An Application of Markov Chain Monte Carlo to Community Ecology
Author(s): George W. Cobb and Yung-Pin Chen
Reviewed work(s):
Published by: Mathematical Association of America
Stable URL: http://www.jstor.org/stable/3647877
Accessed: 06/11/2012 11:29

Your use of the JSTOR archive indicates your acceptance of the Terms & Conditions of Use, available at
http://www.jstor.org/page/info/about/policies/terms.jsp

JSTOR is a not-for-profit service that helps scholars, researchers, and students discover, use, and build upon a wide range of
content in a trusted digital archive. We use information technology and tools to increase productivity and facilitate new forms
of scholarship. For more information about JSTOR, please contact support@jstor.org.
An Application of Markov Chain Monte Carlo to Community Ecology

George W. Cobb and Yung-Pin Chen

1. INTRODUCTION.

**Act 1.** In 1975 Jared Diamond, who has since become famous as the Pulitzer Prize-winning author of *Guns, Germs, and Steel*, published a paper based mainly on a big matrix of 0s and 1s [4]. Each row of Diamond’s matrix\(^1\) corresponded to a bird species; each column corresponded to an island in the New Hebrides (now Vanuatu). The 1s and 0s recorded the presence or absence of the species on the islands. In his paper, Diamond proposed a set of seven “community assembly rules” that he had inferred from the patterns in his co-occurrence matrix. These rules, said Diamond, govern the way species organize themselves into communities.

**Act 2.** In 1979, Connor and Simberloff published a stunningly blunt rebuttal [3] asserting that “every assembly rule is either a tautological consequence of the definitions employed, a trivial logical deduction from the stated circumstances, or a pattern which would largely be expected were species distributed randomly . . . .”

In the twenty years since Connor and Simberloff fired that broadside, ecologists have continued to argue, and in the process have continued to seek a clear understanding of what patterns to expect if species were to distribute themselves randomly. The purpose of this paper is to describe some mathematical ideas related to that quest. In particular, we use an example based on Darwin’s study of finches to introduce the method of Markov chain Monte Carlo (MCMC).

Although much of this paper is expository, there are four original results, Propositions 4 and 5, and the two theorems, that represent joint work by students and faculty from a summer program, Research Experiences for Undergraduates, held in 2001 at Mount Holyoke College. The participants were Valentin Burlacu (Amherst College), Yung-Pin Chen (faculty, Lewis & Clark College), George W. Cobb (faculty, Mount Holyoke College), Rebecca Horowitz (Hampshire College), Lin Lin (Mount Holyoke College), Ana Mocanu (Amherst College), Jane Ni (Mount Holyoke College), Timothy Teräväinen (New College, Florida), and Man Yu Yum (Mount Holyoke College).

Markov chain Monte Carlo is a computer simulation method with broad applicability in physics, astronomy, molecular biology, and statistics, where MCMC has had an especially profound impact by making Bayesian inference workable in ways once considered out of reach. (For a lovely exposition of MCMC, along with other simulation methods, and an extraordinary tour of their applications in the sciences, we enthusiastically recommend a new book by Jun Liu [11].) To set the stage, we first present a generic form of “the ecologists’ question,” which will serve as the applied context for the mathematics, followed by a “mathematical challenge,” a more abstract problem that derives from the applied context. To be concrete, we take as an example the distribution of finch species on the islands of the Galapagos shown in Figure 1 (data from [16], which does not give the original source).

\(^1\)Diamond did not actually publish the matrix. Instead, he published equivalent information as a set of lists.
Measures of competition.

(1) Checkerboards. Jared Diamond defined a checkerboard as a pair of species that never occur together on an island. For data in the form of a ‘species × island’ co-occurrence matrix, a checkerboard is just a pair of rows with dot product zero. In Figure 1, species J and K form a checkerboard. Evidently, a large number of checkerboards reflects a high degree of interspecific competition. For the Galapagos finches, there are ten checkerboards, out of \( \binom{13}{2} = 78 \) possible pairs.

Checkerboards are all-or-nothing: if two species almost always avoid each other, but do occur together on even one island, they don’t qualify as a checkerboard. Counting checkerboard units is more flexible.

(2) Checkerboard units. A species pair and an island pair constitute a checkerboard unit if each species appears on exactly one of the two islands and each island is home to exactly one of the two species. In other words, the \( 2 \times 2 \) submatrix determined by the two rows and two columns has one of two forms:

\[
\begin{pmatrix}
1 & 0 \\
0 & 1
\end{pmatrix}, \quad \begin{pmatrix}
0 & 1 \\
1 & 0
\end{pmatrix}.
\]

(1.1)

In Figure 1, species J and K, and islands 4 and 14 form a checkerboard unit. The total number of checkerboard units (CUs) provides a second measure of the degree of competition, with larger values taken to indicate higher levels of competition. For the finches, there are 333 CUs in all, out of a total of \( \binom{13}{2} \binom{17}{2} = 10608 \) submatrices of size \( 2 \times 2 \).

(3) Species combinations. An alternate approach is to count species combinations. At most, if no species combinations are repeated, there can be a different combination on each island. For the finches, four species combinations occur twice (on island pairs 3/4, 9/10, 12/13, and 16/17) and so there are thirteen different combinations, with seventeen as the theoretical maximum. Competition tends to restrict the number of combinations.

The ecologists’ question begins by defining measures of competition among species distributed over several ecosystems.

**Figure 1.** The co-occurrence matrix of the finches on the Galapagos islands.

![Co-occurrence Matrix](image)
Are these values large or small? Although theoretical upper bounds come from the dimensions of the matrix alone, other features such as the density of 1s, or the values of the row and column totals, can restrict the values of the various measures. Checkerboards, for example, tend to be much more frequent in matrices with few 1s. Ecologists are not unanimous, but there appears to be a working consensus that in trying to decide whether ten is a large number of checkerboards (or 333 is a large number of CUs, or thirteen is a small number of species combinations) the question should be asked relative to the observed row and column totals, that is, taking those totals as given. Row sums, it is argued, tell us more about the dispersal ability of individual species than about how the species interact. Similarly, column sums tell us how hospitable the individual islands are, not about how species may compete for resources on those islands. In statistical terms, the row and column sums are ancillary to the main question, and should be taken as fixed at their observed values. The features of interest derive from the pattern of 0s and 1s given those values.

The ecologists’ question. Given a co-occurrence matrix and the observed value of some measure of competition, what is the probability that a random matrix with the same row and column totals as the original will show a level of competition at least as high as the original? For example, how likely is it that a random $13 \times 17$ matrix of 0s and 1s, with the same row and column sums as in Figure 1, will have at least ten checkerboards? (Here “random” means that all matrices are equally likely.)

In principle, the ecologists’ question has an easy answer: list all the matrices with the right marginal totals, weighting them equally, and find the proportion that have ten or more checkerboards. In practice, the “easy” answer isn’t available, because the number of matrices is too large to list, even in these days of fast computers. For the finches, for example, there are more than $6.71 \times 10^{16}$ matrices with the same marginal totals [11, p. 92]. Even with too many matrices to list, it would still be possible to answer the ecologists’ question if one could generate random matrices uniformly, that is, in a way that assigns equal probabilities to all the matrices, because one could rely on the law of large numbers and estimate the probability of interest as

$$\hat{p} = \frac{\# \text{ random matrices with requisite level of competition}}{\text{total # random matrices generated}}. \quad (1.2)$$

This approach, via simulation, leads to a new question:

The mathematicians’ challenge. Let $\mathcal{A}(r, c)$ be the set of all binary matrices with fixed vector $r$ of row sums and fixed vector $c$ of column sums. How can we generate random elements of $\mathcal{A}(r, c)$ in a way that gives equal probabilities to all matrices?

Ecologists themselves developed two ad hoc approaches to generating random cooccurrence matrices. “Fill algorithms” [16] start with an empty matrix and place 1s sequentially, backing up as necessary, until all rows and columns sum to the required totals. “Swap algorithms” begin with the observed matrix and make a large number of random interchanges, swapping 0s and 1s in a way that leaves marginal totals unchanged.

Both approaches, as originally formulated, give biased results (see [6] and [18]) because they fail to generate matrices with uniform (equal) probabilities. However, each
approach can be understood and analyzed as an instance of a more general simulation method. During the past year a team of researchers at Stanford and Harvard has published papers that develop and analyze an unbiased variant of the fill algorithm as an instance of sequential importance sampling (see [11, p. 92] and references cited in that paper). It is the purpose of this paper to show how the swap algorithm can be regarded as an instance of Markov chain Monte Carlo [5]. Understood within that framework, the algorithm lends itself to systematic analysis and several improvements.

For the particular purpose of solving the ecologists’ problem, the approach based on importance sampling turns out to be much more efficient. However, the swap algorithm operating on (0, 1)-matrices offers a concrete way to introduce the ideas of Markov chain Monte Carlo, and it turns out that there is nice interplay between convergence properties of the algorithm and groups of symmetries operating on the co-occurrence matrices.

In what follows, section 2 shows how swapping \(2 \times 2\) submatrices determines a random walk on a graph whose vertices are the matrices of \(A(r, c)\). Section 3 shows how the standard theory of discrete Markov chains lets us see that the \(2 \times 2\) swaps do not give the equal probabilities we seek. Section 4 explains the Metropolis-Hastings algorithm and applies it to \(2 \times 2\) swaps in a way that does give equal probabilities to all matrices. Section 5 explains how some elementary work with groups, combined with results about “lumping” Markov chains, gives a faster way to generate the matrices by accelerating convergence to a uniform stationary distribution. Section 6 extends swapping to submatrices larger than \(2 \times 2\). Finally, section 7 wraps up with a comparison of methods using some simple examples. Most of the exposition is based on results that are well known. Propositions 4 and 5, and the two theorems, are original.

2. WALKING AT RANDOM USING \(2 \times 2\) SWAPS. Our goal, then, is to generate random, equiprobable matrices with fixed row and column sums. One common method used by ecologists relies on \(2 \times 2\) swaps. Two distinct binary matrices are called swappable if they have the same row and column totals. The only swappable \(2 \times 2\) matrices are the checkerboard units defined in (1.1). In any (0,1)-matrix, if a swappable \(2 \times 2\) submatrix is replaced by its counterpart, the row and column totals of the matrix are preserved. Moreover, any two co-occurrence matrices with the same row and column totals are connected by some series of \(2 \times 2\) swaps (see [2] and [15]). Based on these observations, ecologists use the following two-step algorithm to randomly generate co-occurrence matrices starting from an initial matrix:

**Algorithm 22** \((2 \times 2\) swap)\n
1. **Random:** Choose two rows uniformly at random without replacement; choose two columns uniformly at random without replacement.

2. **Swap:** Is the resulting \(2 \times 2\) submatrix swappable?
   a. No: Return to step 1. (Do not count this as a swap.)
   b. Yes: Make the swap.

Iterating steps 1 and 2 a large number of times produces a random matrix, although it turns out that not all matrices are equally likely.

**Example 1.** Consider a miniature ecosystem with only three islands, only two species, and exactly one species living on each island. This simple ecosystem, together with all three possible co-occurrence matrices \(a, b,\) and \(c\) satisfying these constraints of marginal totals, is shown in Figure 2. Starting from matrix \(a\), if we pick columns 2...
and 3 and perform the $2 \times 2$ swap, we obtain matrix $b$. If we pick columns 1 and 3 and perform the swap, we obtain matrix $c$, and so forth.

<table>
<thead>
<tr>
<th>Islands</th>
<th>Species</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>total</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td>3</td>
</tr>
</tbody>
</table>

$$a = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad b = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}, \quad c = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

**Figure 2.** An ecosystem with row sums $r = (2, 1)$ and column sums $c = (1, 1, 1)$.

Let $\mathcal{A}(r, c)$ be the set of all co-occurrence matrices with the prescribed row sums $r$ and column sums $c$. Then the set $\mathcal{A}(r, c)$ and the possible $2 \times 2$ swaps form a graph. The vertex set of the graph is represented by the matrices of $\mathcal{A}(r, c)$ and the edge set by the $2 \times 2$ swaps. Figure 3 displays the graph and the corresponding adjacency matrix $A = \{a_{ij}\}$ of Example 1. (By definition, $a_{ij} = 1$ if and only if there is an edge from $i$ to $j$.)

Moreover, performing random $2 \times 2$ swaps on $\mathcal{A}(r, c)$, as in ALGORITHM 22, is equivalent to performing a random walk $\{X_t\}$ on the graph. Here, $X_t$ denotes the state of the walk at time $t = 0, 1, \ldots$. Given $X_t$, the state of $X_{t+1}$ is chosen uniformly at random from the adjacent neighbors of $X_t$. Because the transition probability from a state at time $t$ to a state at time $t + 1$ only depends on $X_t$, the random walk can also be seen as a Markov chain. The transition matrix of the chain is obtained by normalizing each row of the adjacency matrix so that rows add to 1. The transition matrix of the random walk on the graph in Figure 3 is shown in equation (2.1):

$$P = \begin{pmatrix} 0 & 1/2 & 1/2 \\ 1/2 & 0 & 1/2 \\ 1/2 & 1/2 & 0 \end{pmatrix}.$$  \hspace{1cm} (2.1)

The simplicity of this first example can be deceptive. Generally, the number of co-occurrence matrices of moderate size with the same marginal totals is very large. The sheer size of the state space of the corresponding Markov chain poses a variety of computational challenges. In what comes next, we will rely on the following example to illustrate some of these challenges and some strategies for dealing with them.

April 2003] MARKOV CHAIN MONTE CARLO APPLIED TO COMMUNITY ECOLOGY 269
Example 2. Consider all possible matrices of $\mathcal{A}(r, c)$ with row sums $r = (3, 2, 1)$ and column sums $c = (2, 2, 1, 1)$. There are eight such matrices, which are shown in Figure 4 along with the graph and transition matrix determined by the $2 \times 2$ swaps.

$$
\begin{align*}
  a &= \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} &
  b &= \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} &
  c &= \begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix} &
  d &= \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix}
\end{align*}
$$

$$
\begin{align*}
  e &= \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} &
  f &= \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} &
  g &= \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix} &
  h &= \begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}
\end{align*}
$$

Figure 4.

Ecologists currently use the $2 \times 2$ swap algorithm to generate random matrices that are then used to test hypotheses about interspecific competition [6]. This use raises two questions: First, does their method generate matrices that are equally likely? Second, how many swaps are needed to generate a matrix that can reasonably be regarded as random?

3. LIMITING DISTRIBUTIONS AND THE METROPOLIS ALGORITHM.

An answer to the first question—Are the matrices equally likely?—comes from a standard result for Markov chains, applied to the case of nearest neighbor random walks on graphs. Let $p^{(t)}$ be the row vector whose $i$th element is the probability that the walk is in state $i$ after $t$ steps, i.e., $p_i^{(t)} = \Pr[X_t = i]$. Thus $p^{(t)}$ gives the probability distribution over the states of the chain at time $t$. Under mild regularity conditions (see, for example, [10] or [14]), $p^{(t)}$ converges to a unique limiting distribution $\pi$, regardless of the starting distribution $p^{(0)}$. For the ecologists’ $2 \times 2$ swap walk, $p^{(t)}$ gives the probabilities for the various co-occurrence matrices, and the requirement that all matrices be equally likely means that $p^{(t)}$ must converge to a uniform distribution as the limiting distribution. Unfortunately, that might not happen.

We rely here on the Chapman-Kolmogorov equation $p^{(t+1)} = p^{(t)}P$. In words, moving forward one step in time corresponds to multiplication by the transition matrix $P$. An immediate consequence of this relation is that we can find the limiting distribution $\pi$ as the solution to a linear system: if $\pi$ is the limiting distribution, then moving ahead one step must leave $\pi$ unchanged, and so $\pi P = \pi$. (Any vector $x$ that solves $xP = x$ is called a stationary vector for $P$.)
It follows almost immediately that, for a random walk on a graph, the limiting probabilities for the vertices are proportional to their degrees. To see this let $d_i$ be the degree of vertex $i$, and let $a_{ij} = 1$ if $\{i, j\}$ is an edge, 0 otherwise. Then $p_{ij} = a_{ij}/d_i$ and $\sum_i a_{ij} = d_j$. Now set $\pi_i = d_i/\sum d_i$. It is straightforward to check that

$$\sum_i \pi_i p_{ij} = \sum_i a_{ij} / \sum d_i = d_j / \sum d_i = \pi_j,$$

and thus $\pi P = \pi$.

Intuitively, the result makes sense. A vertex with many edges and thus many neighbors will be visited more often than a vertex with few neighbors, and in the limit the visiting rate of a vertex is proportional to the number of neighbors it has.

For Example 1, symmetry leads one to expect limiting probabilities to be equal, and it is easy to check that $\pi = (1/3, 1/3, 1/3)$ solves $\pi P = \pi$. Equivalently, all three vertices have degree 2, and so they are visited equally often by the random walk.

For Example 2, on the other hand, the inner vertices $a, b, c$, and $d$ have degree 5, whereas the outer vertices $e, f, g$, and $h$ have degree 4. The limiting distribution $\pi$ is $(5, 5, 5, 5, 4, 4, 4, 4)/36$.

As the example illustrates, if random matrices are generated using the $2 \times 2$ swap algorithm, the result will be biased in favor of matrices with higher degrees. Because each swappable $2 \times 2$ matrix is a checkerboard unit, the degree of a vertex equals the number of checkerboard units in the corresponding matrix. The bottom line: the method of swaps overrepresents matrices that correspond to higher levels of competition. In other words, the bias makes it somewhat harder to declare an observed level of competition to be statistically significant.

One way to remove the bias is to add self-loops to the graph walks by “holding in place” whenever a randomly chosen submatrix is unswappable:

**Algorithm 22H**

1. **Random.** Choose two rows uniformly at random without replacement; choose two columns uniformly at random without replacement.

2. **Swap or hold.** Is the resulting submatrix swappable?
   a. No. Stay in place, but count this as a step of the walk.
   b. Yes. Make the swap.

Counting each self-loop as an edge gives all vertices the same degree, and it follows that Algorithm 22H gives a uniform limiting distribution, as required.

Unfortunately, holding in place is inefficient. Take the finch data as an example. There are $\binom{13}{2} \binom{17}{2} = 10608$ $2 \times 2$ submatrices, but only about 300, or 3%, are swappable. One would expect, then, to hold in place for thirty-odd steps before each move to a new vertex. (Although this sluggishness in the algorithm is undesirable, in practice the inefficiency is not as extreme as it may seem, because holding in place doesn’t use much computing time.)

The Metropolis algorithm (see [8], [9], and [12]) provides a much faster way to eliminate the bias in the swap walk. This algorithm can be used to alter the limiting distribution of any in a large class of Markov chains. Consider first the simple example of a random walk on a 3-point linear graph (Figure 5). Suppose we want to alter the walk to make the limiting distribution uniform. This requires reducing the frequency of visits to state 2, while increasing the frequency of visits to states 1 and 3. The Metropolis algorithm replaces each transition $i \rightarrow j$ with a two-stage transition: first
a proposal stage determined by $P$, then a Metropolis coin flip that tells whether to accept the proposal.

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

Figure 5.

To see how this works, suppose first that the chain is in state 2. The original transition probabilities govern the proposal stage, so states 1 and 3 are proposed, each with probability 1/2 ($p_{21} = p_{23} = 1/2$). Because both these states are underrepresented in the limiting distribution for $P$, the Metropolis algorithm accepts either proposal with probability 1.

Now suppose the chain is in state 1. Since $p_{12} = 1$, the proposed state is 2. Because that state is overrepresented, we reduce the chance of going there by inserting a coin flip: if the coin lands heads ($p = 1/2$), we accept the proposal and move to 2, but if it lands tails, we stay in 1 and count that as a “move.” Transitions from 3 are similar, and the resulting “Metropolized” transition matrix is given in Figure 6. In effect, the Metropolis algorithm has added self-loops to the graph, and the new stationary distribution is uniform.

![Figure 6](image)

The Metropolis algorithm, in its general form, applies to any Markov chain that satisfies a symmetry condition called reversibility: if $\pi$ is the stationary vector, the chain is reversible if and only if

$$\pi_i p_{ij} = \pi_j p_{ji}$$

for all $i$ and $j$. All graph walks are reversible, because $\pi_i p_{ij} = \pi_j p_{ji}$ precisely when $\pi_i/\pi_j = p_{ji}/p_{ij}$; the left-hand ratio equals $d_i/d_j$, and the right-hand ratio equals $(1/d_j)/(1/d_i)$. In particular, swap walks on co-occurrence matrices are reversible.

Given a reversible chain $P$, its stationary vector $\pi$, and a desired (target) distribution $\tilde{\pi}$, the Metropolis algorithm defines a new chain $\tilde{P}$ by

$$\tilde{p}_{ij} = \begin{cases} 
1 - \alpha_{ij} & \text{if } i = j, \\
\frac{p_{ij} \alpha_{ij}}{\sum_{j \neq i} \tilde{p}_{ij}} & \text{if } i \neq j,
\end{cases}$$

where $\alpha_{ij} = \min\{1, \pi_i \tilde{\pi}_j / \pi_i \pi_j\}$. Here $p_{ij}$ corresponds to proposing a move from $i$ to $j$, and $\alpha_{ij}$ is the “acceptance probability” for the Metropolis coin flip.

A straightforward matrix multiplication confirms that $\pi \tilde{P} = \tilde{\pi}$. To solve the ecologists’ problem, the limiting distribution $\tilde{\pi}$ needs to be uniform. For uniform $\tilde{\pi}$, the
acceptance probabilities simplify, since $\tilde{\pi}_i = \tilde{\pi}_j$; for random walks on graphs, these simplified acceptance probabilities become

$$\alpha_{ij} = \min\{1, d_i/d_j\}, \ j \neq i.$$  

For the $2 \times 2$ swap walk, this gives the following three-part Metropolized swap step.

**Algorithm 22M** (Metropolized $2 \times 2$ swap)

1. **Random:** Choose two rows uniformly at random without replacement; choose two columns uniformly at random without replacement.

2. **Proposed:** Is the resulting $2 \times 2$ submatrix swappable?
   a. No: Return to step 1. (Do not count this as a step.)
   b. Yes: Take the 1s complement as a proposed swap.

3. **Metropolis:** Let $a_j$ be the matrix obtained from $a_i$ by making the swap in 2b. Let $d_j$ and $d_i$ be their respective vertex degrees, and let $\alpha = \min\{1, d_i/d_j\}$. Generate a uniform random number $u$ in $[0, 1]$.
   a. If $u \leq \alpha$, accept: move from $a_i$ to $a_j$.
   b. If $u > \alpha$, reject: remain at $a_i$ (self-loop).

**Proposition 1.** Algorithm 22M determines the edges of a connected graph (with self-loops) whose vertex set is $A(r, c)$. The corresponding Markov chain has a symmetric transition matrix, and thus a uniform limiting distribution.

**Proof.** Connectivity is a corollary of Ryser’s result [15] that $2 \times 2$ swaps connect $A(r, c)$. Symmetry of $\tilde{P}$ follows from $p_{ij} = a_{ij}/d_i$ and the definitions of $\tilde{p}_{ij}$ and $\alpha_{ij} = \min\{1, d_i/d_j\}$:

$$\tilde{p}_{ij} = p_{ij} \alpha_{ij} = (a_{ij}/d_i)(\min\{1, d_i/d_j\}) = a_{ij}/\max\{d_i, d_j\} = \tilde{p}_{ji},$$

since $a_{ij} = a_{ji}$.

**Example 3.** In Figure 4, if you are allowed to see the whole graph at once, there is a simple way to get a uniform limiting distribution: just add a self-loop at each of the four outer vertices. Then each of the eight vertices has degree 5, and it is easy to check that the transition matrix is symmetric.

Adding self-loops at vertices $e$ through $h$ reduces all transition probabilities that originate at outer vertices from 1/4 to 1/5. The Metropolis algorithm achieves the same uniform limiting distribution, but its action is more selective, because it is based on comparing incoming and outgoing vertex degrees. Transitions between two outer vertices have the same probabilities in both directions, and are left unchanged, as are transitions between inner vertices. However, transitions like $a$ to $e$, from outer to inner, are less likely in the original chain than are transitions from inner to outer. The Metropolis algorithm reduces the chance of transitions from inner to outer to match the chance for the opposite direction.

Thus the transition matrix for the Metropolized walk has the same first four rows as in Figure 4. In the lower left $4 \times 4$ submatrix, for transitions from outer to inner vertices, each 1/4 becomes 1/5. In the lower right $4 \times 4$, off-diagonal elements are unchanged; diagonal elements go from 0 to 1/10. The resulting matrix is again symmetric, and so the limiting distribution is uniform.
As in Example 3, the Metropolis algorithm operates “locally” in the sense that transition probabilities $\tilde{p}_{ij}$ can be computed without knowing the entire matrix $P$. Given current state $i$, the transition probabilities $p_{ij}$ are enough to determine the proposed new state $j$. Given $j$, the acceptance probability depends only on the limiting probabilities $\pi_i$, $\pi_j$, $\tilde{\pi}_i$, and $\tilde{\pi}_j$.

These features can be crucial in practice. For swap walks on co-occurrence matrices, the state space is much too large to list, but because the Metropolis algorithm uses only local information, all that we need at any given step is a pair of co-occurrence matrices—the current one, and the proposed destination.

Here is how one uses Algorithm 22M to solve the ecologists’ problem. First, run steps 1–3 a large enough number of times to ensure that the chain has reached its equilibrium distribution. Methods for determining the required number of steps are known (see, for instance, [13]). After these preliminary iterations (called the “burn-in” period in the MCMC literature), use the next several thousand iterations to estimate the probability of interest: each iteration gives a random co-occurrence matrix; for each, determine whether the measure of competition is at least as extreme as the actual observed values; finally, compute the proportion of matrices for which such extreme values occur, as in (1.2).

Of course, in (1.2) successive random matrices are independent, like coin tosses. Here, successive matrices generated by Algorithm 22H are highly dependent. Nevertheless, the ratio in (1.2) converges to the true probability. (This result is the Ergodic Theorem; see [14].)

4. HOW QUICKLY DO THE MATRICES BECOME UNIFORM? When ecologists first used randomly generated matrices to attack Jared Diamond’s community assembly rules, desktop computers barely existed. It is a mark of how primitive things were at the time that the first implementations of the swap algorithm used only “several” swaps [3]. In retrospect, this is a major source of bias, because a handful of swaps can’t take you very far from the original matrix. Taking so few steps makes it all but certain that the “random” matrices will look very much like the original. Reversing perspective, the original will look very much like the matrices that are regarded as random. This is precisely the conclusion that Connor and Simberloff presented in their 1979 paper [3]. The same ideas are easier to see by thinking about shuffling cards. In effect, Connor and Simberloff took an ordered deck of cards, shuffled only once or twice, called the result “random,” and observed that the original ordered deck was itself not very different from “random.”

Although in hindsight it is obvious that, for data like the finch matrix, a few steps cannot take you far from your starting point, it is not so obvious how many steps are in fact needed: 100? 1,000? 10,000? The number of matrices with the same row and column sums as for the finches is astronomically large, in the vicinity of $10^{17}$. Does that mean we need to take comparably many steps to get a random matrix?

The size of the vertex set for the finch data is truly daunting, but it nevertheless turns out that no two vertices are very far apart. Brualdi [2, p. 172] shows that the largest number of $2 \times 2$ swaps required to connect two $(0, 1)$-matrices $b$ and $c$ is at most $d(b, c)/2 - 1$, where $d(b, c)$ is the number of positions $(i, j)$ for which $b_{ij} \neq c_{ij}$. For the finch data, the row-sum constraints impose an upper bound of 53 on $d(b, c)$, so it takes at most twenty-five swaps to get from one matrix to another.

Although the comparatively low upper bound on the distance between matrices offers reassurance that the swap-walk algorithm can indeed generate random matrices in a workable number of steps, there remains the question of how quickly $P^{(n)}$ converges to $\pi$. The standard answer (for all reversible Markov chains, and thus for all
The function $p(t)$ is given by the spectral gap of the transition matrix $P$. The matrix has 1 as an eigenvalue, and all other eigenvalues have modulus at most 1. Let $\theta_*$ be the second largest eigenvalue modulus (SLEM), possibly equal to 1. The spectral gap is $1 - \theta_*$. The next few paragraphs set out the logic that relates the spectral gap to the convergence behavior of $p(t)$. (Both “modulus” and “SLEM” are standard terminology, because in general eigenvalues may be complex. For reversible matrices, however, and thus for all graph walks, all eigenvalues are real, and “modulus” reduces to absolute value.)

We first define the total variation distance between two probability vectors of $k$ entries:

$$\|p^{(t)} - \pi\| = \frac{1}{2} \sum_{i=1}^{k} |p_i^{(t)} - \pi_i|.$$ 

The factor 1/2 ensures that the distance is never greater than 1. In practice, we want $\|p^{(t)} - \pi\|$ to be small, because when $\pi$ is uniform and $p^{(t)}$ is close to $\pi$, we know that all matrices have (nearly) equal probabilities. In typical applications, a chain is run for a “burn-in” period of $T$ steps, and these initial results are discarded. Values for $t$ greater than $T$ are considered to be random, and are used.

The result we seek states (see [14]) that, for a reversible chain, there is a constant $M$ such that

$$\|p^{(t)} - \pi\| \leq M\theta_*^t,$$

where $\theta_*$ is the second largest modulus of the eigenvalues of $P$. This result follows from the spectral decomposition of $P$ of size $k \times k$. For the special case when $P$ is symmetric, its eigenvalues are real, and there exists an orthonormal basis consisting of left eigenvectors $x_i$ with $x_iP = \lambda_i x_i$ and

$$P = \sum_{i=1}^{k} \lambda_i E_i,$$

where $E_i = x_i^T x_i$. It follows from the orthonormality that $E_i^2 = E_i$ and $E_iE_j = 0$ for $i \neq j$, whence

$$P' = \sum_{i=1}^{k} \lambda_i^t E_i.$$

What if $P$ is not symmetric? If $P$ is reversible, then $P$ is similar to a symmetric matrix, the eigenvalues are real, and there is a parallel spectral decomposition in which matrices $DE_iD^{-1}$ replace the $E_i$. (To check this, let $D$ be the diagonal matrix formed from the square roots of the $\pi_i$. Reversibility of $P$ is then equivalent to symmetry of $DPD^{-1}$, and the decomposition follows.)

The fact that $P$ is stochastic ensures that the largest eigenvalue $\lambda_1 = 1$, and $|\lambda_i| \leq 1$ for $i \neq 1$. Weak additional conditions (e.g., that $P$ is irreducible and aperiodic) give $|\lambda_i| < 1$ for $i \neq 1$. It then follows from the spectral decomposition that $P' \rightarrow E_1$. Moreover, since $p^{(0)}P' \rightarrow \pi$ for all $p^{(0)}$, we must have $p^{(0)}E_1 = \pi$ for all $p^{(0)}$, which can only be true if each row of $E_1$ coincides with $\pi$. Thus

$$\|p^{(t)} - \pi\| = \|p^{(0)}P' - p^{(0)}E_1\| = \left\| \sum_{i=2}^{k} \lambda_i^t p^{(0)}E_i \right\|.$$
The last quantity can be bounded by $M\theta^t$ for a suitable choice of $M$. A chain with this property is called geometrically ergodic, and $\theta_*$ is called its rate of convergence. A good MCMC algorithm is one for which $\theta_*$ is small, or, equivalently, the spectral gap $1 - \theta_*$ is large.

From theory, then, the smaller the value of $\theta_*$, the fewer steps are needed to get $\|p^{(t)} - \pi\| < \epsilon$ for any prescribed value of $\epsilon$. In practice, it can be hard to make direct use of the theory. For one thing, some kinds of steps take longer to execute than others, so fewer steps are not automatically better. For example, the holding step 2a of ALGORITHM 22H requires no additional computing, whereas Metropolizing a chain makes each step slower than its unmetropolized version.

Another difficulty comes from the size of the state space. For large spaces, computing the transition matrix is not practical, in which event implementation depends on finding some alternative to working directly with the matrix $P$; for example, via $2 \times 2$ swaps. Without knowing $P$ explicitly, it is ordinarily impossible to find its eigenvalues, meaning that the bound $M\theta^t$ can’t be used to find the number of steps needed to bring $p^{(t)}$ within a distance $\epsilon$ of $\pi$. Here, too, finding $T$ depends on some alternative to working directly with $P$. (See, for example, [13].)

In practice, then, one doesn’t ordinarily compute eigenvalues of $P$. Nevertheless, the bound $M\theta^t$ is useful because it suggests that we can accelerate convergence by employing strategies that reduce the value of $\theta_*$. The next two sections present two such strategies for swap walks on $A(r, c)$.

5. ACCELERATING CONVERGENCE BY LUMPING AND HOMOGENIZING.

**Lumping a Markov chain.** A shortcoming of the swap algorithm, and a potential shortcoming of any Markov chain Monte Carlo algorithm, is that convergence may be slow. For large co-occurrence matrices, it may take a very large number of swaps to produce a random matrix. This section uses the strategy of lumping a Markov chain to reduce the size of the state space, increase the spectral gap, and accelerate convergence. We first present the ideas informally by example.

**Example 4.** Refer to the graph in Figure 4, and consider simpler graph walks obtained by “folding” along lines of symmetry, as in Figure 7. Folding the graph corresponds to partitioning the states, and raises two questions. First recall that, in these graphs, vertices correspond to binary matrices. What is it about the matrices that gives rise to the symmetries? Second, of all the possible ways to partition the states, which ones are in some sense compatible with the original set of transition probabilities?

![Figure 7. Folding a graph along lines of symmetry.](image)
The answer to the first question is straightforward: the symmetries in the graph arise from repetitions in the row and column totals. Thus, for example, in the matrices \(a\) through \(h\) of Figure 4, the first two column totals are both equal (to 2); folding across the horizontal line of symmetry lumps together matrices obtained from one another by interchanging the first two columns (\(c, d; e, g; f, h\)). Similarly, the vertical line of symmetry corresponds to permuting the last two columns, whose totals are both equal to 1.

The answer to the second question is well known (see [10]):

**Definition 1.** Let \(\mathcal{L} = \{L_1, L_2, \ldots, L_J\}\) be a partition of the set of states of a Markov chain \(\mathbf{P}\). Then \(\mathbf{P}\) is lumpable with respect to \(\mathcal{L}\) if and only if for any pair of sets \(L\) and \(L'\) in \(\mathcal{L}\) and any state \(i\) in \(L\), the probability of going from \(i\) to \(L'\) doesn’t depend on \(i\):

\[
\Pr\{X_{t+1} \in L' \mid X_t = i\} = \sum_{j \in L'} p_{ij}
\]

is the same for all \(i\) in \(L\). The common probabilities define a new Markov chain, the lumped chain \(\mathbf{P}_L\), with state space \(\mathcal{L}\) and transition probabilities \(p_{LL'} = \Pr\{X_{t+1} \in L' \mid X_t = L\}\).

**Example 5.** Consider the partition \(L_1 = \{a, b\}, L_2 = \{c, d\}\), and \(L_3 = \{e, f, g, h\}\). Figure 8 shows two sets of transition probabilities. The left-hand matrix gives probabilities from individual states \(a, b, \ldots\) to the “lumps” \(L_1, L_2, \) and \(L_3\), showing that the partition meets the requirement of the definition. The right-hand matrix gives transition probabilities for the “lumped chain.” Comparing eigenvalues for the original and lumped chains shows why lumping is a useful strategy for speeding convergence. The second largest eigenvalue modulus for the original chain is 0.5; the corresponding value for the lumped chain is 0.2.

<table>
<thead>
<tr>
<th></th>
<th>(L_1)</th>
<th>(L_2)</th>
<th>(L_3)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(a)</td>
<td>(b)</td>
<td>(c)</td>
</tr>
<tr>
<td>(a)</td>
<td>0</td>
<td>1/5</td>
<td>1/5</td>
</tr>
<tr>
<td>(b)</td>
<td>1/5</td>
<td>0</td>
<td>1/5</td>
</tr>
<tr>
<td>(c)</td>
<td>1/5</td>
<td>1/5</td>
<td>0</td>
</tr>
<tr>
<td>(d)</td>
<td>1/5</td>
<td>1/5</td>
<td>1/5</td>
</tr>
<tr>
<td>(e)</td>
<td>1/4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(f)</td>
<td>0</td>
<td>1/4</td>
<td>0</td>
</tr>
<tr>
<td>(g)</td>
<td>1/4</td>
<td>0</td>
<td>1/4</td>
</tr>
<tr>
<td>(h)</td>
<td>0</td>
<td>1/4</td>
<td>0</td>
</tr>
</tbody>
</table>

**Figure 8.** Lumping a Markov chain and the lumped chain.

Lumping a Markov chain corresponds to a kind of projection, one that reduces dimensionality from the number of states to the number of lumps. To show this, we introduce a standard reformulation of the lumpability criterion [10, pp. 123–132].

**Proposition 2.** Given a partition \(\{L_j\}\) of the state space of a Markov chain with transition matrix \(\mathbf{P}\), form the matrix \(\mathbf{V}\) whose columns are indicator variables for...
the elements of the partition: \( v_{ij} = 1 \) if state \( i \) belongs to set \( L_j \), 0 otherwise. Let \( U = (V^T V)^{-1} V^T \). Then \( P \) is lumpable if and only if \( VU P V = PV \), in which case the transition matrix of the lumped chain is \( P_L = UP \).

**Example 6.** If we take the lumping in Figure 8 as our example, then \( V \), \( U \), and \( VU \) are as follows:

\[
V = \begin{pmatrix}
L_1 & L_2 & L_3 \\
1 & 0 & 0 \\
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1 \\
0 & 0 & 1 \\
0 & 0 & 1
\end{pmatrix}, \quad U = \begin{pmatrix}
1/2 & 1/2 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1/2 & 1/2 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1/4 & 1/4 & 1/4 & 1/4 \\
0 & 0 & 0 & 0 & 1/4 & 1/4 & 1/4 & 1/4
\end{pmatrix},
\]

\[
VU = \begin{pmatrix}
1/2 & 1/2 & 0 & 0 & 0 & 0 & 0 & 0 \\
1/2 & 1/2 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1/2 & 1/2 & 0 & 0 & 0 & 0 \\
0 & 0 & 1/2 & 1/2 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1/4 & 1/4 & 1/4 & 1/4 \\
0 & 0 & 0 & 0 & 1/4 & 1/4 & 1/4 & 1/4 \\
0 & 0 & 0 & 0 & 1/4 & 1/4 & 1/4 & 1/4 \\
0 & 0 & 0 & 0 & 1/4 & 1/4 & 1/4 & 1/4
\end{pmatrix}.
\]

The matrices \( V \) and \( VU \) have several features worth noting:

1. The columns of \( V \) are necessarily orthogonal. Because they are indicators for the “lumps,” we refer to their span \( \langle V \rangle \), the column space of \( V \), as “lump space.” This is the geometric way of saying that a state belongs to one and only one lump: the indicator variables for a partition are always orthogonal.
2. \( VU = V(V^T V)^{-1} V^T \) is the (symmetric) matrix of the projection onto lump space. Thus the lumpability criterion can be stated in words: \( PV \) equals its projection onto lump space, i.e., \( PV = P_L PV \), where \( P_L = VU \).
3. \( P_L = VU \) is a transition matrix in its own right. The corresponding graph has one component for each lump; each component is a complete subgraph.
4. It follows from the lumpability criterion \( P_L PV = PV \) that \( P_L PP_L = PP_L \) and \( (PP_L)^t = P^t P_L \) for \( t \geq 1 \). In words, \( t \) steps of the lumped chain are equivalent to \( t \) steps of the original, followed by a projection onto lump space.

**Example 7.** Exploiting symmetries due to repeated marginal totals is by no means the only way to lump a Markov chain. For our example, there are many valid lumpings that do not correspond to those symmetries. It is straightforward, though tedious, to verify that all of the following are lumpable partitions, and that there are no others. Note that we have shown Partition 3 in Figure 8.

Partition 1. \( L_1 = \{a, b, c, d\}, L_2 = \{e, f, g, h\} \)
Partition 2. \( L_1 = \{a, b, c, d\}, L_2 = \{e, f\}, L_3 = \{g, h\} \)
Partition 3. \( L_1 = \{a, b\}, L_2 = \{c, d\}, L_3 = \{e, f, g, h\} \)
Partition 4. \( L_1 = \{a, b\}, L_2 = \{c, d\}, L_3 = \{e, f\}, L_4 = \{g, h\} \)
Proposition 3. The eigenvalues of $P_L$ are a subset of the eigenvalues of $P$.

Thus the spectral gap for the lumped chain is never smaller than that of the original chain. This fact is what makes lumping a possible strategy for accelerating convergence.

Using symmetry to lump a Markov chain. In general, the criterion for lumping a chain requires that we know the original transition matrix in order to find lumpable partitions. In practice, this requirement can make the search for a suitable partition unworkable for chains with very large state spaces. For the case of co-occurrence matrices, however, we can take advantage of symmetries based on repeated values in the row and column sums. This section shows how the symmetries determine a group of actions whose orbits give a lumpable partition of the state space.

Example 8. For the co-occurrence matrices in Example 2, the row sums $r = (3, 2, 1)$ contain no repetitions, but the column sums $c = (2, 2, 1, 1)$ contain two sets of repetitions. Define two actions on the set of co-occurrence matrices in Figure 4: $C_{12}$ interchanges columns 1 and 2; $C_{34}$ interchanges columns 3 and 4. Let $G$ be the group generated by $C_{12}$ and $C_{34}$. (In fact, $G$ is isomorphic to the Klein group $S_2 \times S_2$.) Note that the elements of $G$ preserve the edge structure of the graph. Consider, for example, matrices $a$ and $d$, which are connected via a $2 \times 2$ swap of the submatrix determined by rows 1 and 3, columns 1 and 4. The column permutation $C_{12}C_{34}$ maps $a$ to $b$ and $d$ to $c$; the images $b$ and $c$ are connected by a $2 \times 2$ swap operating on the submatrix determined by the same rows 1 and 3, but new columns 2 and 3 resulting from the column permutation.

For this example, regard two matrices as equivalent if you can get from one to the other by any of the column interchanges $C_{12}$, $C_{34}$, or $C_{12}C_{34}$. This equivalence partitions the matrices into three subsets, the orbits of $G$. They are $\{a, b\}$, $\{c, d\}$, and $\{e, f, g, h\}$. These are precisely the lumps in Figure 8.

More generally, given $A(r, c)$ define two rows to be equivalent if and only if they have the same row sums, and likewise for columns. Then permutations of rows and/or columns within equivalence classes constitute a group of actions that map $A(r, c)$ onto itself, with composition as the group operation. Two properties are immediate: first, by definition, the elements of $G$ preserve marginal sums; second, $G$ is isomorphic to a product of symmetric groups that correspond to the equivalence classes.

Moreover, $G$ preserves the edge structure of the graph $G$ whose vertices are the elements of $A(r, c)$ and whose edges are given by the $2 \times 2$ swaps.

Proposition 4. Let $G$ and $G$ be as just described. Then for any matrices $a$ and $a'$ in $A(r, c)$ and any member $g$ of $G$, $a$ and $a'$ are joined by an edge if and only if $ga$ and $ga'$ are joined by an edge.
**Proof.** Consider an arbitrary 2 \( \times \) 2 submatrix of \( a \), corresponding to rows \( i, j \) and columns \( k, l \). Denote this submatrix by \( a(i, j, k, l) \). Given \( g \) in \( G \), suppose that \( g \) maps row \( i \) into row \( gi \) (possibly equal to \( i \) or \( j \)) and maps row \( j \) into row \( gj \neq gi \)— and similarly for columns. By construction, \( ga(gi, gj, gk, gl) = a(i, j, k, l) \), with a parallel equality for \( a' \). It follows that for any pair of rows \( i, j \) and columns \( k, l \) there is a swap from \( a \) to \( a' \) if and only if there is a swap from \( ga \) to \( ga' \).

The concept of preserving edges extends easily to other graphs as well. Let \( G = (V, E) \) be a graph with vertex set \( V \) and edge set \( E \), and let \( G \) be a group of actions \( g : V \rightarrow V \). We call \( G \) edge-preserving if, for all vertex pairs \( \{x, y\} \) and members \( g \) of \( G \), \( \{x, y\} \) belongs to \( E \) precisely when \( \{gx, gy\} \) does. In our example, we used row and column interchanges to partition the co-occurrence matrices: Two matrices were in the same equivalence class if you could get from one to the other by permuting rows and/or permuting columns whose sums were equal. For an arbitrary graph \( G \) and group \( G \), define two vertices \( x \) and \( y \) to be equivalent if and only if there is a member \( g \) in \( G \) for which \( x = gy \). The resulting equivalence classes are the orbits of \( G \).

**Theorem 1 (Orbit lumping theorem).** Let \( G \) be a group of edge-preserving actions on a graph \( G = (V, E) \). Then the nearest neighbor random walk on \( G \) is lumpable with respect to the orbits of \( G \).

**Proof.** Let \( Gx \) and \( Gy \) be two arbitrary orbits of \( G \), and let \( x \) and \( x' \) be two arbitrary elements of \( Gx \). It suffices to show that the number of edges from \( x \) to \( Gy \) equals the number of edges from \( x' \) to \( Gy \). Let \( y_1, y_2, \ldots, y_n \) in \( Gy \) be the neighbors of \( x \) that belong to \( Gy \), and let \( s \) in \( G \) be an action for which \( x' = sx \). Then \( sy_1, sy_2, \ldots, sy_n \) belong to \( Gy \), and because \( G \) is edge-preserving, \( x' \) is connected to each of them. Moreover, the elements of \( G \) are bijections, so \( \{x, y_i\} \) is an edge if and only if \( \{x', sy_i\} \) is an edge.

**Homogenizing a lumpable Markov chain.** A quick look back may be helpful: Our goal is to generate random elements of \( A(r, c) \) via a sequence of random swaps. To accelerate convergence, and thus reduce the number of swaps required, we reduce the size of the state space, lumping the chain to a set of equivalence classes given by the orbits of a group. One obstacle remains before we can implement this strategy in practice; namely, for co-occurrence matrices it is not possible to run a Markov chain directly on the lumped chains, because the lumps themselves aren’t available to us. Our inability to list the states of the original chain also prevents us from working directly with equivalence classes of those states. Fortunately, we can run the equivalent of the lumped chain on the states of the original. We call this new chain the “homogenized chain.”

In practice, the idea is quite simple. To carry out one step of the homogenized chain, first take one step of the original chain, e.g., do a random swap. Then homogenize: randomly permute rows and columns within the subsets that have equal totals. For the co-occurrence matrices of Example 2, this means choosing one of the orders 1, 2 or 2, 1 for the first two columns, and one of the two orders 3, 4 or 4, 3 for the last two columns. The resulting transition probabilities are the same as those one would get by taking the transition probabilities for the lumped chain and dividing them equally among the states of the destination lump.

**Definition 2.** Let \( L \) and \( L' \) be states of a lumping \( L \) of a Markov chain, and let \( i \) in \( L \) and \( j \) in \( L' \) be states of the original chain. Then the homogenized chain \( P_H \) has
transition probabilities

\[ \tilde{p}_{ij} = p_{LL'}/|L'|. \]

In words, the probability of moving from \( i \) to \( j \) is the product of the probability of moving from \( L \) to \( L' \) in the lumped chain and the probability of choosing \( j \) uniformly at random from the states of \( L' \).

It follows that the transition matrix \( P_H = \{ \tilde{p}_{ij} \} \) for the homogenized chain is \( PVU = PP_\perp \).

**Example 9.** The transition matrix of the homogenized chain obtained from Example 2 is

\[
P_H = \begin{pmatrix}
a & b & c & d & e & f & g & h \\
1/10 & 1/10 & 1/5 & 1/5 & 1/10 & 1/10 & 1/10 & 1/10 \\
1/10 & 1/10 & 1/5 & 1/5 & 1/10 & 1/10 & 1/10 & 1/10 \\
1/5 & 1/5 & 1/10 & 1/10 & 1/10 & 1/10 & 1/10 & 1/10 \\
1/5 & 1/5 & 1/10 & 1/10 & 1/10 & 1/10 & 1/10 & 1/10 \\
1/8 & 1/8 & 1/8 & 1/8 & 1/8 & 1/8 & 1/8 & 1/8 \\
1/8 & 1/8 & 1/8 & 1/8 & 1/8 & 1/8 & 1/8 & 1/8 \\
1/8 & 1/8 & 1/8 & 1/8 & 1/8 & 1/8 & 1/8 & 1/8 \\
1/8 & 1/8 & 1/8 & 1/8 & 1/8 & 1/8 & 1/8 & 1/8 \\
\end{pmatrix}
\]

The characterization \( P_H = PVU \) leads directly to the eigenvalues and eigenvectors of the transition matrix \( P_H \) of the homogenized chain:

**Theorem 2.** Let \( P_H = PVU \) be a homogenized chain based on a lumping \( V \) of \( P \). Then the eigenvectors of \( P_H \) are of two kinds:

(a) eigenvectors \( x \) of \( P \) orthogonal to the lump space \( \langle V \rangle \), with 0 as the corresponding eigenvalue for \( P_H \); and

(b) vectors \( xP_\perp \), for which \( x \) is an eigenvector of \( P \) with eigenvalue \( \lambda \), \( xV \neq 0 \), and the corresponding eigenvalue of \( P_H \) is also \( \lambda \).

**Proof.** Let \( xP = \lambda x \).

(a) If \( x \perp \langle V \rangle \), then \( xP_H = xPP_\perp = \lambda xP_\perp = 0 \neq 0x \).

(b) If \( xV \neq 0 \), then \( (xP_\perp)P_H = x(P_\perp P_\perp) = xPP_\perp = (\lambda x)P_\perp = \lambda(xP_\perp) \).

It follows that the eigenvalues of \( P_H \) are the eigenvalues of \( P_L \), augmented by \( k \) zeros, where \( k = \text{dim} \langle V \rangle \perp \).

A second consequence relates the stationary distributions of \( P \) and \( P_H \). Let \( \pi \) be a stationary vector for \( P \), so that \( \pi P = \pi \). Then \( \pi V \neq 0 \) and \( P_H \) has stationary vector \( \pi P_\perp \). The two chains will have the same stationary vector precisely when \( \pi = \pi P_\perp \), i.e., when \( \pi \) lies in \( \langle V \rangle \), or, in words, when \( \pi \) is constant on lumps. Under what conditions will a lumping \( V \) of \( P \) yield \( \pi \in \langle V \rangle \)? A comprehensive answer would lead away from the main theme of this paper, but in the presence of symmetries, there is a simple answer: for lumps defined by the orbits of an edge-preserving group, all the states in a given orbit have the same vertex degree. Thus \( \pi \) is constant on orbits, and \( \pi P_\perp = \pi \).

Nominally, we can implement any homogenized chain by alternating two kinds of half-step moves. Because \( P_H = PP_\perp \), a half-step move of the first kind corresponds to
one step of the original chain \( P \); a half-step move of the second kind corresponds to one step of the projection chain \( P_\perp \). In practice, by virtue of the fact that \( P_\perp^t = (PP_\perp)^t = PP_\perp^t \), it is not necessary to perform half-steps of the second kind until the last step of the walk. For future reference, we speak of this final half-step \( P_\perp \) as homogenizing the chain within lumps.

For co-occurrence matrices, this gives two versions of any swap walk—the original version and the homogenized version. To carry out \( t \) steps of the homogenized version, first carry out \( t \) steps of the original. Then choose, uniformly at random, one element of the group of row and column permutations discussed in Proposition 4 and apply it to the current state of the chain to get \( p^{(0)}P_\perp \).

6. SWAPPING LARGER SUBMATRICES. To this point, transitions between co-occurrence matrices have relied on swapping \( 2 \times 2 \) submatrices. Such swaps are simple, because a swappable \( 2 \times 2 \) matrix has only one other counterpart with the same marginal totals. However, \( 2 \times 2 \) swaps carry the disadvantage that typical matrices have few swappable submatrices. Consider Darwin's finches: only 333 of 10,608 submatrices are swappable, which means that on average it takes about thirty-two tries to produce each swappable submatrix.

One strategy for increasing the chance of drawing a swappable submatrix is to work with larger submatrices. Not only do \( 3 \times 3 \) swaps increase the chance of drawing a swappable matrix; they also offer more possible destinations per swap. Whereas a swappable \( 2 \times 2 \) matrix has only one \( 2 \times 2 \) counterpart with the same row and column sums, a swappable \( 3 \times 3 \) matrix can be one of as many as six variants with the same margins (e.g., the six permutation matrices of order 3). In principle at least, using \( 3 \times 3 \) swaps offers a way to accelerate convergence by taking “longer” steps, that is, steps that correspond to more than one \( 2 \times 2 \) swap. It is easy to check that every \( 2 \times 2 \) swap is an instance of some \( 3 \times 3 \) swap, from which it follows that \( 3 \times 3 \) swaps connect the matrices of \( A(r, e) \).

Consider the simple example that arises from regarding the \( 3 \times 3 \) identity matrix as a co-occurrence matrix. In all, there are six \( 3 \times 3 \) matrices with the same margins, namely, the six permutation matrices of order 3. To solve the ecologists’ problem, we need to create a random walk on this set of six matrices, with a limiting distribution that assigns equal probabilities to all six. Suppose first we use the \( 2 \times 2 \) swap walk. At each step, there are \( \binom{3}{2} \binom{3}{2} = 9 \) possible \( 2 \times 2 \) submatrices, but only three will be swappable, so on average it will take three tries to get a swappable \( 2 \times 2 \). Moreover, with only three swaps possible, there will be two of the six co-occurrence matrices that can’t be reached from the current matrix in just one step. The limited number of possible swaps means that the \( 2 \times 2 \) swap mixes rather slowly. To get faster mixing, we can use \( 3 \times 3 \) swaps: replace a randomly chosen \( 3 \times 3 \) submatrix with any other having the same margins. For this simple example, this means we can get to any of the six permutation matrices in a single swap. If we allow staying in place as one of the possibilities, then the \( 3 \times 3 \) swap corresponds to a walk on the complete graph of order 6, and “convergence” to a uniform distribution is immediate.

Using \( 3 \times 3 \) swaps, we can define analogues of the various \( 2 \times 2 \) swaps: a basic \( 3 \times 3 \) swap (Algorithm 33), whose limiting distribution is not generally uniform; two holding algorithms (33H and 33H2), and a Metropolized version (33M).

**Algorithm 33**

1. **Random.** Choose three rows uniformly at random without replacement; choose three columns uniformly at random without replacement.
2. **Swap.** Is the resulting $3 \times 3$ submatrix swappable?
   
a. No. Go to step 1. (Do not count this as a step of the walk.)
   
b. Yes. Choose uniformly from all $3 \times 3$ matrices having the same margins as the resulting submatrix, but not equal to that matrix.

**ALGORITHMS 33H AND 33H2** Use the same step 1, but replace step 2 with:

2. **Swap or hold.** Is the resulting $3 \times 3$ submatrix swappable?
   
a. No. Stay in place, and count this as a step of the walk.
   
b. Yes. Choose uniformly at random from all $3 \times 3$ submatrices with the same margins as the resulting submatrix, either excluding that matrix itself (33H), or including it (33H2).

Note that **ALGORITHM 33** does not allow self-loops. For 33H2, there are two ways to execute a self-loop: by choosing an unswappable $3 \times 3$ (step 2a), or by choosing a swappable $3 \times 3$, but then choosing that same matrix in the second part of step 2b. (For $2 \times 2$ swaps, there are also two versions of the holding algorithm. The one (22H) given in section 3 holds only at step 2a, and corresponds to 33H. It is possible to hold also at step 2b, either swapping or staying in place with equal chances; this gives an analogue 22H2 of 33H2.) The Metropolis algorithm excludes self-loops from its first two steps, then allows them as part of a Metropolis step.

**ALGORITHM 33M**

1. **Random.** Same as for step 1 of **ALGORITHM 33**.
2. **Proposed.** Execute step 2 of **ALGORITHM 33** to get a proposed swap.
3. **Metropolis.** Let $N_c$ be the number of swappable $3 \times 3$ submatrices of the current state, let $N_p$ be the corresponding number for the proposed state, and let $\alpha = \min\{1, N_c/N_p\}$. Generate a uniform random number $u$ in $[0, 1]$.
   
a. If $u \leq \alpha$, accept: make the swap.
   
b. If $u > \alpha$, reject: stay in place (and count this as a step of the walk).

Computing $N_c$ and $N_p$ will be time-consuming. As a consequence, the Metropolis algorithm will be much slower than suggested by its geometric convergence rate.

**Example 10.** For **ALGORITHM 33**, here is how the transition probability from matrix $A_1 = a$ to matrix $A_2 = b$ is calculated. Excluding self-loops means that the submatrix formed by columns 1–3 is excluded, so each of the other three has probability $1/3$. For each of these, there are five $3 \times 3$ matrices with the given margins; excluding self-loops leaves four equally likely swaps. Conditional on the choice of columns, the chance of a swap leading to $A_2 = b$ is either 0 (if columns 1,2,4 were chosen) or 1/4 (for the remaining two choices).

$$p_{12} = \Pr[\text{cols } 1,2,3] \Pr[A_2 | \text{cols } 1,2,3] + \Pr[\text{cols } 1,2,4] \Pr[A_2 | \text{cols } 1,2,4]$$

$$+ \Pr[\text{cols } 1,3,4] \Pr[A_2 | \text{cols } 1,3,4] + \Pr[\text{cols } 2,3,4] \Pr[A_2 | \text{cols } 2,3,4]$$

$$= 0 + \left(\frac{1}{3}\right) \left(\frac{1}{3}\right) + \left(\frac{1}{3}\right) \left(\frac{1}{4}\right) + \left(\frac{1}{3}\right) \left(\frac{1}{4}\right) = \frac{2}{12}.$$
Proceeding in this way leads to

$$ P = \begin{pmatrix}
0 & 2/12 & 2/12 & 2/12 & : & 2/12 & 1/12 & 2/12 & 1/12 \\
2/12 & 0 & 2/12 & 2/12 & : & 1/12 & 2/12 & 1/12 & 2/12 \\
2/12 & 2/12 & 0 & 2/12 & : & 1/12 & 2/12 & 2/12 & 1/12 \\
2/12 & 2/12 & 2/12 & 0 & : & 2/12 & 1/12 & 1/12 & 2/12 \\
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\
2/16 & 1/16 & 1/16 & 2/16 & : & 0 & 0 & 5/16 & 5/16 \\
1/16 & 2/16 & 2/16 & 1/16 & : & 0 & 0 & 5/16 & 5/16 \\
2/16 & 1/16 & 2/16 & 1/16 & : & 5/16 & 5/16 & 0 & 0 \\
1/16 & 2/16 & 1/16 & 2/16 & : & 5/16 & 5/16 & 0 & 0
\end{pmatrix}. $$

**Proposition 5.** Algorithm 33 defines a reversible Markov chain. Algorithms 33M, 33H, and 33H2 determine Markov chains with symmetric transition matrices.

*Proof.* Denote co-occurrence matrices by $a_i$ and $a_j$ as before, but let $i$ and $j$ refer to the corresponding states of the walk on $A(r, c)$. A $3 \times 3$ submatrix is determined by a choice of three rows $i_1, i_2, i_3$ and three columns $j_1, j_2, j_3$. Let $S = (i_1, i_2, i_3, j_1, j_2, j_3)$ denote such a choice, and denote by $a(S)$ the actual submatrix of $a$ determined by the choice $S$.

Given states $i$ and $j$, let $S_{ij}$ be the set of choices $S$ for which there is a $3 \times 3$ swap from $i$ to $j$, i.e., for which $a_i(S)$ and $a_j(S)$ have the same marginal totals. Note that because any swap can be reversed, $S_{ij} = S_{ji}$. Let $n_i(S)$ be the number of distinct $3 \times 3$ matrices with the same margins as $a_i(S)$, and observe that for $S$ in $S_{ij}, n_i(S) = n_j(S)$. Finally, let $N_i$ be the number of choices $S$ for which there is a swap from $i$ to some other state $j$.

Reversibility for Algorithm 33 follows from two facts: (1) $N_ip_{ij} = N_jp_{ji}$, and (2) $\pi_i = N_i/\sum N_i$. To verify (1) we compute

$$ p_{ij} = \sum_{S \in S_{ij}} P_{i}(S) Pr_i(A_j | S) = \sum_{S \in S_{ij}} N_i^{-1}[n_i(S) - 1]^{-1}, $$

which implies that

$$ N_ip_{ij} = \sum_{S \in S_{ij}} [n_i(S) - 1]^{-1} = \sum_{S \in S_{ji}} [n_j(S) - 1]^{-1} = N_jp_{ji}. $$

To verify (2), let $\pi_i = N_i/\sum N_i$. Then it suffices to show that $\sum \pi_ip_{ij} = \pi_j$:

$$ \sum \pi_ip_{ij} = \sum \frac{N_ip_{ij}}{\sum N_i} = \sum \frac{N_jp_{ji}}{\sum N_i} = \left(\frac{N_j}{\sum N_i}\right)\left(\sum p_{ji}\right) = \pi_j. $$

An immediate consequence of this reversibility is that the Metropolized version of the chain has a symmetric transition matrix. Finally, we remark that the transition matrix
for Algorithm 33H is also symmetric, because all choices \( S \) have equal probabilities regardless of \( i \) and \( j \). Algorithm 33H2 is symmetric for similar reasons.

7. A BRIEF LOOK AT A FEW EXAMPLES. How well do the various strategies work? There seems to be no simple answer. This section provides the bare beginnings of an exploration, based on a set of eight examples that are listed in Figure 9. This choice of examples is somewhat arbitrary. The first five were chosen mainly because they were small enough to be studied without relying heavily on computers. The last three examples were added later in order to extend the range of situations represented. Thus the sixth case (the \( 4 \times 4 \) permutation matrices) was chosen for its symmetry, and the last two examples were chosen to be roughly comparable in size but without the high degree of symmetry. Note that these last two differ strikingly in the number of lumps determined by their row and column symmetries.

<table>
<thead>
<tr>
<th>Row sums ( r )</th>
<th>Column sums ( c )</th>
<th># vertices</th>
<th>Mean degree ( \bar{d} )</th>
<th>Symmetry group ( G )</th>
<th># lumps (orbits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,2,2)</td>
<td>(1,2,2)</td>
<td>5</td>
<td>3.20</td>
<td>( S_2 \times S_2 )</td>
<td>2</td>
</tr>
<tr>
<td>(1,1,3)</td>
<td>(1,1,1,2)</td>
<td>7</td>
<td>4.28</td>
<td>( S_2 \times S_3 )</td>
<td>2</td>
</tr>
<tr>
<td>(1,2,3)</td>
<td>(1,1,2,2)</td>
<td>8</td>
<td>4.50</td>
<td>( S_2 \times S_2 )</td>
<td>3</td>
</tr>
<tr>
<td>(1,2,2)</td>
<td>(1,1,1,2)</td>
<td>12</td>
<td>5.50</td>
<td>( S_2 \times S_3 )</td>
<td>2</td>
</tr>
<tr>
<td>(2,2,2)</td>
<td>(1,1,2,2)</td>
<td>15</td>
<td>6.00</td>
<td>( S_3 \times S_2 \times S_2 )</td>
<td>2</td>
</tr>
<tr>
<td>(1,1,1,1)</td>
<td>(1,1,1,1)</td>
<td>24</td>
<td>6.00</td>
<td>( S_4 \times S_4 )</td>
<td>1</td>
</tr>
<tr>
<td>(2,2,2,3)</td>
<td>(1,2,3,3)</td>
<td>27</td>
<td>7.93</td>
<td>( S_3 \times S_2 )</td>
<td>2</td>
</tr>
<tr>
<td>(1,2,3,4,5)</td>
<td>(1,2,3,4,4)</td>
<td>39</td>
<td>7.54</td>
<td>( S_2 \times S_2 )</td>
<td>24</td>
</tr>
</tbody>
</table>

Figure 9. An eight-example comparison.

Figure 10 presents a comparison based on convergence rates, as measured by the second largest eigenvalue modulus (\( \text{SLEM} = \theta_2 \)). For each of the eight examples, values of \( \theta_2 \) for sixteen Markov chains are shown, as described in the caption to the figure.

A few patterns stand out. None really deserves to be called surprising. If anything, it is the exceptions that are more interesting.

1. Homogenizing. In accordance with the results of section 5, homogenizing either speeds convergence (lower \( \theta_2 \)) or leaves it unchanged. The effect is most pronounced for the set of \( 4 \times 4 \) permutation matrices, which constitute a single lump and for which homogenizing gives the limiting distribution in a single step. For the 39-vertex example, there are twenty-four lumps, and homogenizing has no detectable effect. One might expect a general pattern consistent with these extreme examples, namely, the fewer the lumps, the greater the reduction in \( \theta_2 \) brought about by homogenizing. Surely this must be true in some sense, but note that the 27-vertex example can be reduced to just two lumps, yet the effect of homogenizing is quite small.

2. Holding. The holding algorithms tend to be slower than either the basic swaps or the Metropolized swaps. Moreover, as might be expected, holding at both stages tends to give slower convergence than holding only at the first stage. For \( 2 \times 2 \) swaps, there are no exceptions to these patterns among the eight examples. For \( 3 \times 3 \) swaps, there are four examples for which the swap that holds at both stages is in fact the fastest of the \( 3 \times 3 \) swaps, thereby violating both patterns. These four examples (the ones with 5, 8, 12 and 24 vertices) are the ones with the most symmetry due to repeated row and column sums.
3. **Submatrix size.** Because $3 \times 3$ swaps “take in more territory” than $2 \times 2$ swaps, one might expect the larger swaps to move more quickly among co-occurrence matrices and each $3 \times 3$ swap algorithm to converge more rapidly than its $2 \times 2$ counterpart. Figure 10 shows that this is indeed the case most of the time, but there are two sets of exceptions. For the 8-vertex and 12-vertex examples, the basic $2 \times 2$ swap and its Metropolized version converge faster than their respective $3 \times 3$ counterparts. Here, as with holding, the exceptions to the general patterns come from examples with the most symmetry due to repeated marginal sums.

4. **Metropolizing.** On the surface, the results appear puzzling. Because the Metropolis algorithm adds self-loops to a Markov chain, one might naturally expect the self-loops to retard mixing, and reduce the convergence rate. Thus we would see $\theta_s$ increase in going from the third to fourth pair of points within each set of four. Inspection of Figure 10, however, shows that this is almost never the case among the examples studied. The explanation? Adding self-loops increases the values on the diagonal of $P$, moving $P$ toward the identity $I$, and thus, by continuity, moving the eigenvalues of $P$ toward the eigenvalues of $I$, i.e., toward 1. If the eigenvalue of $P$ having the second largest modulus is positive, shifting the eigenvalues of $P$ toward 1 will increase $\theta_s$. If, however, the eigenvalue with the second largest modulus is negative, the shift toward 1 may well reduce $\theta_s$. All the exceptions in Figure 10 are accounted for by this phenomenon.

8. **CONCLUSION.** Three sets of observations provide a summary of sorts. First, from a purely practical viewpoint, the methods described here for generating random matrices are substantially slower than the method of sequential importance sam-

---

Figure 10. Geometric convergence rates for 128 Markov chains: Each panel shows the second largest eigenvalue modulus (SLEM) for sixteen Markov chains. Each pair of points joined by a vertical segment corresponds to a pair of chains, one homogenized (lower point) and one not. From left to right within each panel, the eight pairs correspond to four $2 \times 2$ swaps (solid dots) and four $3 \times 3$ swaps (open circles). Within each set of four, the order is: ‘2’ = holding at both stages; ‘h’ = holding at the first stage only; ‘s’ = basic swap; ‘m’ = Metropolized swap. Note that in general, values of $\theta_s$ decrease from left to right within each set of four, although there are several exceptions.
plung [11]. (To date, however MCMC has proved to be adaptable to a vastly broader range of applications than has sequential importance sampling.)

Second, the results reported here leave unanswered a number of questions relating the symmetries of $A(r, e)$ to the convergence properties of the various swap walks: given a set $(r, e)$ of row and column sums, and thus a group of symmetries on $A(r, e)$, we are led to ask:

- What determines the size of the effect that homogenizing will have on $\theta_+$?
- What determines whether the one-stage holding algorithm will converge faster than the two-stage version?
- Under what conditions on $r$ and $e$ will the $2 \times 2$ swap outperform the $3 \times 3$?
- Under what conditions will Metropolizing accelerate rather than retard convergence of the basic swap?

Finally, what about the finches? Is the pattern of their distribution on the islands of the Galapagos different in any substantial way from what we could expect from a purely random (uniform) distribution? A thoughtful look at the co-occurrence matrix suggests the answer should be yes: the 1s show what looks like a strong tendency to concentrate in the upper left, with the 0s concentrated in the lower right. Granted, this is due in part to the ordering of rows and columns so that marginal sums are monotone. Even so, it is much easier to find swaps that reduce the number of checkerboards, rather than increase that number. Similarly, it is much easier to eliminate existing checkerboard units than to create new ones.

These facts are reflected in the $p$-values, obtained by applying any of the Metropolized (M) or holding (H) algorithms. The minimum number of burn-in steps needed for estimating the $p$-values remains an open question, but one can simply be conservative and use tens of thousands of steps. The chance that a random matrix has at least ten checkerboards is only 0.03; so is the chance of thirteen or fewer species combinations. The chance of 333 or more checkerboard units is 0.0002. Conclusion: the distribution of finch species among the Galapagos shows evidence of competition.

REFERENCES


**GEORGE W. COBB** was an undergraduate Russian major (A.B. 1968 Dartmouth) before turning to statistics in graduate school (M.S. 1971, Medical College of Virginia; Ph.D. 1974 Harvard). He maintains a precarious professional existence on the cusp between the scholarly and the sophomoric, where his recent unpublished efforts include “Three Ways to Gum Up a Statistics Course,” a seminal paper on the emerging science of confectionery ballistics (a.k.a. launching Gummy Bears), and “Brains, Slime and Elasticities,” which uses Elmer’s glue to cement a long suspected connection between econometricians and paleontologist Stephen Jay Gould.

*Department of Mathematics and Statistics, Mount Holyoke College, South Hadley, MA 01075
gcobb@mtholyoke.edu*

**YUNG-PIN CHEN** received his Ph.D. in statistics from Purdue University. He taught at Occidental College, Pomona College, and Smith College before coming to Lewis & Clark College in 2002. He is fascinated by how one can use randomness to study certainty. His research interests include sequential designs, stochastic processes, and probabilistic number theory.

*Department of Mathematical Sciences, Lewis & Clark College, Portland, OR 97219
ychen@lclark.edu*