting with the sequence of all 0s as the initial state of the chain. The MCMC estimate for the expected number of 1s is $\mu \approx 27.833$. An exact analysis gives $\mu = 27.7921$.

R : Expected Number of 1s in Good Sequences

```
> # adjacent.R
> # init: initial sequence
> # n: number of steps to run the chain
> adjacent <- function(init, n)
+ { k <- length(init)
+   tot <- 0 # total number of 1s
+   new <- c(2, init, 2) # pad sequence at the ends
+   for (i in 1:n) {
+     index <- 1 + sample(1:k, 1)
+     newbit <- 0 + !new[index] # flip the bit
+     if (newbit == 0) {
+       new[index] <- 0
+       tot <- tot + sum(new)
+     } else {
+       if (new[index-1] == 1 | new[index+1] == 1) {
+         tot <- tot + sum(new)
+         next
+       } else {new[index] <- 1}
+       tot <- tot + sum(new)
+     }
+   }
+   tot/n - 4 } # subtract both endpoints
> m <- 100
> init <- rep(0, m) # Start at sequence of all 0s
> adjacent(init, 100000)
[1] 27.83333
```

Remarks:

1. The number of vertices of the random walk graph is huge—about 10^{21} good sequences for $m = 100$. However, the number of steps required for the walk to get sufficiently close to the limiting distribution is a small fraction of that—in this implementation 100,000 steps.
2. The uniform distribution on the set of good sequences assigns probability $1/c$ to each sequence, where c is the number of good sequences. The actual value of c was never needed in the implementation, and for all practical purposes could have been unknown.
3. A benefit of the algorithm that cannot be overstated is that it is easily and efficiently coded and is intuitive based on the structure of the problem.

5.2 METROPOLIS–HASTINGS ALGORITHM

The most common Markov chain Monte Carlo method is the Metropolis–Hastings algorithm. The algorithm is named after Nicholas Metropolis, a physicist who led the research group at Los Alamos National Laboratory, which first proposed the method in the early 1950s, and W.K. Hastings, a Canadian statistician, who extended the scope of the algorithm in 1970.

Let $\pi = (\pi_1, \pi_2, \dots)$ be a discrete probability distribution. The algorithm constructs a reversible Markov chain X_0, X_1, \dots whose stationary distribution is π .

Let T be a transition matrix for *any* irreducible Markov chain with the same state space as π . It is assumed that the user knows how to sample from T . The T chain will be used as a *proposal* chain, generating elements of a sequence that the algorithm decides whether or not to *accept*.

To describe the transition mechanism for X_0, X_1, \dots , assume that at time n the chain is at state i . The next step of the chain X_{n+1} is determined by a two-step procedure.

1. Choose a new state according to T . That is, choose j with probability T_{ij} . State j is called the *proposal state*.
2. Decide whether to accept j or not. Let

$$a(i, j) = \frac{\pi_j T_{ji}}{\pi_i T_{ij}}.$$

The function a is called the *acceptance function*. If $a(i, j) \geq 1$, then j is accepted as the next state of the chain. If $a(i, j) < 1$, then j is accepted with probability $a(i, j)$. If j is not accepted, then i is kept as the next step of the chain.

In other words, assume that $X_n = i$. Let U be uniformly distributed on $(0, 1)$. Set

$$X_{n+1} = \begin{cases} j, & \text{if } U \leq a(i, j), \\ i, & \text{if } U > a(i, j). \end{cases}$$

Metropolis–Hastings Algorithm

The sequence X_0, X_1, \dots constructed by the Metropolis–Hastings algorithm is a reversible Markov chain whose stationary distribution is π .

Proof. It should be clear that X_0, X_1, \dots is in fact a Markov chain, as each X_{n+1} , given the past history X_0, \dots, X_n , only depends on X_n . Let P be the transition matrix of the chain. The theorem will be proven by showing that the detailed balance equations $\pi_i P_{ij} = \pi_j P_{ji}$ are satisfied.

For $i \neq j$, consider $P_{ij} = P(X_1 = j | X_0 = i)$. Given $X_0 = i$, then $X_1 = j$ if and only if (i) j is proposed, and (ii) j is accepted. The first occurs with probability T_{ij} .

The second occurs if $U \leq a(i, j)$, where U is uniform on $(0, 1)$. We have that

$$\begin{aligned} P(U \leq a(i, j)) &= \begin{cases} a(i, j), & \text{if } a(i, j) \leq 1, \\ 1, & \text{if } a(i, j) > 1, \end{cases} \\ &= \begin{cases} a(i, j), & \text{if } \pi_j T_{ji} \leq \pi_i T_{ij}, \\ 1, & \text{if } \pi_j T_{ji} > \pi_i T_{ij}. \end{cases} \end{aligned}$$

Thus, for $i \neq j$,

$$P_{ij} = \begin{cases} T_{ij} a(i, j), & \text{if } \pi_j T_{ji} \leq \pi_i T_{ij}, \\ T_{ij}, & \text{if } \pi_j T_{ji} > \pi_i T_{ij}. \end{cases}$$

The diagonal entries of \mathbf{P} are not needed for the proof. They are determined by the fact that the rows of \mathbf{P} sum to 1.

To show the detailed balance equations are satisfied, assume that $\pi_j T_{ji} \leq \pi_i T_{ij}$. Then,

$$\pi_i P_{ij} = \pi_i T_{ij} a(i, j) = \pi_i T_{ij} \frac{\pi_j T_{ji}}{\pi_i T_{ij}} = \pi_j T_{ji} = \pi_j P_{ji}.$$

Similarly, if $\pi_j T_{ji} > \pi_i T_{ij}$,

$$\pi_i P_{ij} = \pi_i T_{ij} = \pi_j T_{ji} \frac{\pi_i T_{ij}}{\pi_j T_{ji}} = \pi_j T_{ji} a(j, i) = \pi_j P_{ji}.$$

■

Remarks:

1. The exact form of π is not necessary to implement Metropolis–Hastings. The algorithm only uses ratios of the form π_j/π_i . Thus, π needs only to be specified up to proportionality. For instance, if π is uniform on a set of size c , then $\pi_j/\pi_i = 1$, and the acceptance function reduces to $a(i, j) = T_{ji}/T_{ij}$.
2. If the proposal transition matrix \mathbf{T} is symmetric, then $a(i, j) = \pi_j/\pi_i$.
3. The algorithm works for *any* irreducible proposal chain. Thus, the user has wide latitude to find a proposal chain that is efficient in the context of their problem.
4. If the proposal chain is ergodic (irreducible and aperiodic in the finite case) then the resulting Metropolis–Hastings chain is also ergodic with limiting distribution π .
5. The generated sequence X_0, X_1, \dots, X_n gives an approximate sample from π . However, if the chain requires many steps to get close to stationarity, there may be initial bias. *Burn-in* refers to the practice of discarding the initial iterations and retaining X_m, X_{m+1}, \dots, X_n , for some m . In that case, the strong law of large numbers for Markov chains gives

$$\lim_{n \rightarrow \infty} \frac{r(X_m) + \dots + r(X_n)}{n - m + 1} = \sum_x r(x) \pi_x.$$

■ **Example 5.2** Power-law distributions are positive probability distributions of the form $\pi_i \propto i^s$, for some constant S . Unlike distributions with exponentially decaying tails (e.g., Poisson, geometric, exponential, normal), power-law distributions have *fat tails*, and thus are often used to model skewed data. Let

$$\pi_i = \frac{i^{-3/2}}{\sum_{k=1}^{\infty} k^{-3/2}}, \text{ for } i = 1, 2, \dots$$

Implement a Metropolis–Hastings algorithm to simulate from π .

Solution For a proposal distribution, we use simple symmetric random walk on the positive integers with reflecting boundary at 1. From 1, the walk always moves to 2. Otherwise, the walk moves left or right with probability 1/2. The proposal chain transition matrix is

$$T_{ij} = \begin{cases} 1/2, & \text{if } j = i \pm 1 \text{ for } i > 1, \\ 1, & \text{if } i = 1 \text{ and } j = 2, \\ 0 & \text{otherwise.} \end{cases}$$

The acceptance function is

$$a(i, i+1) = \left(\frac{i}{i+1}\right)^{3/2}, \text{ and } a(i+1, i) = \left(\frac{i+1}{i}\right)^{3/2}, \text{ for } i \geq 2,$$

with

$$a(1, 2) = \frac{\pi_2 T_{21}}{\pi_1 T_{12}} = \left(\frac{1}{2}\right)^{3/2} \frac{1}{2} = \left(\frac{1}{2}\right)^{5/2} \text{ and } a(2, 1) = 2^{5/2}.$$

Observe that $a(i+1, i) \geq 1$, for all i .

The Metropolis–Hastings algorithm can be described as follows. From state $i \geq 2$, flip a fair coin. If heads, go to $i-1$. If tails, set $p = (i/(i+1))^{3/2}$. Flip another coin whose heads probability is p . If heads, go to $i+1$. If tails, stay at i . If the chain is at 1, then move to 2 with probability $(1/2)^{5/2} \approx 0.177$. Otherwise, stay at 1.

The chain was run for one million steps. See Table 5.1 to compare the simulated and exact probabilities.

TABLE 5.1 Comparison of Markov chain Monte Carlo Estimates with Exact Probabilities for Power-Law Distribution

i	1	2	3	4	5	6	7	8	≥ 9
Simulation	0.389	0.137	0.075	0.048	0.034	0.026	0.021	0.017	0.252
Exact	0.383	0.135	0.074	0.048	0.034	0.026	0.021	0.017	0.262

R : Power Law Distribution

```
# powerlaw.R
> trials <- 1000000
> simlist <- numeric(trials)
> simlist[1] <- 2
> for (i in 2:trials) {
+   if (simlist[i-1] ==1) {
+     p <- (1/2)^(5/2)
+     new <- sample(c(1,2),1,prob=c(1-p,p))
+     simlist[i] <- new
+   } else { leftright <- sample(c(-1,1),1)
+     if (leftright == -1) {
+       simlist[i] <- simlist[i-1] - 1} else {
+     p <- (simlist[i-1]/(simlist[i-1]+1))^(3/2)
+     simlist[i] <- sample(c(simlist[i-1],
+       1+simlist[i-1]), 1,prob=c(1-p,p))
+   } } }
> tab <- table(simlist)/trials
> tab[1:8]
```

1	2	3	4	5	6	7	8
0.389	0.137	0.075	0.048	0.034	0.026	0.021	0.017

■

Example 5.3 (Cryptography)

ahicainqcaqx ic zqcqwbl bwq zwqbj xjustlicz tlhamx ic jyq kbr ho jyb j albx ho
 jyicmwx kyh ybgq tqqc qnuabj qn jh mchk chjyicz ho jyq jyqhwr ho dwhtbtlijqjx jyb
 jyqhwr jh kyiaj jyq shxj zlhwhihux htpqajx ho yusbc wqxqbway bwq icnqtjqn ohw jyq
 shxj zlhwhihux ho illuxjwbjihcx

—qnzbow bllqc dhq jyq suwnqwx ic jyq wuq shwzuq

The coded message was formed by a simple substitution cipher—each letter standing for one letter in the alphabet. The hidden message can be thought of as a coding function from the letters of the coded message to the regular alphabet. For example, given an encrypted message *xoaaoaaoggo*, the function that maps *x* to *m*, *o* to *i*, *a* to *s*, and *g* to *p* decodes the message to *mississippi*.

If one keeps tracks of all 26 letters and spaces, there are $27! \approx 10^{28}$ possible coding functions. The goal is to find the one that decodes the message.

This example is based on Diaconis (2009), who gives a compelling demonstration of the power and breadth of MCMC applied to cryptography.

My colleague Jack Goldfeather assisted in downloading the complete works of Jane Austen (about four million characters) from Project Gutenberg's online site and recording the number of transitions of consecutive text symbols. For simplicity,

we ignored case and only kept track of spaces and the letters a to z . The counts are kept in a 27×27 matrix \mathbf{M} of transitions indexed by $(a, b, \dots, z, [\text{space}])$. For example, there are 6,669 places in Austen's work where b follows a and thus $M_{ab} = 6,669$.

The encoded message has 320 characters, denoted as (c_1, \dots, c_{320}) . For each coding function f , associate a score

$$\text{score}(f) = \prod_{i=1}^{319} M_{f(c_i), f(c_{i+1})}.$$

The score is a product over all successive pairs of letters in the decrypted text $(f(c_i), f(c_{i+1}))$ of the number of occurrences of that pair in the reference Austen library. The score is higher when successive pair frequencies in the decrypted message match those of the reference text. Coding functions with high scores are good candidates for decryption. The goal is to find the coding function of maximum score.

A probability distribution proportional to the scores is obtained by letting

$$\pi_f = \frac{\text{score}(f)}{\sum_g \text{score}(g)}. \quad (5.1)$$

From a Monte Carlo perspective, we want to sample from π , with the idea that a sample is most likely to return a value of high probability. The denominator in Equation (5.1) is intractable, being the sum of $27!$ terms. But the beauty of Metropolis–Hastings is that the denominator is not needed since the algorithm relies on ratios of the form π_f / π_g .

The MCMC implementation runs a random walk on the set of coding functions. Given a coding function f , the transition to a proposal function f^* is made by picking two letters at random and switching the values that f assigns to these two letters. This method of *random transpositions* gives a symmetric proposal matrix T , simplifying the acceptance function

$$a(f, f^*) = \frac{\pi_{f^*} T_{f^*, f}}{\pi_f T_{f, f^*}} = \frac{\pi_{f^*}}{\pi_f} = \frac{\text{score}(f^*)}{\text{score}(f)}.$$

The algorithm is as follows:

1. Start with any coding function f . For convenience, we use the identity function.
2. Pick two letters uniformly at random and switch the values that f assigns to these two letters. Call the new proposal function f^* .
3. Compute the acceptance function $a(f, f^*) = \text{score}(f^*) / \text{score}(f)$.
4. Let U be uniformly distributed on $(0, 1)$. If $U \leq a(f, f^*)$, accept f^* . Otherwise, stay with f .

We ran the algorithm on the coded message at the start of this example. See the script file **decode.R**. At iteration 2658, the message was decoded.

R : Decoding the Message

[100] xlitxisatxau it hatawnc nwa hwand udepocith oclxfu it dra mng ly drnd xcnuu ly dritfawu mrl rnza oaat asexndas dl flm tldrith ly dra dralwg ly bwlonoicidiau drnd dralwg dl mrixx dra plud hclwileu lojaxdu ly repnt wauanwxr nwa itsaodas ylw dra plud hclwileu ly icceudwndiltu – ashnw nccat bla dra pewsawu it dra wea plwhea

[500] gsadgaredgen ad wedeoil ioe woeit ntubcladw clsgyn ad the kif sm thit glinn sm thadyeon khs hive ceed erugiter ts ydsk dsthaw sm the thesof sm poscicalataen thit thesof ts khagh the bsnt wlsoasun szegtn sm hubid oeneiogh ioe adrecter mso the bsnt wlsoasun sm alluntoitasdn – erwio illed pse the buoreon ad the oue bsowue

[1000] goidgimedges id wedenal ane wneat strpelidw clogus id the fak oy that glass oy thiduens fho have ceed emrgatem to udof dothidw oy the theonk oy bnocacilities that theonk to fhgh the post wloniors ocxegts oy hrpad neseangh ane idmectem yon the post wloniors oy illrstnatiods – emwan alled boe the prnmens id the nre ponwre

[1400] goingidenges in beneral are breat stumplinb plogks in the way of that glass of thinkers who have been edugated to know nothinb of the theory of cropapilities that theory to whigh the most blorious opxegts of human researgh are indepted for the most blorious of illustrations – edbar allen coe the murders in the rue morbue

[1600] goingidenges in ceneral are creat stumplinc plogks in the way of that glass of thinkers who have been edugated to know nothinc of the theory of bropapilities that theory to whigh the most clorious opxegts of human researgh are indepted for the most clorious of illustrations – edcar allen boe the murders in the rue morcue

[1800] coincidences in general are great stumpling plocks in the way of that class of thinkers who have been educated to know nothing of the theory of bropapilities that theory to which the most glorious opzects of human research are indepted for the most glorious of illustrations – edgar allen boe the murders in the rue morgue

[2658] coincidences in general are great stumbling blocks in the way of that class of thinkers who have been educated to know nothing of the theory of probabilities that theory to which the most glorious objects of human research are indebted for the most glorious of illustrations – edgar allen poe the murders in the rue morgue

■ **Example 5.4 (Darwin's finches)** The following example is based on Cobb and Chen (2003).

Charles Darwin, on his visit to the Galapagos Islands in 1835, discovered several species of finches, which varied from island to island. It is said that the variation

and diversity of the birds he observed was a significant factor which led Darwin to develop his theory of natural selection.

Darwin chronicled the presence of 13 species of finches on 17 islands. The data are presented in the *co-occurrence matrix* in Figure 5.2. Ecologists use such species-presence matrices to study levels of competition and cooperation among species.

Species	Islands																	Total
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	
A	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	17
B	1	1	1	1	1	1	1	1	1	1	1	0	0	1	0	1	1	14
C	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	0	14
D	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	13
E	1	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	12
F	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	11
G	1	1	1	1	1	0	1	1	0	0	0	0	0	1	0	1	1	10
H	1	1	1	1	1	1	0	1	1	1	1	0	0	0	0	0	0	10
I	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	10
J	1	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	6
K	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	2
L	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
M	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1
Total	11	10	10	10	10	9	9	9	8	8	7	4	4	4	3	3	3	122

Figure 5.2 Co-occurrence matrix for Darwin’s finches.

A species pair and an island pair constitute a *checkerboard* if each species appears on different islands. A checkerboard is a 2×2 submatrix of the form

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \text{ or } \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

For instance, in Figure 5.2, species E and G on islands 10 and 14, respectively, form a checkerboard.

The number of checkerboards in a co-occurrence matrix is a measure of competition. A large number of checkerboards would indicate a high degree of competition. For the finches data, there are 333 checkerboards. Is this number large or small?

To explore the question, consider the expected number of checkerboards for a typical 0–1 co-occurrence matrix, which is constrained by the row and column totals of the finches matrix.

From a theoretical point of view, the expectation is obtained by listing all such 0–1 matrices, counting the number of checkerboards in each, and taking an overall average. From a practical point of view, however, things are not so neat.

Counting 0–1 matrices with fixed marginal totals is a well-studied problem in applied mathematics. The number of matrices with the same row and column sums as the finches data are not known. Liu (2001) reports that Susan Holmes at Stanford University estimated the number as about 6.715×10^{16} . It is not computationally feasible to list all such matrices and count the checkerboards in each.

We implement a Metropolis–Hastings algorithm to generate a uniformly random co-occurrence matrix and to estimate the probability that such a matrix has at least 333 checkerboards.

Consider the following *swap* operation. Assume that \mathbf{M} is a co-occurrence matrix with at least one checkerboard. Pick a checkerboard in the matrix and swap 0s and 1s. That is, if the chosen checkerboard submatrix is $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$, then change it to $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$. If the submatrix is $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$, change it to $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$. Let \mathbf{N} be the matrix that results from \mathbf{M} after the swap operation. The swap operation preserves row and column sums. That is, \mathbf{N} is a co-occurrence matrix with the same row and column totals as \mathbf{M} .

The swap operation is the basis of an MCMC random walk algorithm on a graph whose vertex set is the set of all co-occurrence matrices with fixed row and column sums. Matrices \mathbf{M} and \mathbf{N} are joined by an edge if \mathbf{N} can be obtained from \mathbf{M} by one checkerboard swap.

For example, there are five 3×3 co-occurrence matrices with successive row sums 2, 2, and 1, and column sums 2, 2, and 1. The corresponding graph is shown in Figure 5.3. Simple random walk on this graph has transition matrix

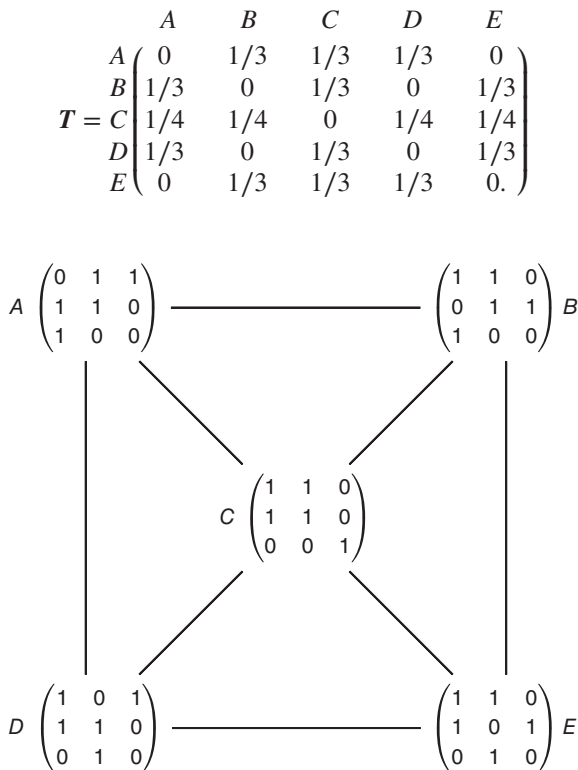


Figure 5.3 Transition graph for co-occurrence matrices.

Consider simple random walk on a general co-occurrence graph. Although it is not completely obvious, such a Markov chain is irreducible (see Ryser 1957) and has a unique stationary distribution. Unfortunately, the stationary distribution is not the uniform distribution, which is the one we want. The stationary distribution for simple random walk is proportional to the vertex degrees. For the graph in Figure 5.3, the stationary distribution s is

$$s = (s_A, s_B, s_C, s_D, s_E) = \left(\frac{3}{16}, \frac{3}{16}, \frac{4}{16}, \frac{3}{16}, \frac{3}{16} \right).$$

Let π denote the uniform distribution on the set of co-occurrence matrices with fixed row and column totals. That is, $\pi_M = 1/c$, for all co-occurrence matrices M , where c is the number of such matrices. For MCMC, simple random walk on the co-occurrence graph is used as the proposal distribution. Since $\pi_i/\pi_j = 1$ for all i and j , the acceptance function, for adjacent vertices i and j , is

$$a(i, j) = \frac{\pi_j T_{ji}}{\pi_i T_{ij}} = \frac{\deg(j)}{\deg(i)} = \frac{\text{Number of checkerboards in } j}{\text{Number of checkerboards in } i}.$$

The algorithm is described as follows. From a co-occurrence matrix i , pick two rows and two columns uniformly at random, until the resulting submatrix is swappable. Do the swap and let j be the resulting co-occurrence matrix. Count the number of checkerboards in i and j , and accept the new matrix according to the acceptance function $a(i, j)$.

The algorithm is implemented in **darwin.R** for 5,000 steps, obtaining a distribution of the number of checkerboards in an approximately uniform co-occurrence matrix. See Figure 5.4. Estimates for the mean and standard deviation are 246.9 and 15.4, respectively. Of 5,000 matrices, only two had 333 or more checkerboards. Thus, a *typical* co-occurrence matrix contains about 247 checkerboards give or take about 30, and the estimated probability that a random co-occurrence matrix has at least 333 checkerboards is $2/5,000 = 0.0004$. The large number of co-occurrence matrices in the data is unusual if the distribution of finches on the islands was random. An ecologist might conclude that the distribution of finch species in the Galapagos Islands shows evidence of competition. ■

Continuous State Space

Although the target distributions in the previous examples are discrete, MCMC can also be used in the continuous case, when π is a probability density function.

We have not yet studied continuous state space stochastic processes. Intuitively, for a continuous state space *Markov process* a transition *function* replaces the transition matrix, where P_{ij} is the value of a conditional density function given $X_0 = i$.

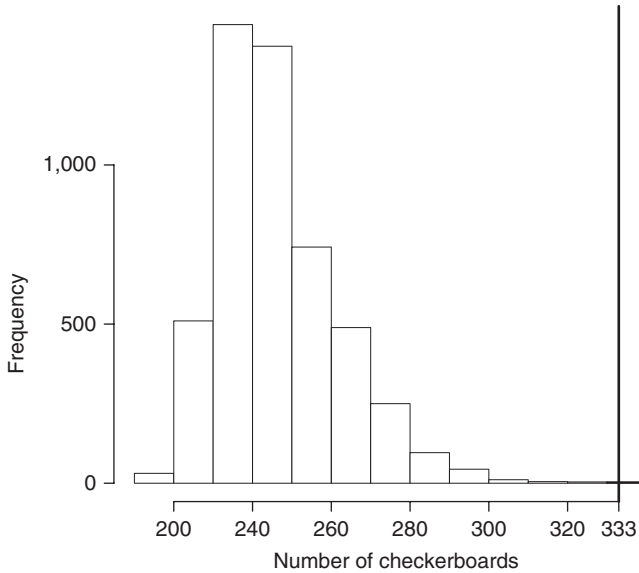


Figure 5.4 Simulated distribution of the number of checkerboards in a uniformly random co-occurrence matrix. The probability of at least 333 checkerboards is 0.0004.

The Metropolis–Hastings algorithm is essentially the same as in the discrete case, with densities replacing discrete distributions. Without delving into any new theory, we present the continuous case by example.

■ **Example 5.5** Using only a uniform random number generator, simulate a standard normal random variable using MCMC.

Solution Write the standard normal density function as $\pi_t = e^{-t^2/2}/\sqrt{2\pi}$. For the Metropolis–Hastings proposal distribution, we choose the uniform distribution on an interval of length two centered at the current state. From state s , the proposal chain moves to t , where t is uniformly distributed on $(s - 1, s + 1)$. Hence, the conditional density given s is constant, with

$$T_{st} = \frac{1}{2}, \text{ for } s - 1 \leq t \leq s + 1.$$

The acceptance function is

$$a(s, t) = \frac{\pi_t T_{ts}}{\pi_s T_{st}} = \left(\frac{e^{-t^2/2}}{\sqrt{2\pi}} \right) (1/2) \bigg/ \left(\frac{e^{-s^2/2}}{\sqrt{2\pi}} \right) (1/2) = e^{-(t^2 - s^2)/2}.$$

Notice that the length of the interval for the uniform proposal distribution does not affect the acceptance function.

See Figure 5.5 for the results of one million iterations of the MCMC sequence. The overlaid curve is the standard normal density function.

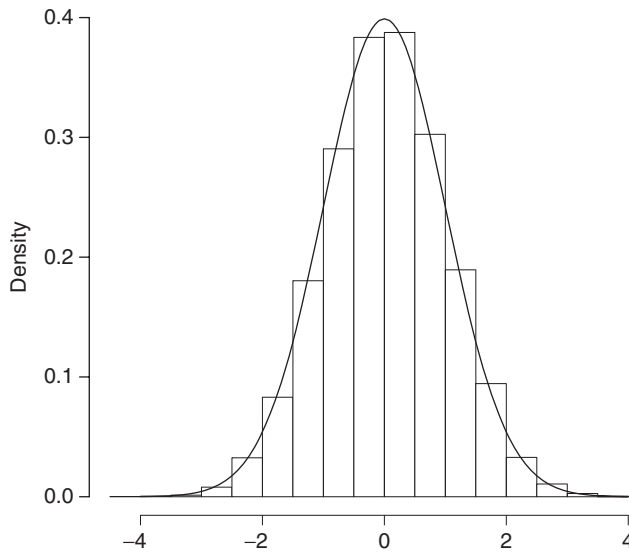


Figure 5.5 Simulation of standard normal distribution with MCMC.

R : Standard Normal Simulation

```
# standardnormal.R
> trials <- 1000000
> simlist <- numeric(trials)
> state <- 0
> for (i in 2:trials) {
+   prop <- runif(1, state-1, state+1)
+   acc <- exp(-(prop^2 - state^2)/2)
+   if (runif(1) < acc) state <- prop
+   simlist[i] <- state }
> hist(simlist, xlab="", main="", prob=T)
> curve(dnorm(x), -4, 4, add=T)
```

■

5.3 GIBBS SAMPLER

The original Metropolis–Hastings algorithm was discovered in 1953 and motivated by problems in physics. In 1984, a landmark paper by brothers Donald and Stuart Geman showed how the algorithm could be adapted to the high-dimensional problems

that arise in Bayesian statistics. Their new algorithm was named after the physicist Josiah Gibbs with reference to connections to statistical physics.

In the Gibbs sampler, the target distribution π is an m -dimensional joint density

$$\pi(\mathbf{x}) = \pi(x_1, \dots, x_m).$$

A multivariate Markov chain is constructed whose limiting distribution is π , and which takes values in an m -dimensional space. The algorithm generates elements of the form

$$\begin{aligned} \mathbf{X}^{(0)}, \mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \dots \\ = (X_1^{(0)}, \dots, X_m^{(0)}), (X_1^{(1)}, \dots, X_m^{(1)}), (X_1^{(2)}, \dots, X_m^{(2)}), \dots \end{aligned}$$

by iteratively updating each component of an m -dimensional vector conditional on the other $m - 1$ components. We show that the Gibbs sampler is a special case of the Metropolis–Hastings algorithm, with a particular choice of proposal distribution.

The algorithm is sometimes hard to understand because of an abundance of notation. We start off with a relatively simple two-dimensional example.

■ **Example 5.6** Consider a bivariate standard normal distribution with correlation ρ . For background, see Appendix B, Section B.4. The bivariate normal distribution has the property that conditional distributions are normal. If (X, Y) has a bivariate standard normal distribution, then the conditional distribution of X given $Y = y$ is normal with mean ρy and variance $1 - \rho^2$. Similarly, the conditional distribution of Y given $X = x$ is normal with mean ρx and variance $1 - \rho^2$.

The Gibbs sampler is implemented to simulate (X, Y) from a bivariate standard normal distribution with correlation ρ . At each step of the algorithm, one component of a two-element vector is updated by sampling from its conditional distribution given the other component. Updates switch back and forth. The resulting sequence of bivariate samples converges to the target distribution.

1. Initialize: $(x_0, y_0) \leftarrow (0, 0)$
 $m \leftarrow 1$.
2. Generate x_m from the conditional distribution of X given $Y = y_{m-1}$. That is, simulate from a normal distribution with mean ρy_{m-1} and variance $1 - \rho^2$.
3. Generate y_m from the conditional distribution of Y given $X = x_m$. That is, simulate from a normal distribution with mean ρx_m and variance $1 - \rho^2$.
4. $m \leftarrow m + 1$.
5. Return to Step 2.

We simulated a bivariate standard normal distribution with $\rho = 0.60$ using the Gibbs sampler. The chain was run for 2,000 steps. In R, the output is a 2000×2 matrix. A scatterplot of the output is seen in Figure 5.6.

■

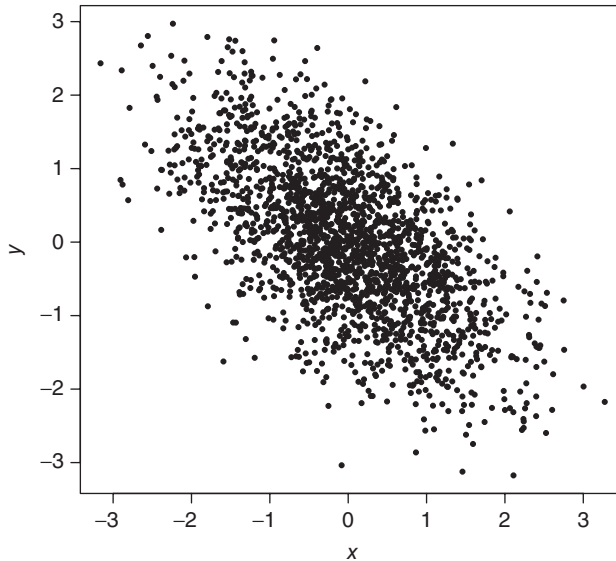


Figure 5.6 Bivariate standard normal simulation.

R : Bivariate Standard Normal

```
# bivariatestandard.R
> rho <- -0.60
> trials <- 2000
> sdev <- sqrt(1-rho^2)
> simlist <- matrix(rep(0,2*trials),ncol=2)
> for (i in 2:trials) {
+ simlist[i,1] <- rnorm(1,rho*simlist[i-1,2],sdev)
+ simlist[i,2] <- rnorm(1,rho*simlist[i,1],sdev) }
> plot(simlist,pch=20,xlab="x",ylab="y",main="")
```

■ **Example 5.7** The following implementation of the Gibbs sampler for a three-dimensional joint distribution is a classic example based on Casella and George (1992). Random variables X , P , and N have joint density

$$\pi(x, p, n) \propto \binom{n}{x} p^x (1-p)^{n-x} \frac{4^n}{n!},$$

for $x = 0, 1, \dots, n$, $0 < p < 1$, $n = 0, 1, \dots$. The p variable is continuous; x and n are discrete.

The Gibbs sampler requires being able to simulate from the conditional distributions of each component given the remaining variables. The trick to identifying these conditional distributions is to treat the two conditioning variables in the joint density function as fixed constants.

The conditional distribution of X given $N = n$ and $P = p$ is proportional to $\binom{n}{x} p^x (1-p)^{n-x}$, for $x = 0, 1, \dots, n$, which is binomial with parameters n and p .

The conditional distribution of P given $X = x$ and $N = n$ is proportional to $p^x (1-p)^{n-x}$, for $0 < p < 1$, which gives a beta distribution with parameters $x + 1$ and $n - x + 1$.

The conditional distribution of N given $X = x$ and $P = p$ is proportional to $(1-p)^{n-x} 4^n / (n-x)!$, for $n = x, x+1, \dots$. This is a shifted Poisson distribution with parameter $4(1-p)$. That is, the conditional distribution is equal to the distribution of $Z + x$, where Z has a Poisson distribution with parameter $4(1-p)$.

The Gibbs sampler, with arbitrary initial value, is implemented as follows:

1. Initialize: $(x_0, p_0, n_0) \leftarrow (1, 0.5, 2)$
 $m \leftarrow 1$
2. Generate x_m from a binomial distribution with parameters n_{m-1} and p_{m-1} .
3. Generate p_m from a beta distribution with parameters $x_m + 1$ and $n_{m-1} - x_m + 1$.
4. Let $n_m = z + x_m$, where z is simulated from a Poisson distribution with parameter $4(1 - p_m)$.
5. $m \leftarrow m + 1$
6. Return to Step 2.

The output of the Gibbs sampler is a sequence of samples

$$(X_0, P_0, N_0), (X_1, P_1, N_1), (X_2, P_2, N_2), \dots$$

In R, the output is stored in a matrix with three columns. Each column gives a sample from the marginal distribution. Each pair of columns gives a sample from a bivariate joint distribution. See Figure 5.7 for graphs of joint and marginal distributions. ■

R : Trivariate Distribution

```
# trivariate.R
> trials <- 5000
> sim <- matrix(rep(0,3*trials),ncol=3)
> sim[1,] <- c(1,0.5,2)
> for (i in 2:trials) {
+   sim[i,1] <- rbinom(1,sim[i-1,3],simlist[i-1,2])
+   sim[i,2] <- rbeta(1,sim[i,1]+1,sim[i-1,3]-sim[i,1]+1)
+   sim[i,3] <- rpois(1,4*(1-sim[i,2])+sim[i,1])
+ }
```

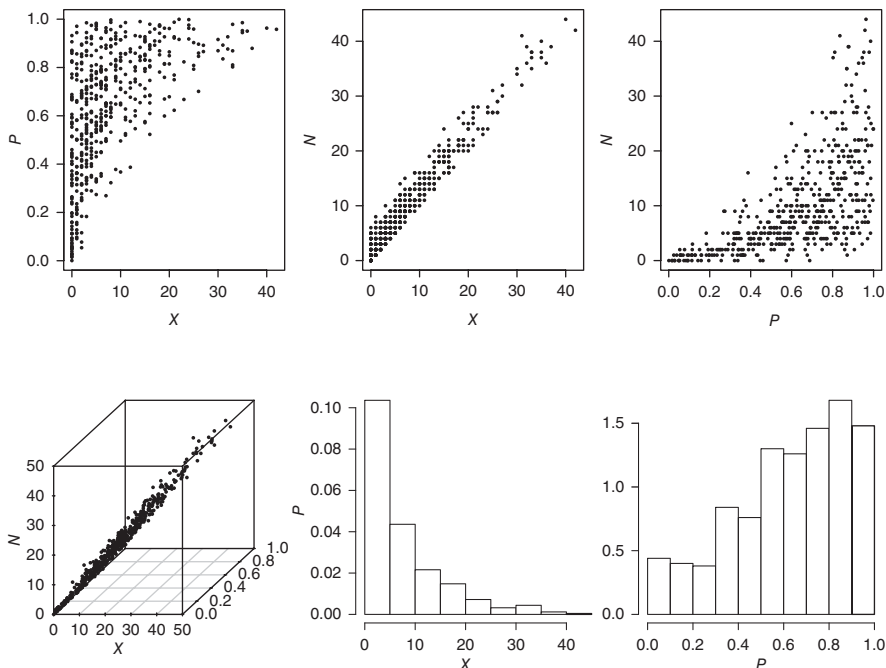


Figure 5.7 Joint and marginal distributions for trivariate distribution.

The Gibbs sampler is a special case of the Metropolis–Hastings algorithm. To see this, assume that π is an m -dimensional joint distribution. To avoid excessive notation and simplify the presentation, we just consider one step of the Gibbs sampler when the first component is being updated.

Assume that $\mathbf{i} = (x_1, x_2, \dots, x_m)$ is the current state, and $\mathbf{j} = (x'_1, x_2, \dots, x_m)$ is the proposed state. The proposal distribution T is the conditional distribution of X_1 given X_2, \dots, X_m . The acceptance function is $a(\mathbf{i}, \mathbf{j}) = \pi_j T_{ji} / \pi_i T_{ij}$.

We have that

$$\begin{aligned}
 \pi_j T_{ji} &= \pi(x'_1, x_2, \dots, x_m) f_{X_1|X_2, \dots, X_m}(x_1 | x_2, \dots, x_m) \\
 &= \pi(x'_1, x_2, \dots, x_m) \left(\frac{\pi(x_1, x_2, \dots, x_m)}{\int \pi(x, x_2, \dots, x_m) dx} \right) \\
 &= \pi(x_1, x_2, \dots, x_m) \left(\frac{\pi(x'_1, x_2, \dots, x_m)}{\int \pi(x, x_2, \dots, x_m) dx} \right) \\
 &= \pi(x_1, x_2, \dots, x_m) f_{X_1|X_2, \dots, X_m}(x'_1 | x_2, \dots, x_m) \\
 &= \pi_i T_{ij}.
 \end{aligned}$$

Thus, $a(i, j) = 1$. The proposal state is always accepted. The same is true for all of the m components. The algorithm can be implemented by either successively updating each component of the m -dimensional distribution, or by selecting components to update uniformly at random.

The Gibbs sampler is remarkably versatile, and can be applied to a large variety of complex multidimensional problems.

■ **Example 5.8 (Ising model)** The Ising model was originally proposed in physics as a model for magnetism. It also arises in image processing.

Consider a graph consisting of *sites* (vertices), in which each site v is assigned a *spin* of $+1$ or -1 . A *configuration* σ is an assignment of spins to each site. That is, $\sigma_v = \pm 1$, for all v . We will assume a square $n \times n$ grid of sites, where each site is connected to four neighbors (up, down, left, and right), except at the boundary. Thus, there are n^2 sites and 2^{n^2} possible configurations.

Associated with each configuration σ is its *energy*, defined as

$$E(\sigma) = - \sum_{v \sim w} \sigma_v \sigma_w,$$

where the sum is over all pairs of sites v and w , which are neighbors. Note that if most neighbors have similar spins, the energy is negative; if most neighbors have different spins, the energy is positive; and for a uniformly random assignment of spins, the energy is about 0. One checks that for the configuration σ in Figure 5.8, $E(\sigma) = 4$.

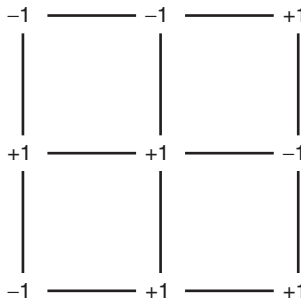


Figure 5.8 Sample configuration σ on 3×3 grid.

The *Gibbs distribution* is a probability distribution on the set of configurations, defined by

$$\pi_{\sigma} = \frac{e^{-\beta E(\sigma)}}{\sum_{\tau} e^{-\beta E(\tau)}},$$

where the sum in the denominator is over all configurations. The parameter β has a physical interpretation as the reciprocal of temperature. If $\beta = 0$ (infinite temperature), the distribution is uniform on the set of configurations. For $\beta > 0$,

the Gibbs distribution puts more mass on low-energy configurations, which favors neighbors of similar spin. For $\beta < 0$, the distribution puts more mass on high-energy configurations.

The Ising model is studied, in part, because of its *phase transition* properties. In two dimensions, the system undergoes a radical change of behavior at the critical temperature $1/\beta = 2/\ln(1 + \sqrt{2}) \approx 2.269$ ($\beta = 0.441$). Above that temperature, the system appears disorganized and chaotic. Below the critical temperature, the system is *magnetized*, and a phase transition occurs.

The Gibbs sampler is ideally suited for simulating the Ising model. Given a configuration σ , a site v is chosen uniformly at random. The spin at that site is then updated from the conditional distribution of that site given the other sites of σ .

Denote the sites as $1, \dots, m$. Let σ_k be the spin at site k . Let

$$\sigma_{-k} = (\sigma_1, \dots, \sigma_{k-1}, \sigma_{k+1}, \dots, \sigma_m)$$

denote the other $m - 1$ sites of the configuration. For fixed k , write

$$\sigma^+ = (\sigma_1, \dots, \sigma_{k-1}, +1, \sigma_{k+1}, \dots, \sigma_m)$$

and

$$\sigma^- = (\sigma_1, \dots, \sigma_{k-1}, -1, \sigma_{k+1}, \dots, \sigma_m).$$

Then,

$$\begin{aligned} P(\sigma_k = +1 | \sigma_{-k}) &= \frac{P(\sigma^+)}{P(\sigma_{-k})} = \frac{P(\sigma^+)}{P(\sigma^+) + P(\sigma^-)} \\ &= \frac{e^{-\beta E(\sigma^+)}}{e^{-\beta E(\sigma^+)} + e^{-\beta E(\sigma^-)}} \\ &= \frac{1}{1 + e^{\beta[E(\sigma^+) - E(\sigma^-)]}}. \end{aligned}$$

Observe that

$$E(\sigma^+) = - \left(\sum_{\substack{i \sim j \\ i, j \not\sim k}} \sigma_i \sigma_j + \sum_{i \sim k} \sigma_i \right),$$

where the first sum is over all sites that are not neighbors of k . Also,

$$E(\sigma^-) = - \left(\sum_{\substack{i \sim j \\ i, j \not\sim k}} \sigma_i \sigma_j - \sum_{i \sim k} \sigma_i \right).$$

This gives

$$E(\sigma^+) - E(\sigma^-) = -2 \sum_{i \sim k} \sigma_i,$$

and thus

$$P(\sigma_k = +1 | \sigma_{-k}) = \frac{1}{1 + e^{\beta[E(\sigma^+) - E(\sigma^-)]}} = \frac{1}{1 + e^{-2\beta \sum_{i \sim k} \sigma_i}}.$$

Also, $P(\sigma_k = -1 | \sigma_{-k}) = 1 - P(\sigma_k = +1 | \sigma_{-k})$.

The probabilities are easily computed as they only depend on the neighbors of k . The Gibbs sampler is implemented by randomly picking sites on the grid and updating each site according to these conditional distributions.

See Figure 5.9 for simulations of the Ising model on a 60×60 grid with varying values of β . The Gibbs sampler was run for 100,000 steps for each realization. The state space has 2^{3600} elements. The simulations took less than 10 seconds on a laptop computer. The model in graph B was run at the critical inverse temperature $\beta = \ln(1 + \sqrt{2})/2$. Simulations B and C are for attractive systems, with $\beta > 0$. In D , $\beta < 0$ and the system is repelling. ■

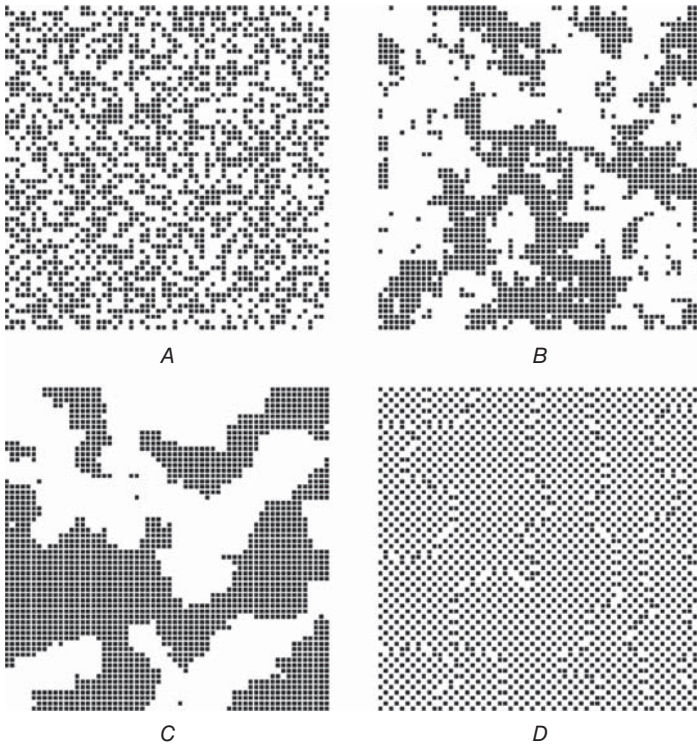


Figure 5.9 Ising simulation. Parameter values are A : $\beta = 0$, B : $\beta = 0.441$, C : $\beta = 0.75$, D : $\beta = -1.5$.

R : Ising Model

```

# ising.R
> par(mfrow=c(2,2))
> betalist <- c(0,0.441,0.75,-1.5)
> for (z in 1:4) {
+   g <- 100
+   beta <- betalist[z]
+   trials <- 100000
+   grid <- matrix(sample(c(-1,1), (g+2)^2, rep=T), nrow=g+2)
+   grid[c(1,g+2),] <- 0
+   grid[,c(1,g+2)] <- 0
+   for (m in 1:trials) {
+     i <- sample(2:(g+1), 1)
+     j <- sample(2:(g+1), 1)
+     deg <- grid[i,j+1]+grid[i,j-1]+grid[i-1,j]+grid[i+1,j]
+     p <- 1/(1 + exp(-beta*2*deg))
+     if (runif(1) < p) grid[i,j] <- 1 else grid[i,j] <- -1
+   }
+   final <- grid[2:(g+1), 2:(g+1)]
+   image(final, yaxt="n", xaxt="n", col=c(0,1)) }

```

5.4 PERFECT SAMPLING*

A central question when implementing Markov chain Monte Carlo is how long to run the chain to get close to stationarity. In the mid-1990s, an algorithm for implementing MCMC was introduced by Jim Propp and David Wilson, called *coupling from the past*. This *perfect sampling* algorithm “determines on its own when to stop, and then outputs samples in exact accordance with the desired [stationary] distribution.” (From Propp and Wilson (1996).)

The idea is disarmingly simple. Assume that an ergodic Markov chain with limiting distribution π is started in the infinite past and run forward in time. Then, at $t = 0$, after having run for an infinitely long time, the chain will have reached its stationary distribution. Thus, if one could simulate $\dots, X_{-2}, X_{-1}, X_0$, then X_0 would give a sample from π .

If a chain is in stationarity then the distribution of the current state is independent of the initial state. For a k -state chain, the algorithm proceeds by running k copies of the chain, each started from a different state, but with each chain using the same source of randomness to determine transitions. Say the chain has *coalesced* at time t if all k copies of the chain are at the same state at time t . The algorithm moves backwards from $t = -1$ until there is a time $t = s < 0$ such that each of the k copies of the chain $X_s, \dots, X_{-2}, X_{-1}, X_0$ have coalesced at $t = 0$. Then, X_0 is taken as a sample—an exact sample—from π .

To explain the perfect sampling algorithm more carefully, we represent Markov transitions with an *update function*, based on the following basic method for simulating discrete random variables.

Given a probability distribution $\mathbf{p} = (p_1, \dots, p_k)$ on (s_1, \dots, s_k) , the *inverse transform method* is a simple and intuitive way to simulate an outcome X from \mathbf{p} , using a uniform random number generator. Let U be uniformly distributed on $(0, 1)$. If $U = u$, then set $X = s_j$, where j is the smallest index such that $u < p_1 + \dots + p_j$. The method works since

$$\begin{aligned} P(X = s_j) &= P(p_1 + \dots + p_{j-1} < U < p_1 + \dots + p_j) \\ &= (p_1 + \dots + p_j) - (p_1 + \dots + p_{j-1}) \\ &= p_j. \end{aligned}$$

For example, consider the distribution $\mathbf{p} = (0.1, 0.2, 0.3, 0.4)$ on $\{a, b, c, d\}$. To simulate X from \mathbf{p} , let U be uniformly distributed on $(0, 1)$. If $U = u$, set

$$X = \begin{cases} a, & \text{if } u < 0.1, \\ b, & \text{if } 0.1 \leq u < 0.3, \\ c, & \text{if } 0.3 \leq u < 0.6, \\ d, & \text{if } 0.6 \leq u < 1. \end{cases}$$

For a Markov chain X_0, X_1, \dots , the update function value $g(x, u)$ is the outcome obtained from the conditional distribution of X_1 given $X_0 = x$ based on using the inverse transform method with $U = u$. For example, for a Markov chain with transition matrix

$$\mathbf{P} = \begin{matrix} & \begin{matrix} a & b & c & d \end{matrix} \\ \begin{matrix} a \\ b \\ c \\ d \end{matrix} & \begin{pmatrix} 1/4 & 1/4 & 1/4 & 1/4 \\ 1/3 & 1/3 & 1/3 & 0 \\ 0 & 1/3 & 1/3 & 1/3 \\ 0 & 1 & 0 & 0 \end{pmatrix} \end{matrix}, \quad (5.2)$$

the update function is

$$g(a, u) = \begin{cases} a, & \text{for } 0 \leq u < 0.25, \\ b, & \text{for } 0.25 \leq u < 0.50, \\ c, & \text{for } 0.50 \leq u < 0.75, \\ d, & \text{for } 0.75 \leq u < 1, \end{cases} \quad g(b, u) = \begin{cases} a, & \text{for } 0 \leq u < 0.33, \\ b, & \text{for } 0.33 \leq u < 0.67, \\ c, & \text{for } 0.67 \leq u < 1, \end{cases}$$

$$g(c, u) = \begin{cases} b, & \text{for } 0 \leq u < 0.33, \\ c, & \text{for } 0.33 \leq u < 0.67, \\ d, & \text{for } 0.67 \leq u < 1, \end{cases} \quad \text{and} \quad g(d, u) = b, \text{ for } 0 \leq u < 1.$$

For a general Markov chain, we can represent the chain recursively as $X_{n+1} = g(X_n, U_n)$, for $n = 0, 1, \dots$, where g is an update function, and U_0, U_1, \dots , is an i.i.d. sequence of uniform (0,1) random variables.

For an ergodic Markov chain on k states, coupling from the past works as follows:

1. Let U_{-1}, U_{-2}, \dots be an i.i.d. sequence of uniform (0, 1) random variables.
2. Let $t = -1$. Run k copies of the chain from time t , each started from a different state. For the chain started in x , set $X_0 = g(x, U_{-1})$. Thus, k copies of X_0 will be generated, one from each state. The key point of the algorithm is that even though k different chains are generated, they all use the *same* source of randomness U_{-1} . If all the X_0 agree, the chains have coalesced, the algorithm stops, and the common X_0 is output as a sample from π .
3. If the chains have not coalesced, move back in time to $t = -2$. Again, run k copies of the chain each from different starting states. If $X_{-2} = x$, generate $X_{-1} = g(x, U_{-2})$, and $X_0 = g(X_{-1}, U_{-1})$. Thus,

$$X_0 = g(g(x, U_{-2}), U_{-1}).$$

A new random variable U_{-2} is used in the simulation, but the old value of U_{-1} from the previous step is retained. Again, if the chains coalesce at $t = 0$ return the common value of X_0 as a sample from π .

4. The algorithm proceeds by iteratively moving back in time. Propp and Wilson suggest, for efficiency, to double the number of steps at each iteration. Thus, for the next iteration, start at $t = -4$, then $t = -8$, and so on. At $t = -4$, uniform variables U_{-4} and U_{-3} are generated, while retaining U_{-2} and U_{-1} from the previous steps.

See Figure 5.10 for an illustration of the algorithm for the transition matrix \mathbf{P} of Equation (5.2). The stationary distribution is $\pi = (2/11, 9/22, 3/11, 3/22)$.

At the first iteration, transitions from each state are based on $U_{-1} = 0.394$. This gives $g(a, u_{-1}) = b$, $g(b, u_{-1}) = b$, $g(c, u_{-1}) = c$, and $g(d, u_{-1}) = b$. No coalescence occurs.

At the second iteration, the chain is started from $t = -2$, with $U_{-2} = 0.741$. Keeping the previous value of U_{-1} , the chain moves forward as shown. Again, coalescence has not occurred by $t = 0$.

At the third iteration, the chain is run from $t = -4$. Two new uniform variables are generated $U_{-3} = 0.0407$ and $U_{-4} = 0.123$. Even though coalescence occurs

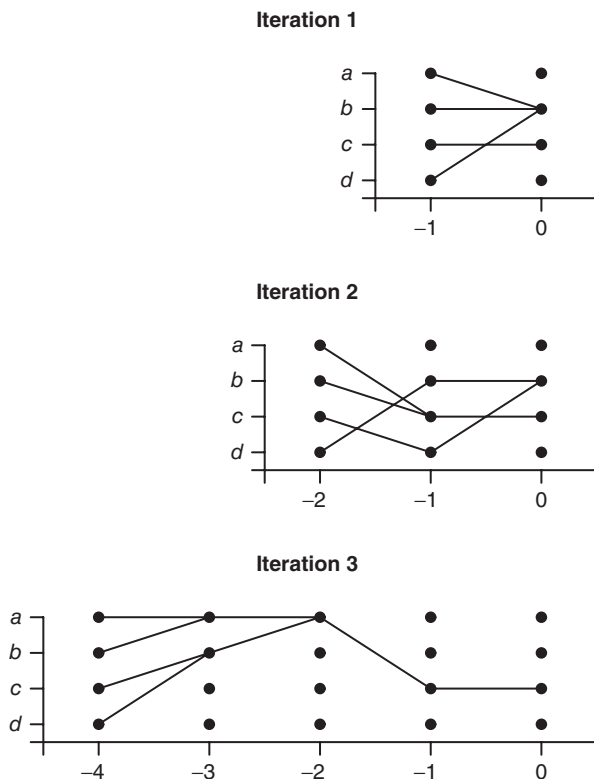


Figure 5.10 Perfect sampling on a 4-state chain. Coalescence occurs for the chain started at time $t = -4$. The algorithm outputs state c as a sample from the stationary distribution.

at $t = -2$, the algorithm outputs state c at $t = 0$ as a sample from π . See the R file **perfect.R** for the implementation.

Monotonicity and the Ising Model

For Markov chains with large state spaces, running copies of the chain from each state is not practical. However, for chains where the state space exhibits a certain *monotonicity* property, the efficiency of the algorithm can be dramatically improved.

Say a Markov chain with update function g is *monotone* if the state space can be ordered in such a way that if $x \leq y$, then $g(x, u) \leq g(y, u)$, for $0 < u < 1$. For example, the chain on $\{1, 2, 3\}$ with transition matrix

$$P = \begin{matrix} & \begin{matrix} 1 & 2 & 3 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \end{matrix} & \begin{pmatrix} 1/2 & 1/3 & 1/6 \\ 3/8 & 3/8 & 1/4 \\ 1/4 & 1/2 & 1/4 \end{pmatrix} \end{matrix}$$

is monotone. We invite the reader to check this claim with the update function

$$g(1, u) = \begin{cases} 1, & \text{for } 0 \leq u < 0.50, \\ 2, & \text{for } 0.50 \leq u < 0.83, \\ 3, & \text{for } 0.83 \leq u < 1, \end{cases} \quad g(2, u) = \begin{cases} 1, & \text{for } 0 \leq u < 0.375, \\ 2, & \text{for } 0.375 \leq u < 0.75, \\ 3, & \text{for } 0.75 \leq u < 1, \end{cases}$$

and

$$g(3, u) = \begin{cases} 1, & \text{for } 0 \leq u < 0.25, \\ 2, & \text{for } 0.25 \leq u < 0.75, \\ 3, & \text{for } 0.75 \leq u < 1. \end{cases}$$

Assume that there exists a minimal state m and a maximal state M such that for all x in the state space, $m \leq x \leq M$. Then, a monotone chain has the property that the path of every chain is sandwiched between the paths of the two chains started in m and M , respectively. An example is shown in Figure 5.11.

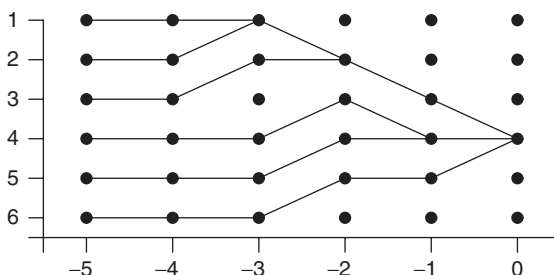


Figure 5.11 Monotone chain. All paths are sandwiched between the maximal and minimal paths.

When implementing coupling from the past for a monotone Markov chain one need only keep track of the chains started from m and M , since if those two chains coalesce at a state x , then all other paths of the chain are sandwiched between the maximal and minimal states and thus also coalesce at x .

The Ising model Gibbs sampler is a chain with an exponentially large state space. For an $n \times n$ grid there are 2^{n^2} configurations. Running chains from each state is practically impossible. However, the Ising model is monotone for $\beta > 0$ (the attractive case), and thus admits a remarkably efficient implementation of coupling from the past.

Given configurations σ and τ , say that $\sigma \leq \tau$, if $\sigma_v \leq \tau_v$ for all sites v . This defines a *partial ordering* on the set of all configurations. The configuration σ^m , in which all sites are -1 , is the minimal configuration in this ordering, and the configuration σ^M , in which all sites are $+1$, is the maximal configuration.

To implement the Propp–Wilson algorithm on the Ising model, it suffices at each step to run the chain from σ^m and σ^M , checking whether or not coalescence occurs.

To show the Ising chain is monotone, assume that $\sigma \leq \tau$. Then, for any site v ,

$$\sum_{i \sim v} \sigma_i \leq \sum_{i \sim v} \tau_i.$$

Updates occur one site at a time. For site v , let X denote the spin at v . For $\beta > 0$,

$$P(X = +1 | \sigma_{-v}) = \frac{1}{1 + e^{-2\beta \sum_{i \sim v} \sigma_i}} \leq \frac{1}{1 + e^{-2\beta \sum_{i \sim v} \tau_i}} = P(X = +1 | \tau_{-v}),$$

which gives the result. See Figure 5.12 for a perfect sample of the Ising model at the critical temperature.

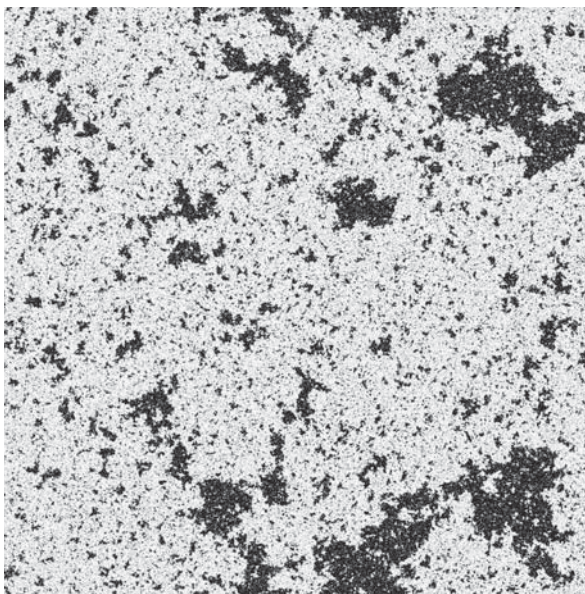


Figure 5.12 An exact sample of the Ising model on a 4200×4200 grid at the critical temperature $1/\beta = 2.269$. *Source:* Propp and Wilson (1996). Reproduced with permission of John Wiley and Sons, Inc.

5.5 RATE OF CONVERGENCE: THE EIGENVALUE CONNECTION*

Perfect sampling is an impressive algorithm, but limited in its use. For all but the simplest chains, it is impractical to run simultaneous copies of the chain from each starting state. And most Markov chains are not monotone. Thus, for all practical purposes users of MCMC must confront the issue of how long to run the chain in order to reach convergence, or near convergence, to the targeted stationary distribution.

In this section, we show that the second largest eigenvalue of the transition matrix is a lead actor in the story.

Assume a finite, reversible ergodic Markov chain, with transition matrix \mathbf{P} and limiting distribution $\boldsymbol{\pi} = (\pi_1, \dots, \pi_k)$. Let

$$\mathbf{Q} = \begin{pmatrix} \sqrt{\pi_1} & 0 & \cdots & 0 \\ 0 & \sqrt{\pi_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sqrt{\pi_k} \end{pmatrix}.$$

Since $\boldsymbol{\pi}$ is positive, the matrix is invertible, and

$$\mathbf{Q}^{-1} = \begin{pmatrix} 1/\sqrt{\pi_1} & 0 & \cdots & 0 \\ 0 & 1/\sqrt{\pi_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1/\sqrt{\pi_k} \end{pmatrix}.$$

Let $\mathbf{A} = \mathbf{Q}\mathbf{P}\mathbf{Q}^{-1}$. Then,

$$A_{ij} = \sum_{r=1}^k \sum_{s=1}^k Q_{ir} P_{rs} Q_{sj}^{-1} = Q_{ii} P_{ij} Q_{jj}^{-1} = \sqrt{\pi_i} P_{ij} \frac{1}{\sqrt{\pi_j}}.$$

Since the chain is reversible,

$$A_{ij} = \sqrt{\pi_i} P_{ij} \frac{1}{\sqrt{\pi_j}} = \frac{\pi_i P_{ij}}{\sqrt{\pi_i} \sqrt{\pi_j}} = \frac{\pi_j P_{ji}}{\sqrt{\pi_i} \sqrt{\pi_j}} = \sqrt{\pi_j} P_{ji} \frac{1}{\sqrt{\pi_i}} = A_{ji}.$$

Thus, \mathbf{A} is symmetric. That is, $\mathbf{A} = \mathbf{A}^T$. The eigenvalues of a symmetric matrix are real. Furthermore, a symmetric matrix is orthogonally diagonalizable. That is, we can write $\mathbf{A} = \mathbf{S}\mathbf{D}\mathbf{S}^T$, where \mathbf{D} is a diagonal matrix whose diagonal entries are the eigenvalues of \mathbf{A} , and \mathbf{S} is a matrix whose columns are an orthonormal set of eigenvectors corresponding to those eigenvalues.

The eigenvalues of a Markov transition matrix are described by Lemma 3.15 in Section 3.10. Since the chain is ergodic, the transition matrix is regular, and there is a single largest eigenvalue in absolute value $\lambda_1 = 1$. As the eigenvalues are real, they can be written in decreasing order

$$1 = \lambda_1 > \lambda_2 \geq \lambda_3 \geq \cdots \geq \lambda_k > -1.$$

Since $\mathbf{A} = \mathbf{Q}\mathbf{P}\mathbf{Q}^{-1}$, the matrices \mathbf{P} and \mathbf{A} are similar and thus have the same eigenvalues. This gives

$$\mathbf{P} = \mathbf{Q}^{-1}\mathbf{A}\mathbf{Q} = \mathbf{Q}^{-1}(\mathbf{S}\mathbf{D}\mathbf{S}^T)\mathbf{Q} = (\mathbf{Q}^{-1}\mathbf{S})\mathbf{D}(\mathbf{S}^T\mathbf{Q}),$$

where

$$\mathbf{D} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_k \end{pmatrix}.$$

For $n \geq 0$, $\mathbf{P}^n = (\mathbf{Q}^{-1}\mathbf{S})\mathbf{D}^n(\mathbf{S}^T\mathbf{Q})$. Taking the ij th entry,

$$\begin{aligned} P_{ij}^n &= \sum_{t=1}^k \sum_{u=1}^k (\mathbf{Q}^{-1}\mathbf{S})_{it} D_{tu}^n (\mathbf{S}^T\mathbf{Q})_{uj} \\ &= \sum_{t=1}^k (\mathbf{Q}^{-1}\mathbf{S})_{it} \lambda_t^n (\mathbf{S}^T\mathbf{Q})_{tj} \\ &= \frac{\sqrt{\pi_j}}{\sqrt{\pi_i}} \sum_{t=1}^k \lambda_t^n S_{it} S_{jt} \\ &= \frac{\sqrt{\pi_j}}{\sqrt{\pi_i}} S_{i1} S_{j1} + \frac{\sqrt{\pi_j}}{\sqrt{\pi_i}} \sum_{t=2}^k \lambda_t^n S_{it} S_{jt}. \end{aligned} \quad (5.3)$$

Equation (5.3) is known as the *spectral representation formula* for the n -step transition probabilities. Since $P_{ij}^n \rightarrow \pi_j$, as $n \rightarrow \infty$, and $\lambda_t^n \rightarrow 0$, as $n \rightarrow \infty$, for $2 \leq t \leq k$, we have that

$$\pi_j = \frac{\sqrt{\pi_j}}{\sqrt{\pi_i}} S_{i1} S_{j1},$$

and

$$P_{ij}^n - \pi_j = \frac{\sqrt{\pi_j}}{\sqrt{\pi_i}} \sum_{t=2}^k \lambda_t^n S_{it} S_{jt}.$$

To bound the difference between the n -step and stationary probabilities,

$$|P_{ij}^n - \pi_j| \leq \frac{\sqrt{\pi_j}}{\sqrt{\pi_i}} \sum_{t=2}^k |\lambda_t^n S_{it} S_{jt}| \leq \frac{\sqrt{\pi_j}}{\sqrt{\pi_i}} |\lambda_*|^n \sum_{t=2}^k |S_{it} S_{jt}| = |\lambda_*|^n c_{ij}, \quad (5.4)$$

where $|\lambda_*| = \max_{2 \leq t \leq k} |\lambda_t|$, and c_{ij} is a constant that does not depend on n . The quantity λ_* is the second largest eigenvalue in absolute value. The bounds in Equation (5.4) show that the convergence to stationarity is exponentially fast and the rate of convergence is dominated by the second largest eigenvalue.

5.6 CARD SHUFFLING AND TOTAL VARIATION DISTANCE*

The asymptotic rate of convergence only tells part of the story of a Markov chain's path to stationarity. In many applications where a Markov chain is run until it is close to its limiting distribution, such as MCMC, what a user is interested in is not an

asymptotic result about what happens *at infinity*, but rather the more practical issue of how many steps of the chain are *close enough*. The issue is illustrated by card shuffling.

Many card shuffling schemes can be modeled as Markov chains. For a deck of k cards, the state space is the set of $k!$ permutations (orderings) of the deck. A *shuffle* is one step of the chain. If repeated shuffles eventually mix up the deck, then the card shuffling scheme is modeled as an ergodic Markov chain whose limiting distribution is uniform on the set of permutations. If σ is a permutation, then $\pi_\sigma = 1/k!$.

Theoretically, the uniform distribution is achieved as the number of shuffles tends to infinity. This is not much help to a card player, however, who wants to know how many shuffles are needed to bring the deck sufficiently close to random.

To quantify the idea of *how far* a Markov chain is from its limiting distribution, a measure of distance is needed between the distribution of the chain at a fixed time and the limiting distribution. A common measure is *total variation distance*.


Total Variation Distance

Let P be the transition matrix of an ergodic Markov chain with limiting distribution π . The *total variation distance at time n* is

$$v(n) = \max_{i \in S} \max_{A \subseteq S} |P(X_n \in A | X_0 = i) - \pi_A|.$$

In words, total variation distance at time n is the maximum absolute difference over all events A and all starting states between the probabilities $P(X_n \in A)$ and π_A . The distance measure takes values between 0 and 1. If $v(n) = 0$, then the chain is in stationarity at time n . Over time, $v(n) \rightarrow 0$, as $n \rightarrow \infty$. It can be shown that the definition is equivalent to

$$v(n) = \max_i \frac{1}{2} \sum_j |P_{ij}^n - \pi_j|.$$

 **Example 5.9** Compute total variation distance for the two-state Markov chain with transition matrix

$$P = \frac{1}{2} \begin{pmatrix} 1-p & p \\ q & 1-q \end{pmatrix}.$$

Solution Recall that

$$P^n = \frac{1}{p+q} \begin{pmatrix} q + p(1-p-q)^n & p - p(1-p-q)^n \\ q - q(1-p-q)^n & p + q(1-p-q)^n \end{pmatrix}$$

and

$$\pi = \left(\frac{q}{p+q}, \frac{p}{p+q} \right).$$

For $i = 1$,

$$\sum_{j=1}^2 |P_{1j}^n - \pi_j| = \frac{2p}{p+q}(1-p-q)^n.$$

For $i = 2$,

$$\sum_{j=1}^2 |P_{2j}^n - \pi_j| = \frac{2q}{p+q}(1-p-q)^n.$$

Thus,

$$v(n) = \frac{\max(p, q)}{p+q}(1-p-q)^n.$$

Convergence to stationarity, as measured by total variation distance, occurs exponentially fast, with the rate of convergence governed by $1-p-q$. Note that the second largest eigenvalue of \mathbf{P} is $\lambda_2 = 1-p-q$. ■

Top-to-Random Shuffle

For the *top-to-random* shuffling scheme, a shuffle consists of placing the card at the top of the deck into a uniformly random position in the deck. How many such shuffles does it take to mix up a deck of cards?

Admittedly this is not a very efficient shuffling method for mixing up cards. But it is a good example to illustrate our analysis. We show, in a fairly precise sense, that for a k -card deck it takes about $k \ln k$ such shuffles to mix up the deck. For a 52-card deck, that is $52 \ln 52$, which is about 200 shuffles.

Central to the analysis is the notion of a strong stationary time. Recall the discussion of stopping times and the strong Markov property in Section 3.9. A strong stationary time is a stopping time. When a Markov chain stops at a strong stationary time it is in its stationary distribution.

Strong Stationary Time

A *strong stationary time* for an ergodic Markov chain X_0, X_1, \dots is a stopping time T such that

$$P(X_n = j, T = n) = \pi_j P(T = n), \text{ for all states } j \text{ and } n \geq 0.$$

The definition says that if T is a strong stationary time and $T = n$, then the Markov chain is in stationarity at time n . Furthermore, the state of the chain at time T is independent of T .

In Aldous and Diaconis (1986), the top-to-random shuffle is analyzed by constructing a suitable strong stationary time.

Starting with the deck in a fixed order, let b denote the card initially at the bottom of the deck. As the deck is shuffled, cards will eventually be inserted below b , and the position of b in the deck will rise. We show that as soon as b reaches the top of the deck and then is inserted in a random position, the deck will be mixed up in the sense that every ordering is equally likely.

Initially, the probability that the top card is inserted at the bottom of the deck below b is $1/k$, since there are k available positions. Since successive shuffles are independent, the number of shuffles needed for a card to be inserted below b has a geometric distribution with parameter $1/k$.

After b has moved up one position from the bottom, consider the time until a second card moves below b . There are now two available slots, and the number of shuffles needed has a geometric distribution with parameter $2/k$. Furthermore, once two cards are below b , each of the two orderings of those two cards is equally likely.

More generally, assume that there are $i - 1$ cards below b . The probability that the top card is inserted below b is i/k . The number of shuffles needed before the top card is inserted below b has a geometric distribution with parameter i/k . Once i cards are below b , each of the $i!$ orderings of the i cards is equally likely.

Finally, after the original bottom card has risen to the top of the deck it takes one more shuffle, which sends b to a uniformly random position, before all $k!$ orderings of the deck are equally likely. At that (random) time, the deck is mixed up.

Let T be the number of top-to-random shuffles needed for the original bottom card to reach the top of the deck, plus 1. Then, T is a strong stationary time.

The analysis also shows that T can be expressed as the sum of k independent geometric random variables, with successive parameters $1/k, 2/k, \dots, (k-1)/k, 1$. The expectation of a geometric random variable is the reciprocal of the parameter. Hence,

$$E(T) = \frac{k}{1} + \frac{k}{2} + \cdots + \frac{k}{k-1} + \frac{k}{k} = k \left(1 + \frac{1}{2} + \cdots + \frac{1}{k-1} + \frac{1}{k} \right) \approx k \ln k.$$

Strong stationary times are used to bound total variation distance by the following lemma.

Strong Stationary Times and Total Variation Distance

Lemma 5.2. *Let T be a strong stationary time for an ergodic Markov chain. For $n > 0$,*

$$v(n) \leq P(T > n).$$

The lemma is proved at the end of the chapter. To apply the lemma to the top-to-random shuffle, we estimate $P(T > n)$ by relating the strong stationary time T to the classic coupon collector's problem.

Assume that a set of k distinct items (coupons) is sampled repeatedly with replacement. Let C be the number of samples needed to obtain a full set of coupons, that is,

for each distinct item to be sampled at least once. We show that the distributions of C and T are the same.

Let C_i be the number of trials needed before the i th new coupon has been selected, for $i = 1, \dots, k$, so that $C = C_k$. Necessarily, $C_1 = 1$. For $i = 2, \dots, k$, $C_i - C_{i-1}$, the number of samples needed after the $(i-1)$ th coupon is picked until the i th new coupon is picked has a geometric distribution with parameter $(k-i+1)/k$. This is because after the $(i-1)$ th coupon is selected, there are $k-i+1$ remaining coupons, and each one can be chosen with probability $(k-i+1)/k$. Furthermore, the random variables $C_i - C_{i-1}$ are independent. We have that

$$C = C_1 + (C_2 - C_1) + \dots + (C_k - C_{k-1})$$

is a sum of independent geometric random variables with respective parameters $1, (k-1)/k, \dots, 2/k, 1/k$. It follows that C and T have the same distribution, and $P(T > n) = P(C > n)$, for all n .

The event $\{C > n\}$ is the event that after n trials, some coupon has not been sampled. Let A_i denote the event that coupon i has not been picked after n trials. Then,

$$\begin{aligned} P(C > n) &= P\left(\bigcup_{i=1}^n A_i\right) \leq \sum_{i=1}^k P(A_i) \\ &= \sum_{i=1}^k \left(1 - \frac{1}{k}\right)^n \leq \sum_{i=1}^k e^{-n/k} \\ &= ke^{-n/k}, \end{aligned}$$

using that $1 - x \leq e^{-x}$, for all x .

For the top-to-random shuffle, with Lemma 5.2, this gives

$$v(n) \leq P(T > n) = P(C > n) \leq ke^{-n/k}. \quad (5.5)$$

To make total variation distance small, find a value of n that makes the righthand side of Equation (5.5) small. For $c > 0$, let $n = k \ln k + ck$. Then,

$$v(n) = v(k \ln k + ck) \leq ke^{-(\ln k + c)} = e^{-c} \approx 0.$$

The bound is interpreted to mean that for a *little more* than $k \ln k$ shuffles the distance to stationarity is small.

Aldous and Diaconis (1986) also show, by a different analysis, that for a *little less* than $k \ln k$ shuffles, that is, for $n = k \ln k - ck$, with $c > 0$, total variation distance $v(n)$ stays close to 1.

Thus, the claim that for large k it takes about $k \ln k$ top-to-random shuffles to mix up the deck of cards. Fewer than $k \ln k$ steps are insufficient. More than $k \ln k$ steps are not necessary.

The top-to-random shuffle exhibits a remarkable property, shared by many card shuffling Markov chains, known as the *cutoff phenomenon*. The approach to stationarity, as measured by total variation distance, behaves like a phase transition in physics. As the number of shuffles increases, the chain stays far from uniform and then, in a relatively short time window, the chain rapidly gets close to uniform. See Figure 5.13.

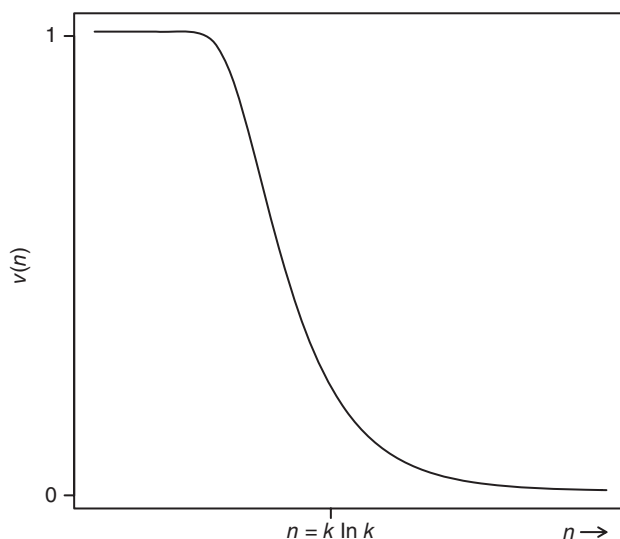


Figure 5.13 Cutoff phenomenon for top-to-random shuffle.

Seven Shuffles is Enough

In 1990, the *New York Times* reported the “fascinating result,” discovered by mathematicians David Bayer and Persi Diaconis, that, “It takes just seven ordinary, imperfect [riffle] shuffles to mix a deck of cards thoroughly” (Kolata, 1990). The analysis by Bayer and Diaconis (1992) was based on calculating the total variation distance for a Markov model for how most people shuffle cards.

The riffle-shuffling scheme is known as the Gilbert–Shannon–Reeds model. A deck of k cards is cut into two piles according to a binomial distribution. That is, the probability that one pile has i cards, and the other pile has $k - i$ cards, is $\binom{k}{i} 2^{-k}$, for $i = 0, 1, \dots, k$. The two piles are then riffled together by successively dropping cards from either pile with probability proportional to the size of the pile. Thus, if there are a cards in the first pile, and b cards in the second pile, the probability that the next card dropped comes from the first pile is $a/(a + b)$.

Bayer and Diaconis give a thorough analysis of the total variation distance for the resulting Markov chain. They show that about $(3/2)\log_2(k)$ shuffles are necessary and sufficient to make total variation distance small. They further demonstrate the existence of the cutoff phenomenon, as shown in Figure 5.14 and Table 5.2, which contains values for total variation distance after n riffle shuffles of 52 cards.

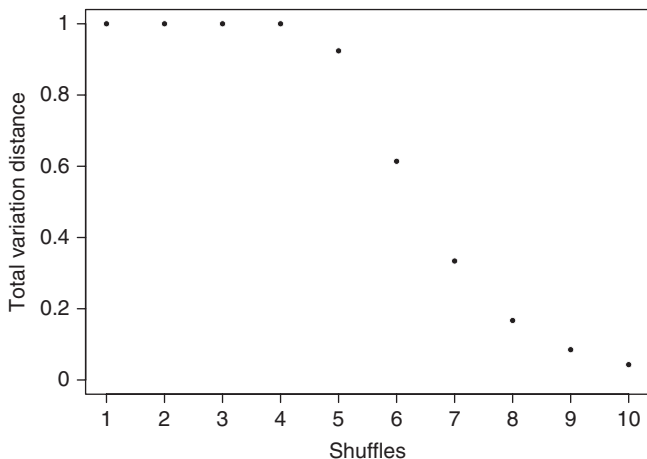


Figure 5.14 Total variation distance for riffle shuffling.

TABLE 5.2 Total Variation Distance for n Shuffles of 52 Cards

n	1	2	3	4	5	6	7	8	9	10
$v(n)$	1.000	1.000	1.000	1.000	0.924	0.614	0.334	0.167	0.085	0.043

Source: Data from Bayer and Diaconis (1992).

For an introduction to the mathematics of riffle shuffling, see Mann (1994).

Proof of Lemma 5.2

Let T be a strong stationary time for a Markov chain X_0, X_1, \dots . We show that the definition of strong stationary time implies that

$$P(X_n = j, T \leq n) = \pi_j P(T \leq n).$$

We have that

$$\begin{aligned}
 P(X_n = j, T \leq n) &= \sum_{m \leq n} P(T = m | X_n = j) P(X_n = j) \\
 &= \sum_{m \leq n} \sum_x P(T = m | X_n = j, X_m = x) P(X_m = x | X_n = j) P(X_n = j) \\
 &= \sum_{m \leq n} \sum_x P(T = m | X_m = x) P(X_m = x, X_n = j) \\
 &= \sum_{m \leq n} P(T = m) \sum_x \pi_x \frac{P(X_m = x, X_n = j)}{P(X_m = x)}
 \end{aligned}$$

$$\begin{aligned}
&= P(T \leq m) \sum_x \pi_x P(X_n = j | X_m = x) \\
&= P(T \leq m) \sum_x \pi_x P_{xj}^{n-m} = P(T \leq m) \pi_j.
\end{aligned}$$

The third equality is because T is a stopping time and the event $\{T = m\}$ depends on X_0, \dots, X_m , and not on X_n for $n > m$. The fourth equality is by the definition of strong stationary time. The last equality is because π is a stationary distribution and $\pi P^{n-m} = \pi$.

For the chain started in i ,

$$\begin{aligned}
P(X_n \in A) &= P(X_n \in A, T \leq n) + P(X_n \in A, T > n) \\
&= \pi_A P(T \leq n) + P(X_n \in A | T > n) P(T > n) \\
&= \pi_A + [P(X_n \in A | T > n) - \pi_A] P(T > n),
\end{aligned}$$

which gives

$$|P(X_n \in A) - \pi_A| = |P(X_n \in A | T > n) - \pi_A| P(T > n) \leq P(T > n),$$

where the last inequality is because the absolute difference of two probabilities is at most 1. Maximizing over initial states i gives the result. ■

EXERCISES

- 5.1** Four out of every five trucks on the highway are followed by a car, while only one out of every four cars is followed by a truck. At a toll booth, cars pay \$1.50 and trucks pay \$5.00. If 1,000 vehicles pass through the tollbooth in one day, how much toll is collected?
- 5.2** Consider simple random walk on $\{0, 1, \dots, k\}$ with reflecting boundaries at 0 and k , that is, random walk on the path from 0 to k . A random walker earns \$ k every time the walk reaches 0 or k , but loses \$1 at each internal vertex (from 1 to $k - 1$). In 10,000 steps of the walk, how much, on average, will be gained?
- 5.3** Commuters in an urban center either drive by car, take the bus, or bike to work. Based on recent changes to local transportation systems, urban planners predict yearly annual behavior changes of commuters based on a Markov model with transition matrix

$$P = \begin{matrix} & \begin{matrix} \text{Car} & \text{Bus} & \text{Bike} \end{matrix} \\ \begin{matrix} \text{Car} \\ \text{Bus} \\ \text{Bike} \end{matrix} & \begin{pmatrix} 0.7 & 0.2 & 0.1 \\ 0.1 & 0.8 & 0.1 \\ 0.05 & 0.15 & 0.8 \end{pmatrix} \end{matrix}.$$

- (a) In a city of 10,000 commuters, 6,000 currently drive cars, 3,000 take the bus, and 1,000 bike to work. Two years after transportation changes are made, how many commuters will use each type of transportation? Over the long term, how many commuters will use each type of transportation?
- (b) The European Cyclists Federation reports the following levels of CO₂ emissions (in grams) per passenger per kilometer traveled: car: 271, bus: 101, bike: 21. What is the current average amount of CO₂ emissions per kilometer traveled? How does the average change over the long-term?

5.4 The Fibonacci sequence 1, 1, 2, 3, 5, 8, 13, ... is described by the recurrence $f_n = f_{n-1} + f_{n-2}$, for $n \geq 3$, with $f_1 = f_2 = 1$.

- (a) See the MCMC example for binary sequences with no adjacent 1s. Show that the number of binary sequences of length m with no adjacent 1s is f_{m+2} .
- (b) Let $p_{k,m}$ be the number of good sequences of length m with exactly k 1s. Show that

$$p_{k,m} = \binom{m-k+1}{k}, \text{ for } k = 0, 1, \dots, \lceil m/2 \rceil.$$

- (c) Let μ_m be the expected number of 1s in a good sequence of length m under the uniform distribution. Find μ_m , for $m = 10, 100, 1000$.
- (d) If you have access to a mathematical symbolic software system, such as *Mathematica*, use it to find

$$\lim_{m \rightarrow \infty} \frac{\mu_m}{m}.$$

5.5 Exhibit a Metropolis–Hastings algorithm to sample from the distribution

1	2	3	4	5	6
0.01	0.39	0.11	0.18	0.26	0.05

with proposal distribution based on one fair die roll.

- 5.6** Show how to generate a Poisson random variable with parameter λ using Metropolis–Hastings. Use simple symmetric random walk as the proposal distribution.
- 5.7** Exhibit a Metropolis–Hastings algorithm to sample from a binomial distribution with parameters n and p . Use a proposal distribution that is uniform on $\{0, 1, \dots, n\}$.
- 5.8** The Metropolis–Hastings algorithm is used to simulate a binomial random variable with parameters $n = 4$ and $p = 1/4$. The proposal distribution is simple symmetric random walk on $\{0, 1, 2, 3, 4\}$ with reflecting boundaries.
- (a) Exhibit the T matrix.
- (b) Exhibit the transition matrix for the Markov chain created by the Metropolis–Hastings algorithm.

(c) Show explicitly that this transition matrix gives a reversible Markov chain whose stationary distribution is the desired binomial distribution.

5.9 Show how to use the Metropolis–Hastings algorithm to simulate from the *double exponential distribution*, with density

$$f(x) = \frac{\lambda}{2} e^{-\lambda|x|}, \text{ for } -\infty < x < \infty.$$

Use the normal distribution as a proposal distribution.

5.10 A Markov chain has transition matrix

$$P = \begin{matrix} & \begin{matrix} 1 & 2 & 3 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \end{matrix} & \begin{pmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1/2 \\ 0 & 1/2 & 1/2 \end{pmatrix} \end{matrix}$$

Show that the chain is monotone.

5.11 Consider the following random walk on $\{1, \dots, n\}$. From state 1, the walk moves to 1 or 2 with probability $1/2$ each. From state n , the walk moves to n or $n-1$ with probability $1/2$ each. From all other states i , the walk moves to $i-1$ or $i+1$ with probability $1/2$ each.

(a) Show that the Markov chain is monotone.

(b) R : Implement perfect sampling, making use of monotonicity, to simulate the random walk on $\{1, \dots, 100\}$.

5.12 Consider random walk on the k -hypercube graph. At each step, one of k coordinates is selected uniformly at random. A coin is then tossed. If heads, the coordinate is set to 0. If tails, it is set to 1. Let T be the first time that all k coordinates are selected. Argue that T is a strong stationary time. What can be said on how many steps it takes for the walk to get uniformly distributed? (*Hint*: consider the coupon collector's problem.)

5.13 Consider the lazy librarian and move-to-front process as described in Example 2.10. It can be shown that the first time that all the books have been selected is a strong stationary time. Assume that p_j is the probability that book j is selected, for $j = 1, \dots, k$. Show that

$$v(n) \leq \sum_{j=1}^k e^{-p_j n}.$$

5.14 Show that the two definitions of total variation distance given in Section 5.6 are equivalent.

5.15 See the description of the Gilbert–Shannon–Reeds model for riffle shuffles in Section 5.6. Give the Markov transition matrix for a three-card deck.

- 5.16** See the discussion on riffle shuffling and total variation distance. Bayer and Diaconis (1992) prove the following. If k cards are riffle shuffled n times with $n = (3/2)\log_2(k) + \theta$, then for large k ,

$$v(n) = 1 - 2\Phi\left(\frac{-2^{-\theta}}{4\sqrt{3}}\right) + g(k), \quad (5.6)$$

where Φ is the cumulative distribution function of the standard normal distribution, and $g(k)$ is a slow-growing function of order $k^{-1/4}$.

- (a) Show that total variation distance tends to 1 with θ small, and to 0 with θ large.
- (b) Verify that Equation (5.6) gives values that are close to the entries in Table 5.2.
- 5.17** R : Random variables X and N have joint distribution, defined up to a constant of proportionality,

$$f(x, n) \propto \frac{e^{-3x} x^n}{n!}, \text{ for } n = 0, 1, 2, \dots \text{ and } x > 0.$$

Note that X is continuous and N is discrete.

- (a) Implement a Gibbs sampler to sample from this distribution.
- (b) Use your simulation to estimate (i) $P(X^2 < N)$ and (ii) $E(XN)$.
- 5.18** R : A random variable X has density function, defined up to proportionality,

$$f(x) \propto e^{-(x-1)^2/2} + e^{-(x-4)^2/2}, \text{ for } 0 < x < 5.$$

Implement a Metropolis–Hastings algorithm for simulating from the distribution. Use your algorithm to approximate the mean and variance of X .

- 5.19** R : Implement the algorithm in Exercise 5.7 for $n = 50$ and $p = 1/4$. Use your simulation to estimate $P(10 \leq X \leq 15)$, where X has the given binomial distribution. Compare your estimate to the exact probability.
- 5.20** R : Consider a Poisson distribution with parameter $\lambda = 3$ conditioned to be nonzero. Implement an MCMC algorithm to simulate from this distribution, using a proposal distribution that is geometric with parameter $p = 1/3$. Use your simulation to estimate the mean and variance.