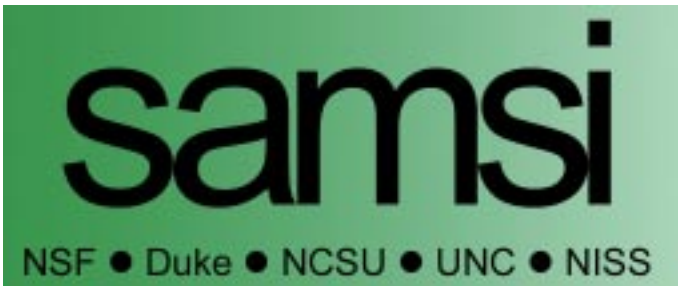

Monte Carlo algorithms for Hardy-Weinberg Proportions

Mark Huber, Yuguo Chen, Ian Dinwoodie,
Adrian Dobra and Mike Nicholas

Technical Report #2003-8
May 2003

This material was based upon work supported by the National Science Foundation under Agreement No. DMS-0112069. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Statistical and Applied Mathematical Sciences Institute
PO Box 14006
Research Triangle Park, NC 27709-4006
www.samsi.info

Monte Carlo algorithms for Hardy-Weinberg Proportions *

By MARK HUBER
Department of Mathematics and ISDS, Duke University, Durham, NC 27708-0320, USA
mhuber@math.duke.edu

YUGUO CHEN
Institute of Statistics and Decision Sciences, Duke University, Durham, NC 27708-0251, USA
yuguo@stat.duke.edu

IAN DINWOODIE†
SAMSI, Research Triangle Park, NC 27709-4006, USA
ihd@stat.duke.edu

ADRIAN DOBRA
Institute of Statistics and Decision Sciences, Duke University, Durham, NC 27708-0251, USA
adobra@stat.duke.edu

and MIKE NICHOLAS
Department of Mathematics, Duke University, Durham, NC 27708-0320, USA
jefe@math.duke.edu

Summary

The Hardy-Weinberg law is of basic importance in studying biological systems, and it is important to be able to determine if a population is in Hardy-Weinberg equilibrium. For finite populations, this means testing if they are a draw from a distribution known as Hardy-Weinberg proportions. Because the state space is exponentially large in the population size, the only efficient means for doing this is Monte Carlo simulation. Here we explore three different Monte Carlo techniques for this problem. To begin, we give the first linear time algorithm for generating random variates exactly from the desired distribution in the absence of extra constraints such as structural zeros. Second, we build and test Markov chains for this problem that work in the presence of structural zeros. Finally, we design and explore the behavior of a sequential importance sampling technique.

Some key words: Hardy-Weinberg; Monte Carlo; Direct Sampling; Exact p-value; Sequential Importance Sampling; Gröbner Basis

*Research reported was partially supported by the National Science Foundation through SAMSI under DMS-0112069. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

†Supported by NSF Grant DMS-0200888

1 Hardy-Weinberg Proportions

The Hardy-Weinberg law is a cornerstone of population genetics, and it is quite important to be able to test whether or not a given population follows Hardy-Weinberg proportions (HWP) (for example see Galbusera et al.(2000)). Consider a particular autosomal locus that is always one of m different alleles in the population. Suppose that allele A_i occurs with frequency p_i for i from 1 to m. Hardy-Weinberg operates under the assumption that the population mates randomly and that there is no selection or mutation affecting the gene frequencies.

Under these assumptions, the Hardy-Weinberg law states that allele combination A_iA_j (where $i < j$) occurs with frequency $2p_ip_j$ in the population, and A_iA_i occurs with frequency p_i^2. These frequencies are limiting behavior for a large population. Now consider a finite population of N individuals and a particular autosomal locus. Since the locus is autosomal the genetics of the population can be described completely by writing out the $2N$ allele types found in the N members of the population. We will say that the population is in Hardy-Weinberg equilibrium or exhibits Hardy-Weinberg proportions if this ordering is equally likely to be any permutation of the $2N$ alleles found in the population.

For our purposes, two individuals possessing the same allele combination at the locus of interest are indistinguishable from one another, and so we are actually only interested in the observed frequency of each pair of alleles.

Suppose that allele A_i occurs f_i times in the population. This is fixed, as any permutation of the alleles can not change the f_i. Let f_{ji} be the number of times allele combination A_jA_i $(i < j)$ occurs. The f_{ij} can change with the permutation, although we note that for all i from 1 to m:

$$f_i = 2f_{ii} + \sum_{j>i} f_{ji} + \sum_{j<i} f_{ij}. \tag{1}$$

Restricting ourselves to frequencies that meet the linear constraints in (1), the distribution of frequencies becomes:

$$\pi(f) = \frac{N!}{(2N)!}\left[\prod_{i\le j}\frac{1}{f_{ji}!}\right]\left[\prod_{i=1}^{m} f_i!\right] 2^{\sum_{i<j} f_{ji}}. \tag{2}$$

To see this, just count the number of permutations that have a given frequency. First, given the allele counts, the number of ways to assign pairs of alleles to people will be

$$\left(\frac{N!}{f_{11}!f_{21}!\cdots f_{mm}!}\right).$$

Once the alleles have been assigned to each member of the population, for $i \neq j$ the allele may be ordered A_iA_j or A_jA_i, giving two ways for each member assigned different alleles. Hence there are 2^d, where $d = \sum_{i<j} f_{ji}$, ways to order these alleles. Each permutation has probability $\prod_{i=1}^{m} f_i!/(2N)!$, which gives (1).

Various tests exist to determine if a particular data set follows HWP. Large sample goodness of fit test such as Pearson's χ^2 (Li, 1955) rely on asymptotic results that are not always valid for the data sets of the type we consider. Guo & Thompson(1992) proposed what they called an exact p test, however, evaluting their p value requires the ability to generate samples from the distribution π in (2). It is this problem of generating random variates (and using such variates efficiently) that will be our focus in this paper.

Slow and fast direct generation of random variates Given that we derived π in (2) from the uniform permutation on $2N$ alleles, we can generate a random variate directly from π by first generating a random permutation of the $2N$ alleles and then just counting the frequencies f_{ij} directly. We will refer to this as the *naive Monte Carlo* method for this problem.

At first glance this appears to be an effective method, but closer examination reveals that it is actually an exponential algorithm. Generation of the permutation of $2N$ elements requires $2N$ space and $2N$ draws of random uniforms. We are only interested in m choose 2 or $\Theta(m^2)$ different frequencies, but N might be exponentially large in the size of the problem instance. Hence directly forming the permutation is actually an exponential algorithm for the problem.

This has led various authors to consider Markov chains for this problem. The drawback is that the Markov chain mixing time used in papers such as Guo & Thompson(1992) have mixing time $O(N \ln N)$, and so are actually worse than just directly drawing the permutation in the first place with respect to time. On the other hand, they do not have the $2N$ space requirement, but only require $\Theta(m^2)$ space.

In Section 3 we present two true linear time algorithms that only require $\Theta(m^2)$ time to generate a random variate from the desired distribution, with no dependence on N. We compare our algorithms to the naive MC method and show that it is much faster in practice as well as theory.

Structural zeros A drawback of both the basic Markov chain method and direct sampling techniques is that they have limited ability to handle the presence of allele combinations that are forbidden for biological reasons. When f_{ij} is constrained to be zero, we refer to it as a *structural zero*. In Section 4 we will discuss ways to construct more complex Markov chains that deal with this issue by adding moves in order to connect the state space. Finally, in Section 5 we employ another technique, sequential importance sampling. This method can also be utilized in the presence of structural zeros to estimate the exact p value.

2 The Problems

In this section we introduce several data sets that are typical for this field, along with the exact p value test of Guo & Thompson (1992).

Example A1. Rhesus data

The following data set was extracted (Guo & Thompson, 1992) from Rhesus data in Cavalli-Sforze and Bodmer's *The Genetics of Human Population* (1971).

1236								
120	3							
18	0	0						
982	55	7	249					
32	1	0	12	0				
2582	132	20	1162	29	1312			
6	0	0	4	0	4	0		
2	0	0	0	0	0	0	0	
115	5	2	53	1	149	0	0	4

For this table f, $\pi(f) = 10^{-54.7}$. The exact p value of Guo & Thompson(1992) works by computing the probability that a random table chosen from π has probability at most $\pi(f)$.

More precisely, let $\mathbf{F} = (f_1, \ldots, f_m)$ and let

$$S_{\mathbf{F}} = \{g = (g_{ij}) : 2g_{ii} + \sum_{j>i} g_{ji} + \sum_{j<i} g_{ij} = f_i \ \ \forall i\}. \tag{3}$$

Then the exact p value is just

$$p = \sum_{g \in S_{\mathbf{F}} : \pi(g) \leq \pi(f)} \pi(g) \tag{4}$$

This data set was analyzed by Guo & Thompson(1992) using both χ^2 and the exact p value. Their analysis of the exact p value was done using Monte Carlo techniques, and so the speed of the analysis is directly proportional to the speed at which random variates can be generated.

Example A2. Retinoblastoma RB1-VNTR genotype data

This data comes from the Ceph database (http://www.cephb.fr) and is based on 44 families. The gene is located on chromosome 13b and there are 7 allele types:

13						
115	175					
16	34	3				
0	16	0	0			
2	4	1	0	1		
35	65	3	1	2	22	
11	16	1	0	0	2	3

The fitted table using maximum likelihood estimates $\hat{p}_i = f_i/(2N)$ is:

19.42						
113.68	166.36					
11.56	33.83	1.72				
3.22	9.43	0.96	.0.13			
2.08	6.10	0.62	0.17	0.06		
28.80	84.29	8.57	2.39	1.55	10.68	
6.82	19.96	2.03	0.57	0.37	5.06	0.60

A goodness of fit test on the Hardy-Weinberg equilibrium family gives a χ^2 statistic of 69.81 on an asymptotic χ^2 distribution with 28 - 1 - (7 - 1) = 21 degrees of freedom, for a p value of 3.8×10^{-7}. On the other hand, the exact p value can be found (using the methods described in the remained of this paper) to be 7.2×10^{-5}.

Example A3. Gaucher disease data.

This genotype data comes from le Coutre et al.(1997). In Beutler et al.(1995), it is stated that the genotype pair IVS2+1/IVS2+1 is lethal, and therefore constitutes a structural zero in the triangular table of genotypes. The genotype data is as follows:

0						
5	2					
2	0	0				
1	0	0	–			
0	0	0	0	0		
0	1	0	0	0	0	
10	2	0	0	1	0	1

where the - indicates the presence of a structual zero.

3 Fast generation of random variates

We now present an algorithm for drawing directly from the Hardy-Weinberg proportions using $m(m+1)/2$ draws from various hypergeometric distributions. Each hypergeometric draw can be accomplished using 2.65 (Stadlober & Zechner, 1999) (in expectation) uniform random variables, and so will be considerably faster than previous methods. We also present a second algorithm that converts the orignal problem of sampling from HWP to the problem of sampling a regular contingency table where row and column sums are fixed.

For instance, Guo & Thompson(1992) performed a Monte Carlo study on the Rhesus data of the previous section, where $N = 8297$, and $m = 7$. Their method for direct draws is what we have been referring to as naive Monte Carlo to distinguish it from the more complex Monte Carlo methods introduced in this section. This method requires 8192 draws of uniform random variables to generate one permutation which in turn gives one sample of frequencies, whereas our method takes only 28 hypergeometric draws (or roughly 75 uniform draws) for a complete set of frequencies, a speedup of over a factor of 100.

Consider writing the frequencies in a lower triangular matrix. Our method fills in the matrix one column at a time from left to right. For a given column, we first tackle a diagonal entry such as f_{11}. This will require two draws of a hypergeometric. Then we fill in each element in the column below, each with a single hypergeometric. Once a column is filled, the numbers for the remaining alleles can be updated accordingly. The process then repeats until we have entered figures in all columns.

Hypergeometric random variables Our main tool in directly generating frequencies will be the ability to generate hypergeometric random variables. Consider the problem of dropping t balls uniformly at random into $s = r + g$ slots where r of the slots are colored red, and g of the slots are colored green. Then if X is the number of balls that fall in red slots, then we say that X has a hypergeometric distribution with parameters s, r, and t. We write $X \sim HG(s, r, t)$.

Several acceptance/rejection methods exist for generating from the hypergeometric distribution in constant time. One method uses an envelope function that is a constant flanked by two exponentials (Kachitvichyanukul & Schmeiser, 1985) while Stadlober(1990) develops a ratio of uniforms method that was later improved by Stadlober & Zechner(1999). All of these methods generate random hypergeometric variates in time constant in the parameters. In particular, computer experiments in Stadlober & Zechner(1999) indicate that at most 2.65 uniform variates are needed (on average) for a single hypergeometric draw.

Column by column Our algorithm proceeds by filling in the columns of the matrix one at a time. We now verify that this procedure is valid. Suppose that we are trying to generate a random permutation of the alleles conditioned on the first column of frequencies being fixed. That is, the counts f_{11} through f_{1m} are known.

Such a permutation can be constructed in the following fashion. For each member with allele A_1A_j, $j \neq 1$, randomly permute the genotype so that it is either A_1A_j or A_jA_1. Now take each of the f_{1j} members with genotype A_1A_j or A_jA_1 and randomly place them among the N members of the population. Finally, for the remaining alleles, randomly place them among the $2(N - f_{11} - \ldots - f_{1m})$ slots that remain.

Any permutation satisfying f_{11} through f_{1m} can be constructed in this fashion, and each of these permutations will have probability

$$\frac{1}{\prod_{j=2}^{m} 2^{f_{1j}}} \cdot \frac{N!}{\prod_{j=1}^{m} f_{1j}!} \cdot \frac{1}{[2(N - f_{11} - \ldots - f_{1m})]!} \tag{5}$$

of occuring. This is constant when conditioned on f_{11} through f_{1m}, and so this is a valid procedure for generating permutations uniformly at random conditioned on the values in the first column.

This gives us the following procedure. First fill in the first column of the matrix. Then update the numbers of alleles f_2 through f_m by subtracting off f_{12} through f_{2m} respectively. Finally, update N by subtracting $f_{11} + \ldots + f_{1m}$. The result is a new problem that we solve recursively. Since the base case where we have no columns is easy to sample from, we can prove via induction that this procedure generates directly from π.

Diagonal entries f_{ii} Suppose that we have generated frequencies for the first $i-1$ columns, and the genotypes of N members of the population are still unknown. We wish to randomly choose the number of members of the population that end up with A_iA_i genotype. Some of the A_i alleles end up being assigned to the first allele for a member, and some end up being the second allele. Equivalently, we are dropping f_i alleles into $2N$ slots, N correspond to the first alleles. We will call these the first slots. Similarly, N of which correspond to the second alleles, and we shall refer to these as the second slots. Hence the number in the first slot will be distributed $HG(2N, N, f_i)$. Call this number a_i.

Conditioned on a_i, precisely $f_i - a_i$ alleles lie in the N second slots. We think of these $f_i - a_i$ alleles as being uniformly distributed among the N second slots. Exactly a_i of the first slots are occupied by A_i alleles, and so again we have a hypergeometric situation, where the number of A_i alleles occupying second slots with the corresponding first slot also occupied is $HG(N, a_i, f_i - a_i)$.

Off diagonal entries Once we have generated f_{ii}, we turn our attention to generating the values that lie in the off diagonal entries in column i. That is, we fill in f_{ji} as j runs from $i+1$ through m.

Conditioned on f_{ii}, the remaining $f_i - 2f_{ii}$ type A_i alleles all are placed in separate members of the population. Suppose that we have filled f_{ii} through $f_{(j-1)i}$. Let f_i' be the remaining A_i alleles that are still unaccounted for. Let $b = 2N - 2f_{ii} - f_i'$ be the number of slots that we still have not assigned any alleles to.

Then the number of type j alleles that fall in the f_i' slots where one slot is occupied by an A_i allele is again hypergeometric, and $f_{ji} \sim HG(b, f_i', f_j)$. Once we have drawn f_{ji}, we must update b, f_i', and f_j.

First, the number of free slots b is reduced by f_j; this is because each A_j allele occupies a slot, whether or not it is paired with an A_i allele. Next we reduce f_i' and f_j by f_{ji} since we have used up f_{ji} of the A_i and A_j alleles.

We continue filling in f_{ij} as j goes from $i+1$ up to m. Once we have filled in the entire column, we note that $\sum_{j \geq i} f_{ji}$ members of the population have been assigned allele combinations.

Then just move to the next column until the entire table has been determined. To avoid evaluating sums, we actually reduce N by $f_i - f_{ii}$ before filling in the the remainder of the column. The algorithm can be represented using the following pseudocode.

Putting this all together, the algorithm looks like this:

Generating from Hardy-Weinberg Proportions

Input: $N, f_1, \ldots, f_m$

1. **For** i from 1 to m
2. **Choose** $a_i \leftarrow HG(2N, N, f_i)$
3. **Choose** $f_{ii} \leftarrow HG(N, a_i, f_i - a_i)$
4. **Let** $N \leftarrow N - (f_i - f_{ii})$
5. **Let** $f_i \leftarrow f_i - 2f_{ii}$
6. **Let** $b \leftarrow 2N - f_i$
7. **For** j from $i + 1$ to m
8. **Choose** $f_{ji} \leftarrow HG(b, f_i, f_j)$
9. **Let** $b \leftarrow b - f_j$
10. **Let** $f_j \leftarrow f_j - f_{ij}$
11. **Let** $f_i \leftarrow f_i - f_{ij}$

Dealing with a diagonal structural zero When one of the elements of the table is constrained to be zero, we call such a situation a structural zero. More precisely, we wish to sample from the same distribution conditioned on some of the entries having value zero. As long as only a single diagonal structural zero exists, we can easily modify the method to generate samples in the same time.

Say without loss of generality that $f_{11} = 0$ (just permute the gene labels if this is not the case.) Then we wish to sample from permutations where each of the A_1 alleles reside in different members of the population. But the rest of the algorithm just samples from the distribution conditioned on the value of f_{11}. Therefore, setting $f_{11} = 0$ and then completing the rest of the algorithm as before generates from the correct distribution.

Method 2 Another approach is to convert the problem from sampling HWP with m alleles to a problem of sampling from the hypergeometric distribution on m by m contingency tables where the constraints take the form of fixed row and column sums. Generating such contingency tables takes m^2 draws from a hypergeometric distribution, and so will take roughly twice as long as generating the HWP tables directly. However, other techniques for approximately counting interesting contingency tables exist. With this conversion approach, those techniques can be applied to the HWP tables in a hybrid Monte Carlo fashion.

By m by m contingency tables with fixed row and column sums, we mean the set of tables n with nonnegative integer entries such that $\sum_j n_{ij} = r_i$ and $\sum_i n_{ij} = c_j$ for fixed vectors $(r_1, \ldots, r_m)$ and $(c_1, \ldots, c_m)$. The distribution we wish to sample from is hypergeometric:

$$\pi'(n) = \frac{r_1! \cdots r_m! c_1! \cdots c_m!}{\left(\prod_{ij} n_{ij}!\right) n!} \tag{6}$$

where $n = \sum_{ij} n_{ij}$ is a fixed constant.

As with the HWP hypergeometric distribution, this can be thought of as the result of a permutation of labeled objects. In this case, we have n slots for the permutation, r_1 of which are labeled 1, r_2 of which are labeled 2, and so on up to r_m. We also have n items, c_1 of which are labeled 1, up to c_m. We randomly permute the items in the slots. Entry n_{ij} is the number of items labeled j that end up in a slot labeled i.

The idea is this: again treat the population of N members as a collection of N first alleles and N second alleles. Assign for each of the m alleles the number that land in first positions and those that land in second positions. Once these numbers are determined, we have reduced

the problem to choosing a random permutation of the second alleles and matching them up with the first alleles. This is exactly the same as our permutation approach to contingency tables with fixed row and column sums: the row sums are just the numbers of each allele type that occupy first alleles, and the column sums are the numbers of each allele type that occupy second alleles.

Once we have chosen a random set of row and column sums, we then just choose a random contingency table from the hypergeometric distribution to finish the procedure. It is well known how to do this using m^2 hypergeometric draws. In the pseudocode below, we use a_i to denote the number of alleles of type A_i that are assigned to first alleles, which makes $b_i = f_i - a_i$ the number that are in second alleles.

Generating from Hardy-Weinberg Proportions

`Input:` $N, f_1, \ldots, f_m$

1. **For** i from 1 to m
2. **Choose** $a_i \leftarrow HG(2N, N, f_i)$
3. **Let** $b_i \leftarrow f_i - a_i$
4. **For** i from 1 to m
5. **Let** $temp \leftarrow N$
6. **Let** $N \leftarrow N - b_i$
7. **For** j from 1 to m
8. **Choose** $n_{ji} \leftarrow HG(temp, a_i, b_i)$
9. **Let** $temp \leftarrow temp - a_j$
10. **Let** $b_i \leftarrow b_i - n_{ji}$
11. **Let** $a_j \leftarrow a_j - n_{ji}$
12. **For** i from 1 to m
13. **For** j from i to m
14. **Let** $f_{ji} \leftarrow n_{ij} + n_{ji}$

Running Time The number of hypergeometric draws employed by Method 1 is m choose 2 plus m, or $(m^2 + m)/2$. The number of hypergeometric draws in Method 2 is $m + m^2$. Given that we can take a hypergeometric draw on average using 2.65 uniform draws Stadlober & Zechner(1999), we immediately have the following theorem.

THEOREM 1. *An upper bound on the average number of uniforms used by Method 1 is* $1.33(m^2 + m)$*, while for Method 2 an upper bound is* $2.65(m^2 + m)$

Of course this method will only be better as N gets larger. Consider Example 1, the Rhesus data. Here $N = 8192$. We can derive larger artifical data sets from this one by simply multiplying the elements of the table by integer multiples. Figure 1 shows the running time for Method 1, Method 2, and the naive Monte Carlo methods for various multiples of the data set.

In Figure 1, the data size is the value by which the original data was multiplied to get the artificial data. Note that even for the original data, Method 1 only takes roughly 0.5 ms per sample, while the naive method takes about 5 so the speedup is a factor of 10. The right panel shows only the running time for our new method. As the figure shows, the running time for the naive method increases linearly, but the running time for the hypergeometric methods actually decrease slightly! This is most likely due to the probability of acceptance in the hypergeometric generator going up as the numbers in the data increase.

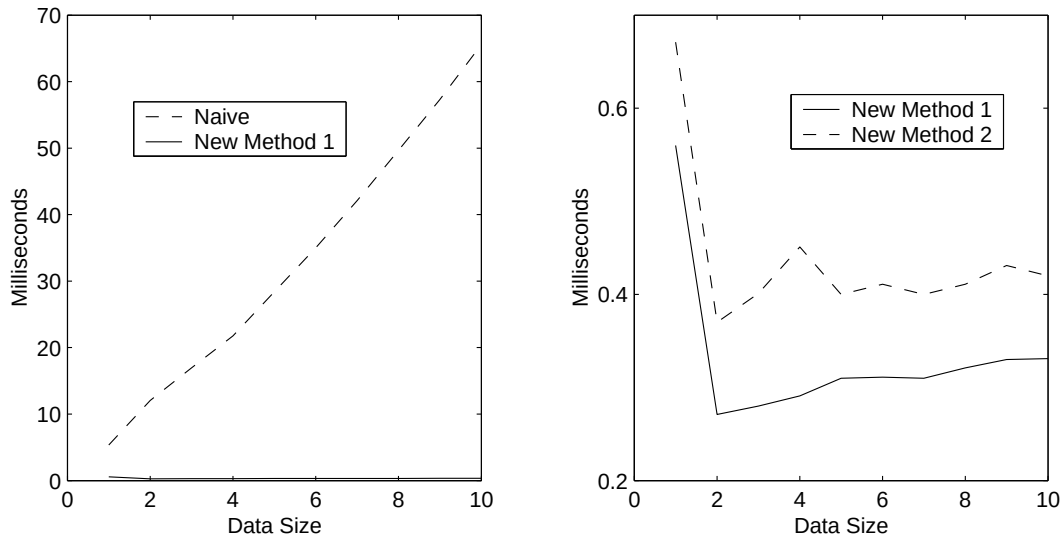

Figure 1: Running time of naive and new variate generation techniques

4 Markov Chains

Let $T_m = \{(i,j) : 1 \leq j \leq i \leq m\}$ be the set of positions in our lower triangular tables, and let $I_0 \subseteq T_m$ be a set of positions for structural zeros. Let $\Gamma(\mathbf{F}, I_0)$ be the set of tables that have the entries in I_0 zero, so that

$$\Gamma(\mathbf{F}, I_0) = S_{\mathbf{F}} \cap \{g : g_{ij} = 0 \quad \forall (i,j) \in I_0\}. \tag{7}$$

For $\pi(\Gamma(\mathbf{F}, I_0)) > 0$, we are interesting in sampling on the conditional distributon

$$\pi_{I_0}(f) = \pi(f | f \in \Gamma(\mathbf{F}, I_0)) = \pi(f) 1_{\Gamma(\mathbf{F}, I_0)}(f) / \pi(\Gamma(\mathbf{F}, I_0)). \tag{8}$$

When I_0 is a single diagonal element, the method of Section 3 can be used. In general, one can run an irreducible symmetric Markov chain together with a Metropolis rejection step for simulation.

In practice, structural zeros will usually be on the diagonal, because impossible combinations are typically lethal recessive pairs, but the Markov chains described work for any collection of zero entries I_0. While the moves for the Markov chain can be found in an *ad hoc* way with methods of Diaconis & Sturmfels(1998), which in fact we do with computational algebra software to minimize the number of necessary moves, it is possible to describe a collection of moves that will always give an irreducible Markov chain for any location of the zero entries. The method is a slight extension of the one for zero entries in a rectangular table with fixed row and column sums.

Consider the complete bipartite graph $K_{m:m}$. A circuit c will move from one side of the partition to the other, so $c = ((i_1, j_1), (i_2, j_1), (i_2, j_2), \ldots, (i_{k-1}, j_k), (i_1, j_k))$ with $2k$ edges where the $\{i_\alpha\}$ and $\{j_\alpha\}$ lie in different partitions.

The circuit c corresponds to the vector $v_c \in Z^{m^2}$ by

$$v_c := e_{i_1 j_1} - e_{i_2 j_1} + e_{i_2 j_2} - \ldots + e_{i_{k-1} j_k} - e_{i_1 j_k} \tag{9}$$

which has row and column sums of zero. Let $\mathcal{C}_m$ be the collection of all circuits. Since we use the notation above we have that cyclical permuations of a circuit give the same vector up to a factor of ± 1, and so this collection is slightly redundant.

For a vector $v \in Z^{m^2}$ define its folded image $F(v)$ to be the vector in Z^{T_m} given by

$$\begin{aligned} F(v)_{ij} &= v_{ij} + v_{ji}, j < i \\ F(v)_{ii} &= v_{ii} \end{aligned}$$

Note that F is linear and that $F(v) = F(v')$.

THEOREM 2. *The Markov chain in $\Gamma(\mathbf{F}, I_0)$ formed of vector increments*

$$\{\pm F(v_c) : F(v_c) \text{ has support in } T_m \setminus I_0, c \in \mathcal{C}_m\} \tag{10}$$

is irreducible.

Remark 1. These moves are also described in Takemura & Aoki(2002), although their argument for irreducibility is different.

Proof. Let $\mathbf{m}$ and $\mathbf{n}$ be two triangular tables that we want to connect through $\Gamma(\mathbf{F}, I_0)$. Form two square tables $\mathbf{m}^s$ and $\mathbf{n}^s$:

$$\begin{aligned} \mathbf{m}^s_{ii} &= 2\mathbf{m}_{ii} \\ \mathbf{m}^s_{ij} &= \mathbf{m}_{ij}, j < i \\ \mathbf{m}^s_{ij} &= \mathbf{m}_{ji}, j > i \end{aligned}$$

and similarly for $\mathbf{n}$. Since both of $\mathbf{m}$ and $\mathbf{n}$ are in $\Gamma(\mathbf{F}, I_0)$, both $\mathbf{m}^s$ and $\mathbf{n}^s$ will have the same row sums a and columns sums b.

Now connect the two tables $\mathbf{m}^s$ and $\mathbf{n}^s$ with k moves from circuits which avoid the zero positions I_0 and I_0' and which preserve the row i sum a_i and the column j sum b_j. Say the connecting sequence is $\mathbf{m}^s = \mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_k = \mathbf{n}^s$, where each move at step i is $\mathbf{x}_{i-1} - \mathbf{x}_i = v_{c_i}$.

Since $\mathbf{m}^s$ and $\mathbf{n}^s$ are symmetric, they can also be connected with the symmetric sequence $\mathbf{m}^s = \mathbf{x}^s_0, \mathbf{x}^s_1, \ldots, \mathbf{x}^s_k = \mathbf{n}^s$, where the moves are now $\mathbf{x}^s_{i-1} - \mathbf{x}^s_i = v_{c_i} + v'_{c_i}$. The new sequence of states is nonnegative, by matrix symmetry. Now consider the sequence of triangular states $\mathbf{y}_i = F(\mathbf{x}^s_i)/2$ for i from 1 to k. The differences in the $\mathbf{y}$ sequence are moves $F(v_{c_i})$:

$$\mathbf{y}_{i-1} - \mathbf{y}_i = F(x^s_{i-1} - x^s_i)/2 = F(v_{c_i} + v'_{c_i})/2 = 2F(v_{c_i})/2.$$

Each entry in $F(\mathbf{x}^s_i)$ is even, so each state $\mathbf{y}_i$ is has nonnegative integer entries. □

Gaucher data. Let us return to the Gaucher disease data (Example 3). Fitting HWP and ignoring the structural zero, one gets a fitted table

3.24						
4.32	1.44					
0.72	0.48	0.04				
0.36	0.24	0.04	0.01			
0.36	0.24	0.04	0.02	0.01		
0.36	0.24	0.04	0.02	0.02	0.01	
5.40	3.60	0.60	0.30	0.30	0.30	2.25

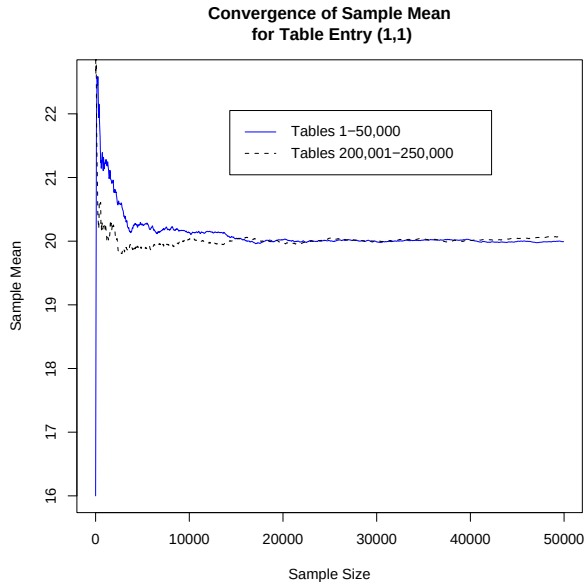

Figure 2: Markov chain results for RB1 data

and a χ^2 statistic of 19.47. The goodness of fit test on $28 - 1 - 6 = 21$ degrees of freedom gives a p value on 0.55.

The Markov chain uses 180 different moves from each position, computed with CoCoA (see Capani, Niesi, and Robbiano(2002)) as a minimal generating set for the toric ideal I given by

$$I := \{X^n - X^m : \mathbf{t}_k \cdot n = \mathbf{t}_k \cdot m, k = 1, \ldots, m, \mathbf{n}_{4,4} = \mathbf{m}_{4,4}, \mathbf{n}_{i,j} = \mathbf{m}_{i,j} \quad \forall j < i\}. \tag{11}$$

We are using the standard notation $X^n := \prod_{i,j} x_{i,j}^{n_{i,j}}$ for indeterminates $x_{1,1}, \ldots, x_{m,m}$ and $\mathbf{t}_k \cdot n$ is the number of alleles of type k for lower triangular n.

The exact p value for this table $f\ (\pi_{I_0}(\{g : \pi_{I_0}(g) \leq \pi_{I_0}\})$ was computed using a Monte Carlo Markov chain run where 250,000 tables where taken at intervals of 100. The resulting p value was 0.0413.

The convergence of the Markov chain can be seen in the following plot of the estimate of the p value versus number of iterations. For the graph, the 1000 samples were taken at intervals of 25,000 from the complete run of 25 million tables.

RB1 Data Now consider the RB1 data of Example 2. Suppose that we fix entry (4,1) to be zero. The direct sampling method of the previous section is no longer applicable. The Markov chain used 179 moves, computed via CoCoA as a generating set for the toric ideal. The exact p value is 0.000072 again computed with 250,000 tables taken at intervals of 100 from the Markov chain. The convergence is illustrated in Figure 2, which superimposes two time series of sample means of the entry (1,1) table values, using both the first 50,000 tables and the last 50,000 tables from the 250,000 tables selected at intervals of 100 from the 25 million total run length.

5 Sequential Importance Sampling

Sequential importance sampling (SIS) is another useful approach for estimating the exact p value as well as the total number of tables with a fixed set of structural zeros I_0. As in the previous section, let $\Gamma(\mathbf{F}, I_0)$ be the set of tables with $\mathbf{F} = (f_1, \ldots, f_m)$ denoting the vector of allele numbers and structural zeros at the entries in I_0.

Let π_{Uni} be the uniform distribution over $\Gamma(\mathbf{F}, I_0)$, so that for all tables $T \in \Gamma(\mathbf{F}, I_0)$,

$$\pi_{Uni}(T) = \frac{1}{|\Gamma(\mathbf{F}, I_0)|}. \tag{12}$$

Suppose that we cannot simulate from π_{Uni}, but that we can simulate from distribution q where $q(T) > 0$ for all $T \in \Gamma(\mathbf{F}, I_0)$. Denote the support of q by Γ_q (note that $\Gamma(\mathbf{F}, I_0) \subseteq \Gamma_q$.) Also, we have

$$1 = \sum_{T \in \Gamma(\mathbf{F}, I_0)} \frac{\pi_{Uni}(T)}{q(T)} q(T) = \frac{1}{|\Gamma(\mathbf{F}, I_0)|} \sum_{T \in \Gamma_q} \frac{1_{\Gamma(\mathbf{F}, I_0)}(T)}{q(T)} q(T), \tag{13}$$

and so

$$|\Gamma(\mathbf{F}, I_0)| = \sum_{T \in \Gamma_q} \frac{1_{\Gamma(\mathbf{F}, I_0)}(T)}{q(T)} q(T). \tag{14}$$

By drawing n independent samples $T_1, \ldots, T_n$ from q, we can estimate $|\Gamma(\mathbf{F}, I_0)|$ by

$$\widehat{|\Gamma(\mathbf{F}, I_0)|} := \frac{1}{n} \sum_{i=1}^{n} \frac{1_{\Gamma(\mathbf{F}, I_0)}(T_i)}{q(T_i)}. \tag{15}$$

Furthermore, consider another distribution π' on $\Gamma(\mathbf{F}, I_0)$. If we are interested in evaluating

$$\mu := E_{\pi'} f(T) = \sum_{T \in \Gamma(\mathbf{F}, I_0)} f(T) \pi'(T)$$

first we rewrite it as

$$\mu = \sum_{T \in \Gamma(\mathbf{F}, I_0)} f(T) \frac{\pi'(T)}{q(T)} q(T) = \sum_{T \in \Gamma_q} f(T) \frac{\pi'(T)}{q(T)} q(T)$$

and so we can use

$$\tilde{\mu} = \frac{1}{n} \sum_{i=1}^{n} f(T_i) \frac{\pi'(T_i)}{q(T_i)} \tag{16}$$

or

$$\hat{\mu} = \frac{\sum_{i=1}^{n} f(T_i) \frac{\pi'(T_i)}{q(T_i)}}{\sum_{i=1}^{n} \frac{\pi'(T_i)}{q(T_i)}} \tag{17}$$

to estimate μ, where $T_1, \ldots, T_n$ are i.i.d. samples from q. For example, if we let

$$f(T) = 1_{\{\pi(T) \leq \pi(f)\}} \tag{18}$$

then these estimates for μ give means for estimating the exact p value of Guo and Thompson.

In (15), (16) and (17), we only needed to focus on samples T_i that lie in $\Gamma(\mathbf{F}, I_0)$. When we only know the importance weights $\pi'(T)/q(T)$ up to a multiplicative constant (which is the case when $\pi' = \pi$), we may use (17) instead of the unbiased estimate (16).

To evaluate the efficiency of a given importance sampling algorithm, we can look at the number of i.i.d. samples that are needed to give the same standard error for $\hat{\mu}$ as n importance samples. A rough approximation for this number is the effective sample size, defined as:

$$\text{ESS} := \frac{n}{1 + cv^2} \tag{19}$$

where cv is the coefficient of variation defined as

$$cv^2 := \frac{\text{Var}_q\left(\frac{\pi'(T_i)}{q(T_i)}\right)}{E_q^2\left(\frac{\pi'(T_i)}{q(T_i)}\right)} \tag{20}$$

Intuitively, the effective sample size measures how many i.i.d. samples the n weighted importance samples are equivalent to. Accurate estimation requires a low cv^2 so that q is sufficiently close to π'.

Because we have i.i.d. samples, estimating the standard error of an estimate $\hat{\mu}$ obtained from importance sampling is strightforward. One way is just to obtain many independent estimates and calculate the standard error of those estimates.

A central problem in implementing an importance sampling approach is the construction of a good trial distribution q. Since the target space $\Gamma(\mathbf{F}, I_0)$ is rather complex, it is not immediately clear how to design q. We found that a computationally efficient method is to sample a table sequentially, column by column and row by row, in such a way as to guarantee that every table in $\Gamma(\mathbf{F}, I_0)$ has a positive probability of being produced. More precisely, we first sample the first column of the table conditional on its marginal sum f_1 and structural zeros. This is similar to what we did in Section 3. Conditional on the realization of the first column, we sample the second column in a similar fashion and then move forward recursively until all the columns are sampled. We throw the table away if any constraint is violated during the process. Denoting the configurations of the columns of T by $t_1, \ldots, t_m$, we can write $q(T)$ as:

$$q((t_1, \ldots, t_m)) = q(t_1)q(t_2|t_1)\cdots q(t_m|t_{m-1}, \ldots, t_1). \tag{21}$$

For the general framework of sequential importance sampling and its applications, see Liu & Chen(1998) and Chen(2001).

5·1 *Triangular Tables*

For a triangular table, given $f_i = 2f_{ii} + \sum_{j>i} f_{ji} + \sum_{i>j} f_{ij}$, and some structural zeros, it is not easy to guarantee that every table we generated by the sequential importance sampling algorithm is a valid table. So instead we will generate tables from a larger set Γ_q, that is, $\Gamma(\mathbf{F}, I_0) \subseteq \Gamma_q$. A table has zero weight if it is not in $\Gamma(\mathbf{F}, I_0)$. We hope to design a sampling strategy so that Γ_q is close to $\Gamma(\mathbf{F}, I_0)$ while maintaining ease of implementation.

We sample column by column, and in each column we sample cell by cell. Suppose the element at the ith row and the j column is f_{ij}. Let s_{ij} be the indicator function of whether cell (i, j) is a structural zero, that is $s_{ij} = 0$ if $(i, j) \in I_0$, and $s_{ij} = 1$ otherwise.

We start in the first column. If cell $(1, 1)$ is in I_0, we set $f_{11} = 0$. Otherwise, we can bound f_{11} by

$$\max\left(0, \frac{f_1 - \sum_{i=2}^{m} s_{i1} f_i}{2}\right) \leq f_{11} \leq \frac{f_1}{2}. \tag{22}$$

So we pick a number f'_{11} between these two bounds randomly (later we describe the distributions we can use) and set $f_{11} = f'_{11}$.

Now suppose that we are at the stage where we have chosen f_{i1} for all $i \leq k-1$. Then if cell $(k,1) \in I_0$, we make $f_{k1} = 0$, otherwise, f_{k1} must satisfy the bounds:

$$\max\left(0, f_1 - f'_{11} - \sum_{i=1}^{k-1} f'_{i1} - \sum_{i=k+1}^{m} s_{i1} f_i\right) \leq f_{k1} \leq \min\left(f_k, f_1 - f'_{11} - \sum_{i=1}^{k-1} f'_{k1}\right). \tag{23}$$

Again we choose a number between these two bounds randomly and let it be f_{k1}. Once we have reached $k = m-1$, the remaining cell $f_{(m-1)1}$ is completely determined by the equation for f_1.

Upon completion of the first column, we update the f_i for i from 2 up to m by substracting off the first column, ignore it, and begin sampling the second and subsequent columns in the same fashion.

The strategy does not guarantee that the table generated will always be in $\Gamma(\mathbf{F}, I_0)$. We stop if any constraint is violated during the sampling process, discard the partial table and begin sampling new table from scratch. In the above process, after we figure out the upper and lower bounds for a cell, there are many ways to choose a number at random in the range.

Here we discuss two possible distributions to use. The first is just to choose a number uniformly at random from the range, we call this the *uniform sampling method*. This method works well when the goal is to approximate the total number of tables with given constraints, since here the goal is to sample uniformly from the set of tables.

On the other hand, when our goal is to estimate the exact p value, we wish to use a hypergeometric sampling scheme similar to what we used in Section 3. In Section 3 we could actually generate exactly from the correct distribution. We cannot do so here because of the presence of structual zeros. However, this method offers a means for getting "close" to the desired distribution.

Let $N = \left(\sum_{i=1}^{m} f_i\right)/2$ be the total population, so there are $2N$ alleles. We follow the first method in Section 3, modifying it as necessary to deal with structural zeros. The upper bound will automatically be satisfied if we use this algorithm, but the lower bound needs to be incorporated into the algorithm so that we will not generate too many invalid tables.

Let ℓ_{ij} denote the lower bound for cell (i,j). Again $HG(s,r,t)$ is the hypergeometric distribution where we drop t balls into s slots, r of which are colored red, and count the number of balls in red slots. With a lower bound, we are trying to guarantee that the number drawn is at least ℓ (where $\ell \leq r$). When $\ell \leq t + r - s$, this condition is automatically satisfied. When $\ell > t + r - s$, we can sample a number x from $HG(s - \ell, r - \ell, t - \ell)$ and then assign $x + \ell$ to be the number of balls in the red slots. Pseudocode for the complete algorithm follows.

Hypergeometric SIS

Input: $N, f_1, \ldots, f_m, I_0$

1. **Set** $S \leftarrow N$
2. **For** j from 1 to $m-1$
3. **If** cell $(j,j) \in I_0$ then **let** $f_{jj} \leftarrow 0$, **let** $S \leftarrow S - f_j$, **Goto** step 10
4. **If** $f_j - S \geq \ell_{jj}$ then **let** $\ell_{jj} \leftarrow 0$.
5. **Let** $f_j \leftarrow f_j - 2\ell_{jj}$, **let** $S \leftarrow S - \ell_{jj}$
6. **Choose** $a_j \leftarrow HG(2S, S, f_j)$
7. **Choose** $f_{jj} \leftarrow HG(S, f_j - a_j, a_j)$
8. **Let** $f_j \leftarrow f_j - 2f_{jj}$, **let** $S \leftarrow S - f_j - f_{jj}$, **let** $f_{jj} \leftarrow f_{jj} + \ell_{jj}$
9. *Update importance weight*
10. **Let** $SS \leftarrow 2S$
11. **For** k from $j+1$ to m
12. **If** cell $(k,j) \in I_0$ then **let** $f_{kj} \leftarrow 0$, **Goto** step 18
13. **If** $f_k - SS \geq \ell_{kj}$ then **let** $\ell_{kj} \leftarrow 0$
14. **Let** $f_j \leftarrow f_i - \ell_{kj}$
15. **Choose** $f_{kj} \leftarrow HG(SS + f_j, f_j, f_k - \ell_{kj})$
16. **Let** $f_j \leftarrow f_j - f_{kj}$, **let** $f_{kj} \leftarrow f_{kj} + \ell_{kj}$, **let** $SS \leftarrow SS - (f_k - f_{kj})$, **let** $f_k \leftarrow f_k - f_{kj}$
17. *Update importance weight*
18. **End** (for statement in step 11)
19. **End** (for statement in step 2)

If any restriction is violated during the sampling process, discard the partial table and begin a new table.

Remark 2. When there are no structural zeros, both uniform sampling and hypergeometric sampling always generate valid tables.

Remark 3. We can change the order of $f_1, \ldots, f_m$ and sample column by column according to the new order. In uniform sampling we found from the examples that ordering $f_1, \ldots, f_m$ from largest to smallest usually gives slightly better results. In hypergeometric sampling, usually ordering $f_1, \ldots, f_m$ from smallest to largest gives slightly better results.

Gaucher Consider the Gaucher data of Example 3 once more. Cell $(4,4)$ is a structural zero. We ordered $f_1, \ldots, f_7$ from largest to smallest and applied uniform sampling. On a Pentium 4 at 2.40GHz machine it took only a few seconds to obtain 10^6 samples, which gave an estimate of 74270 with standard error 136 for the total number of tables. In general, uniform sampling is not efficient for estimating the p value, but for this particular example, uniform sampling gave an estimate of 0.0415 with standard error 0.0004. The coefficient of variation cv^2 is about 4 when the target distribution is uniform.

We also ordered $f_1, \ldots, f_7$ from smallest to largest and applied hypergeometric sampling. It took about 20 seconds to obtain 10^6 samples, and based on these samples, the exact p value can be estimated at 0.0415 with standard error 0.0002. We have $cv^2 = 0$ since we are exactly sampling from the target distribution. Note that the structural zero is automatically satisfied because $f_4 = 1$.

RB1 Now consider the RB1 data of Example 2. Suppose that cell (4,1) is a structural zero. We ordered $f_1, \ldots, f_7$ from largest to smallest and applied uniform sampling. In 40 seconds we obtained 10^6 samples out of which only about 500 were invalid. Based on these samples, we

estimate the total number of tables at 1.93×10^{21} with standard error of 0.03×10^{21}. Here cv^2 is about 150.

Ordering $f_1, \ldots, f_7$ from smallest to largest and using hypergeometric sampling, a few minutes gave 10^6 samples (all of which were valid tables) and based on these samples, we estimate 7.2×10^{-5} with standard error 1.4×10^{-5} for the exact p value. The cv^2 is 0.4.

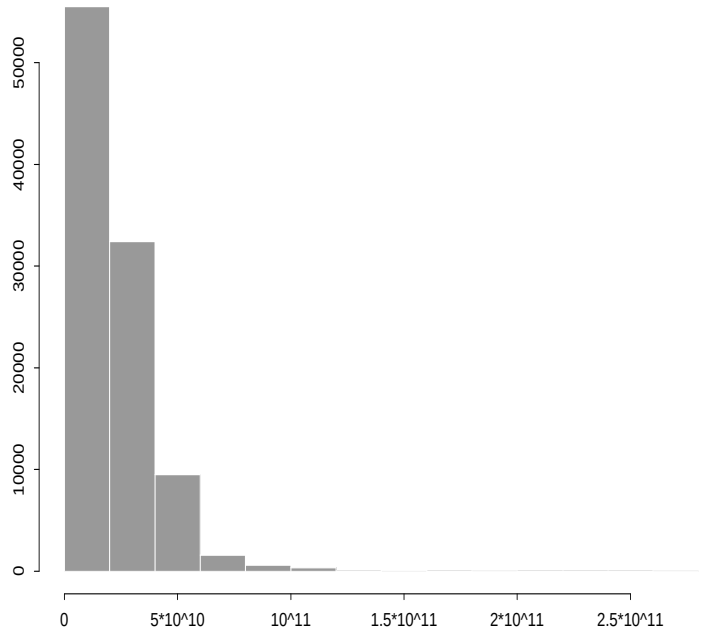

Figure 3: Histogram of 100,000 importance weights

Figure 5·1 is the histogram of 10^5 importance weights. The behavior of the weights can tell us how good the sampler is. Very few weights are very large in comparison to the rest of the samples. The maximum weight divided by the median weight is about 14, and the maximum weight divided by the minimum weight is about 53. This graph shows that the importance weights are very stable, which in turn means that our sampling distribution is close to the true distribution.

Rhesus data For the data in Example 1, there are no strucural zeros, and so the exact p value is estimated in Section 3. Therefore here we just give an estimate of the total number of tables with such constraints. We ordered $f_1, \ldots, f_9$ from largest to smallest and applied uniform sampling. It took about 8 minutes to obtain 10^6 samples, and based on these samples, we have an estimte of 1.9×10^{44} with standard error 0.1×10^{44} for the total number of tables.

6 Conclusions

Both theoretically and in practice, the new algorithm of Section 3 is the fastest method at present for generating draws exactly from Hardy-Weinberg proportions. The fact that the running time only depends on the number of alleles and not the total population is a sharp improvement over previous methods.

In Section 4 we tested several chains for this problem based on Grobner basis ideas. While we lack the theoretical understanding of these chains that we have for the direct method, they appear to be fast in practice. Moreover, these chains can be designed for problems where the state space is restricted (such as with structual zeros).

Finally, we considered sequential importance sampling methods for this problem. Again the results lack the theoretical imprint of the direct method, but appear quite fast in practice. Also, they appear to be better suited to handling the number of possible states under extra constraints.

References

Beutler, E., Gelbart, T., Demina, A., Zimran, A. & le Coutre, P. (1995). Five new Gaucher Disease Mutations. *Blood Cells, Molecules, and Diseases* **21**, 2–24.

Capani, A., Niesi, G. & Robbiano, L. (2002). *CoCoA: A system for doing computations in commutative algebra.* ftp://cocoa.dima.unige.it.

Cavalli-Sforza, L. L. & Bodmer, W. (1971). *The Genetics of Human Populations.* W. H. Freeman.

Chen, Y. (2001). *Sequential importance sampling with resampling: theory and applications.* Ph.D. thesis, Stanford University.

Diaconis, P. & Sturmfels, B. (1998). Algebraic algorithms for sampling from conditional distributions. *Ann. Statist.* **26**, 363–397.

Galbusera, P., Lens, L., Schenck, T., Waiyaki, E. & Matthysen, E. (2000). Genetic variability and gene flow in the globally, critically-engangered Taita thrush. *Conservation Genetics* **1**, 45–55.

Guo, S. & Thompson, E. A. (1992). Performing the exact test of Hardy-Weinberg proportion for multiple alleles. *Biometrics* **48**, 361–372.

Kachitvichyanukul, V. & Schmeiser (1985). Computer generation of hypergeometric random variates. *J. Statist. Comput. Simulation* **22**.

le Coutre, P., Demina, A., Beutler, E., Beck, M. & Petrides, P. (1997). Molecular analysis of Gaucher disease: distribution of eight mutations and the complete gene deletion in 27 patients from Germany. *Hum. Genet.* **99**, 816–821.

Li, C. (1955). *Population Genetics.*

Liu, J. & Chen, R. (1998). Sequential Monte-Carlo methods for dynamic systems. *J. of the Amer. Stat. Assoc.* **93**, 1032–1044.

Stadlober, E. (1990). The ratio of uniforms approach for generating discrete random variates. *J. Comput. Appl. Math.* **31**, 181–189.

Stadlober, E. & Zechner, H. (1999). The patchwork rejection method for sampling from unimodel distributions. *ACM Trans. on Modeling and Comp. Sim.* **9**, 59–80.

Takemura, A. & Aoki, S. (2002). Some characterizations of minimal Markov basis for sampling from discrete conditional distributions. Tech. Rep. 02-04.