Estimating Haplotype Frequencies from Genotypes of Pooled DNA

Bonnie Kirkpatrick

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2007-180
http://www.eecs.berkeley.edu/Pubs/TechRpts/2007/EECS-2007-180.html

December 31, 2007

Estimating Haplotype Frequencies from Genotypes of Pooled DNA

by

Bonnie B. Kirkpatrick

B.S. (Montana State University) 2004

A report submitted in partial satisfaction
of the requirements for the degree of

Masters of Science
in

Electrical Engineering and Computer Sciences

in the

GRADUATE DIVISION

of the

UNIVERSITY OF CALIFORNIA, BERKELEY

Committee in charge:

Professor Richard Karp, Chair
Professor Satish Rao

Spring 2007

Abstract

Estimating Haplotype Frequencies from Genotypes of Pooled DNA

by

Bonnie B. Kirkpatrick

Masters of Science in Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Richard Karp, Chair

Finding the genetic basis for complex phenotypes has the potential to greatly improve our understanding of phenotype expression, genetic disease, and drug response. Association studies are commonly used to find regions of the genome that correlate with the expression of a complex phenotype, and these association studies can be performed using haplotype frequency estimates from blocks of SNP data. In order for these studies to be accurate, there must be high quality haplotype frequency estimates.

Here we present, HAPLOPOOL, a novel and cost-effective method for estimating haplotype frequencies from pooled DNA samples. We assume that the genotyping is done on many unrelated diploid individuals which are pooled into disjoint pools of a small number of individuals (usually two or three individuals per pool). HAPLOPOOL is an implementation of two novel and complementary algorithms: one based on a model of the perfect phylogeny haplotyping problem and the other based on a least squares regression model of linear haplotype frequency constraints.

We compared HAPLOPOOL to three programs for haplotype frequency estimation from pool genotypes. For an objective standard, we also compared HAPLOPOOL to the state-of-the-art haplotype frequency estimation program for non-pool genotypes. HAPLOPOOL runs considerably faster (at least six times faster) than any of the four other programs. This means that it is feasible to estimate haplotype frequencies for the whole genome, after partitioning the genome into blocks of (5-25 SNPs). In addition, pooled DNA is a cost-effective strategy when compared to non-pooled DNA. Assuming that genotyping is more expensive than the sample collection and phenotyping procedures, our results show that pool genotyping and haplotype frequency estimation with HAPLOPOOL is more cost-effective than non-pool genotyping and frequency estimation. Because these assumptions often hold for species grown in laboratories, HAPLOPOOL would be useful for conducting cost-effective association studies both on model organisms and on organisms about which little is known.

Professor Richard Karp
Thesis Committee Chair

Contents

List of Figures

List of Tables

Acknowledgements

I would like to briefly acknowledge the people who aided my work or participated in this project. I want to thank Prof. Richard Karp both for working with me on this project and for being a gracious and knowledgeable mentor. Dr. Eran Halperin, at the International Computer Science Institute, provided much of the impetus for this project and patiently explained a field of research that was new to me. Thanks also go to Carlos Santos who tested many of the early ideas and helped with running the simulations. For providing the software to simulate new haplotypes using recombination, I want to thank Dr. Gad Kimmel, International Computer Science Institute. For critical reading of this manuscript, I thank Sriram Sankararaman. Finally, I am grateful for the support of the DOE Computational Science Graduate Fellowship (under grant number DE-FG02-97ER25308) and to all my fellow CSGF scholars.

Chapter 1

Introduction

Quantifying genetic diversity is often the first step to discovering the genetic mechanisms of complex hereditary diseases. A common research paradigm is to group unrelated individuals into two categories: cases and controls. The cases have the disease, while the controls do not. After estimating the frequencies of each genetic variant present in the cases and separately estimating similar frequencies in the controls, a test of statistical association can identify particular variants that occur more frequently in one group than the other. By combining the results of an association test with evidence from molecular biology, one can establish a cause and effect relationship between genetic variants and the disease. This same paradigm applies to any complex phenotype, not only diseases.

In case-control studies, the principle problem is establishing significant associations when there are many genetic variants. An association study often involves the following steps: 1) design, 2) sample collection and phenotyping, 3) data collection, 4) haplotype estimation, 5) association, and 6) power analysis. The first step, study design, is discussed only in passing, as it is highly technical and study specific. Sample collection and phenotyping are the processes of collecting samples (often blood or tissue samples) from individuals and of determining what phenotype the individual displays. Data collection is the laboratory procedure (usually genotyping) that extracts information about which genetic variants are present in the sample DNA. This is discussed in Section 1.1.3. Step (4) is the process of inferring which genetic variants were inherited from the same parent. This step, performed by HAPLOPOOL, is introduced in Section 1.2 and is the focus of the methods in Chapter 2. This thesis report evaluates the accuracy of estimating haplotype frequencies, using different methods, rather the outcome of an association study, because performing an association introduces additional opportunities for error or uncertainty. As a result, this thesis report discusses the accuracy of associations briefly and focuses on the accuracy of haplotype frequency estimation.

The power of a study is determined by the number of individuals and the number of genetic variants represented in the samples. A well designed experiment, with good power, may require thousands of individuals [7]. We assume that the cost of data collection (genotyping) dominates the cost of sample collection and phenotyping, and that data collection largely determines the cost of a study. As we see in the discussion in Chapter 4 there are many scenarios in which this assumption would be true. When this assumption holds, the cost is proportional to the number of lab experiments that must be conducted to measure genetic diversity. In the traditional case-control study, each lab experiment measures a single individual (non-pool techniques). Since each lab experiment is costly (perhaps \$0.002 to \$0.006 per SNP * 100,000 SNPs = \$200 to \$600 per person), a well designed case-control study with several thousand people may cost millions of dollars. For this reason, it is important that frequency estimation methods are cost-efficient while also obtaining good accuracy.

Lab techniques exist which are more cost-effective, since they measure the genetic diversity of several people with one experiment (pool techniques). Despite the possibility of decreasing the cost of data collection, these methods have not been employed for many case-control studies. This may be due to inaccurate or computationally inefficient programs for quantifying genetic diversity from pooled samples. This thesis report addresses this by describing two novel methods for efficiently estimating the genetic diversity in a group of unrelated individuals (Chapter 2). Since these two methods complement each other, they were combined into a single program HaploPool, which executes the two methods either separately or in concert.

After developing our algorithms, we examine the trade-offs, in terms of accuracy, of pool versus non-pool data collection methods (Chapter 3). We demonstrated that one can obtain equal accuracy for less cost by using pooled samples. The running time and accuracy of HaploPool was compared to four state-of-the-art programs for estimating haplotype frequencies from data of pooled samples. Each program was executed on input data simulated to approximate coding and non-coding genetic variation. While obtaining equal or superior accuracy, HaploPool was consistently more efficient and more capable of handling large input than the other programs.

1.1 Background

The genome, in the form deoxyribonucleic acid (DNA), encodes most of the information for the development and function of living organisms. Over the last decade, large quantities of human genomic data became available. These data specifically identify genetic variation between people. In order to understand genetic variation, we first need to consider the biological medium, DNA molecules, and how they encode variation. We also need to understand inheritance, the origin of similarity between people, and its relationship to DNA. Both variations and similarities appear in most modern data sets, because the locations of the variants are sampled with sufficient density along the DNA sequence. As a result, these biological observations provide the intuition behind the algorithms we develop in later chapters.

In order to formalize the problem that HaploPool addresses in Section 1.2, we must also discuss the laboratory methods used to collect data from a sample of DNA. We will consider two methods for processing genetic samples: non-pool and pool genotyping. Non-pool genotyping is applied to a DNA sample from a single individual. Pool genotyping is applied to a mixture of DNA from several individuals. While pool genotyping yields less information per sample than data obtained from non-pool genotyping, pool techniques are less expensive per individual than non-pool genotyping.

1.1.1 Genetic Variation

DNA and Chromosomes. A strand of DNA is a sequence of repeating units, named *nucleotides*, together with the phosphate and sugar groups which bond covalently to provide the connecting backbone structure between the nucleotides. The four possible nucleotides are represented with the letters: A, C, G, and T. Each nucleotide can form a hydrogen bond with the complementing nucleotide on another DNA strand: usually A bonds with T, and G with C. DNA is energetically stable when found as a double helix containing two hydrogen-bonded and complementary strands of DNA. One long DNA double-helix is coiled, with the aid of histone proteins, into a structure named a *chromatid*. A *chromosome* is an X-shaped structure formed by joining two identical chromatids. Each chromosome contains a 4-fold redundancy of the genetic information, since there are two identical chromatids each containing a double helix which is formed from a DNA strand and its complement.

Humans have 23 types of chromosomes, numbered 1-22 and the 23rd being the sex chromosome. Each type of chromosome encodes a different set of *genes*, with each gene being the blueprint for a functional unit (or protein). Both the unique physical features of a chromosome and the unique set of genes it contains identify a chromosome as a specific one of the 23 chromosome types. The 23rd chromosome is the sex chromosome, and it comes in two varieties, either an X or a Y. For the remainder of this discussion, we will leave aside the complexities of the sex chromosomes and will restrict our discussion to the 22 autosomal chromosomes. Most human cells are diploid, meaning that they contain two 'copies' of each chromosome type. Thus, a diploid human cell contains 46 chromosomes. It is important to note that the two copies of each chromosome, although identical in type, are *not* identical in genetic content. To emphasize this difference, we refer to these copies as *homologous chromosomes*. An analogy from computer science that may be helpful is that of classes and instantiated objects. One can think of the 22 autosomal chromosomes as classes which each contain different variable declarations, or different sequences of nucleotides. Having diploid human genetic material is similar to having two instantiations of each of the chromosome objects, and the exact nucleotide content is slightly variable across instantiations of the same chromosome.

Single-Nucleotide Polymorphisms. A location in the nucleotide sequence where variation occurs is named a *polymorphic site* or *locus.* The variants appearing in the genome at a polymorphic location are referred to as *alleles*. On average, 99.9% of sites in the human genome are identical in all humans [28]. Much genetic variation occurs at single-nucleotide locations, and these sites are known as *single-nucleotide polymorphisms (SNPs).* Thus, in the case of SNPs, the alleles are nucleotides. Continuing with the object analogy introduced above, we can think of the variable content of a single chromosome class as a long array of nucleotide elements. Notice that the array only contains the information on a single piece of DNA from each chromosome, since there is no need to represent the 4-fold redundancies. Each gene is then described as a tuple of indices which give the the start and end positions of the gene in the nucleotide array. For the purposes of the analogy, we can think of a SNP site as a single polymorphic position in the nucleotide array. In the interest of accuracy, we note that although the term 'position' is useful for conceptual explanations, it is not synonymous with 'site' or 'locus' since the latter two terms implicitly account for variations in sequence length across individuals. The alleles of each SNP would be the set of nucleotides that can possibly appear in the sequence at the SNP locus. In practice, the alleles of a particular SNP are determined by examining many genomes for variation at that particular site. While SNPs are the genetic variants on which we focus in this thesis report, there are other types of variation that are amenable to the methods presented in Chapter 2.

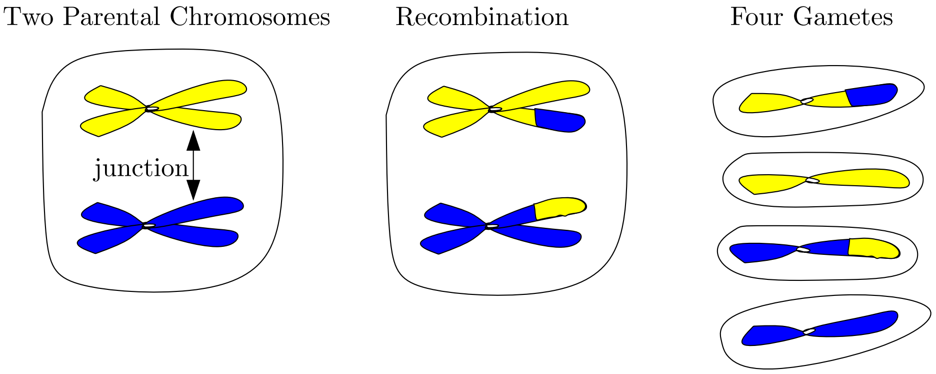


Figure 1.1. **Recombination.** This figure illustrates the two parental chromosomes and four gametes that result from the recombination event at the indicated recombination junction on the parental chromosomes. The parental chromosomes are homologous and distinguished by their colors. The chromosomes that appear together in one cell are encircled by a line. The two recombinants are a collage of the parental chromosomes. Only two of the four resulting gametes contain recombinant chromosomes, which are colored with both yellow and blue.

1.1.2 Inheritance

Since most sites in the genome are identical across all humans, there are some significant sources of genetic similarity. These sources are explained by the process of inheritance which transmits genetic material from parents to their offspring. In diploid organisms with sexual reproduction, each parent contributes half of their genetic material to each offspring. One of the offspring's homologous chromosomes comes from the father, and the other is contributed by the mother. A *haplotype* is a sequence of alleles that were inherited together from a single parent and all appear on the same chromosome in an individual. This simple picture of direct inheritance has two additional processes that produce haplotype diversity: recombination and mutation. Although these two mechanisms are most clearly observed in individuals and in families, the effects of recombination and mutation are seen among groups of 'unrelated' humans due to the evolutionary relatedness of the human species.

Recombination. A *recombination* event occurs in the parent just prior to gamete formation and results in DNA being swapped between the two homologous parental chromosomes. This forms a new combination of alleles in the haplotype being transmitted to the offspring. Thus, the genetic material contributed by a single parent is from both grandparents, because recombination yields a recombinant chromosome containing portions of both grand-parental chromosomes (see Fig. 1.1). For more molecular details about recombination, refer to Hartwell et al [13]. Between one and three recombination events occur per chromosome per generation. The average rate of recombination between contiguous nucleotides is 10^{-8}, but recombination rates vary locally by at least four orders of magnitude [21]

Recombination is usually modeled by the probability θ of a recombination event between two loci. For instance consider two SNPs, L_1 and L_2 with alleles $\{A, T\}$ and $\{G, C\}$ respectively. In a parent having haplotypes AG and TC, the *recombinant* gametes have haplotypes AC or TG while the *parental-type* gametes have AG or TC. The farther apart two loci appear on the DNA sequence, the larger the probability of observing recombinant gametes. As a result, loci that are very far apart are said to be *unlinked* with $\theta \simeq 0.5$. If two loci are contiguous, or nearly contiguous, in the genome, they are said to be *linked* because the vast majority of the gametes carry the parental-type alleles ($\theta << 0.5$). Although the frequency of recombinant gametes is usually only counted in a family study or in sperm study, there is a related concept that describes the historical effects of recombination in a largely-unrelated population. *Linkage disequilibrium (LD)* measures linkage by examining the haplotypes in the population and determining whether the alleles appearing at two loci are correlated. There are several different statistics for LD (see [1] for details).

Mutation. Rather than producing new combinations of existing alleles, a *mutation* is an event that introduces a new allele. The mutation rate per SNP is estimated as being between 10^{-8} and 10^{-9} [29]. One formulation of mutation that is useful at the population-level is the perfect phylogeny. The main assumption underlying it is the infinite-sites assumption, where each site can only mutate once over history. Of course, if there really are countably infinite sites, the genome-wide mutation rate is spread across so many sites that there is little chance of mutating the same site twice. Chapter 2 will discuss the perfect phylogeny problem in more detail.

Among unrelated people, inheritance and evolution have resulted in a small number of short haplotypes being shared by many people. If a haplotype is untouched by either recombination or mutation, it will propagate through a phylogeny and will be shared by many individuals. Recombination would seem to diversify away the effects of inheritance by re-assorting the haplotypes, but linkage results in short regions of the genome having conserved haplotypes. The process of mutation also diversifies haplotypes, but at a sufficiently low rate that local haplotypes will differ in only a few positions. These ideas inspire the two methods that comprise HaploPool.

1.1.3 Genotyping Experiments

Although haplotypes are the genetic variants that are useful for association studies, using laboratory methods to determine haplotypes from diploid cell samples is prohibitively expensive for large numbers of individuals. Instead most studies perform genotyping, an affordable analysis of genetic variation that is performed on diploid cell samples. A genotype experiment examines SNPs at particular loci in the nucleotide sequence and determines which alleles appear in the pair of homologous chromosomes.

The *genotype* of an individual reveals the (unordered) set of alleles that appear at each site. Figure 1.2 illustrates the genotypes of a portion of the DNA sequence belonging to a diploid individual. A single locus can contain two distinct alleles, in which case it is *heterozygous*, or one allele, in which case it is *homozygous*. In the example in Figure 1.2, the second locus is homozygous, while the remaining are heterozygous. Notice that the genotype is symmetric, since the set of alleles revealed by the analysis does not contain any order information. The order information is contained in the haplotypes of an individual. In Figure 1.2, the first homologue has the haplotype CTCA, while the second has ATGC. There are four haplotype pairs that satisfy the genotypes shown in Figure 1.2. In general, when there are H heterozygous sites in the observed genotype, there are 2^{H-1} haplotype pairs that could have produced the genotype.

Since genotyping is inexpensive relative to haplotyping, most studies based on haplotypes collect genotype data and infer the haplotypes from the genotypes. Due to the symmetry properties relating genotypes to haplotypes, the decision to collect genotype data introduces a non-trivial problem

1st Homologue:	C	T	C	A
2nd Homologue:	A	T	G	C
Genotype:	$\{A, C\}$	$\{T\}$	$\{C, G\}$	$\{A, C\}$

Figure 1.2. **The Genotype of a Diploid Individual.** Each line represents the DNA sequence of a homologous chromosome. Only the sequence locations that contain SNPs are labeled with the nucleotides that the individual has. The genotype of the individual is given as the last row of the figure. Notice that the genotype at each site is simply a set of nucleotides. The genotype is symmetric, since it does not reveal which homologous chromosome contains each allele. Thus, the first and last loci in this example have the same genotype.

of inferring the haplotypes. This section describes the technologies used to perform genotyping, and the section following will informally present the haplotype inference problem.

Non-Pool Genotyping. Non-pool genotype experiments are normally conducted on the diploid cells from a single individual. Genotyping is carried out on the sample using oglionucleotide-array technology, which is a matrix of short DNA or RNA probes. Each spot on the microarray has many replicates of a distinct probe. The probes in a single spot are designed to detect the presence of a specific allele at a specific SNP. For each SNP, the array contains four spots (arranged as a column) each containing a distinct probe that detects one of the four possible nucleotide at the SNP position, so each spot is indexed by a SNP and an allele.

Figure 1.3 contains a cartoon drawing of the array and the genotyping process. The DNA of an individual is collected in a sample, broken into fragments, and tagged with a florescent dye. Next, the array is coated with the sample allowing each fragment of DNA the opportunity to bind to probe sequences that complement it. Due to the short length of the probes, sample fragments differing from a probe by a single nucleotide will not bind, or hybridize, well. Thus, a sample fragment only hybridizes to the probes which detect the alleles that fragment contain. Even though each spot is separated from the other spots on the microarray, adjacent spots are in close enough physical proximity to allow some competition for the available sample DNA. These competitive binding forces are unquantified at present.

The visible intensity of the dye at each spot is correlated to the concentration of the allele. ‘Calling’ algorithms use statistical techniques to read off the frequency of each allele in the sample. For non-pooled diploid samples, the expected frequencies are 0 for one homozygote, 0.5 for heterozygotes, and 1 for the other homozygote. In most cases, these arrays detect bi-allelic SNPs, which are SNPs with two possible alleles. For convenience, rather than using letters to refer to bi-allelic SNPs, we will use the notation 0 and 1 for the alleles, where 1 is the less-frequent, or minor, allele.

Pool Genotyping. An alternative is to genotype the pooled DNA of multiple individuals. In a pool genotype experiment, the DNA of the individuals is mixed in equal concentrations in a test tube. Following this, the microarray technology and hybridization process is the same as above.

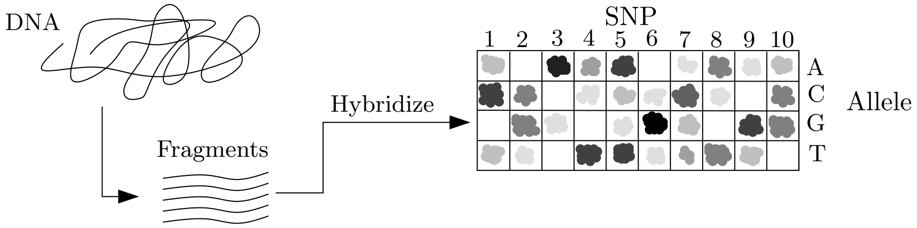
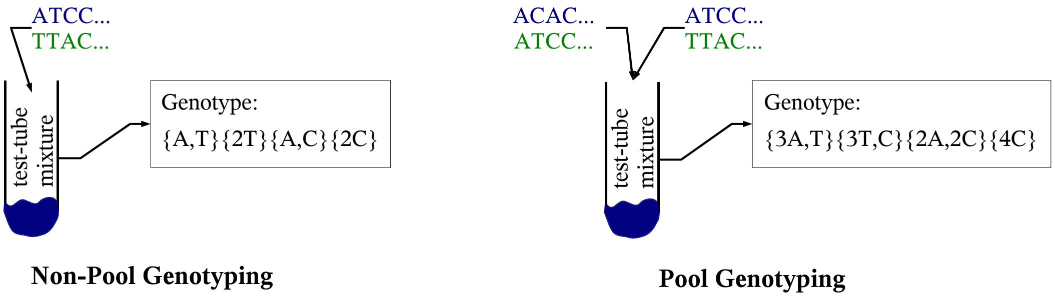


Figure 1.3. **Oglionucleotide Array.** Each spot is pictured as an entry in the matrix representation of the oglionucleotide array and is indexed by a nucleotide allele and a SNP loci. Each spot has many replicates of a probe that detects a specific allele of a SNP. The DNA sample is first broken into small fragments and labeled with a florescent dye. Following that, the array is coated with the sample. The sample fragments with high binding affinity for a probe will bind and mark that probe's spot with dye. After hybridization, a computer can detect the concentration of florescent dye in each spot.



Non-Pool Genotyping

Pool Genotyping

Figure 1.4. **Non-Pool and Pool Genotyping.** In the left panel, non-pool genotyping, is performed on a test-tube sample of DNA from a single individual having two haplotypes. The resulting genotype, shown in the box, contains a count of the number of each allele that appears at a single SNP. This non-pool genotype could have been produced by two distinct pairs of haplotypes. In the right panel, pool genotyping is illustrated by mixing DNA from two individuals. Since there are four haplotypes in the test-tube, the genotype at each SNP has four alleles, some of which are seen in several haplotypes. This pool genotype could have resulted from six (unordered) configurations of four haplotypes.

The frequency of the 1-allele is read by the same calling algorithm as used for non-pooled DNA [26]. Figure 1.4 illustrates the results obtained by genotyping non-pooled and pooled mixtures of DNA.

The main differences between pool genotyping and non-pool genotyping are expense and symmetry information. Provided that the DNA samples are available, a pool genotyping experiment is roughly comparable in expense to a single non-pool genotyping experiment. The extra expense conferred by carefully mixing the DNA in equal concentrations is relatively small [6]. Therefore, for pools of size l, the expense per individual of pool genotyping is roughly $1/l$ of the expense for non-pool genotyping. Since a pool genotype has more symmetry than an individual genotype, pool genotyping looses more of the haplotype information. This loss of information can impose a sample-size cost on a study that involves pooled DNA (see Section 3.2 and Chapter 4 for details). As a result of the increased sample-size, pool genotyping approaches are appropriate when the genotyping costs dominate sample collection and phenotyping costs. In the discussion in Chapter 4, we will discuss the scenarios under which pooling strategies are cost-effective.

1.2 Informal Problem Statements

As previously mentioned, haplotypes are genetic variants that are useful for association studies. Due to expense, most studies based on haplotypes collect genotype data and infer the haplotypes from the non-pool genotypes. Since pool genotyping is less expensive per individual than non-pool genotyping, we compare the two genotyping strategies when given the same genotyping budget for both methods. Under a given budget constraint, we investigate the accuracy of pool genotyping as compared with non-pool genotyping. In order to do this, we must first infer the haplotypes from the genotypes. Rather than just choosing one of the exponential number of haplotype resolutions to the genotype data, haplotype estimation methods take advantage of the effects of inheritance discussed above. This section informally presents the haplotype inference and haplotype frequency estimation problems.

Now we develop the notation required for the problem statements. Clearly, non-pool genotyping is simply a special case of pool genotyping where the number of diploid samples, l, in each pool is one. After the size of the pool is selected, a genotyping experiment reveals the allele frequencies in each pool. The resulting data is a set of n genotypes, each obtained from a pool of l unrelated individuals. Each pool genotype contains an allele frequency for each of the m SNPs that were genotyped. We convert these allele frequencies into allele counts and represent them in the genotype vector $a_i \in \{0, 1, ..., 2l\}^m$ where i is the pool number, $1 \leq i \leq n$. Since each SNP is assumed to be bi-allelic, only the count of a single allele need be represented. Thus, a_{ij} is the number of 1-alleles observed at SNP j in pool i. The sample-wide frequency of the minor allele (1-allele), p_j, is simply the fraction of all minor allele observations at SNP j: $p_j = \frac{1}{2nl} \sum_i a_{ij}$. Let $h_k \in \{0, 1\}^m$ be a binary string denoting a haplotype over m SNPs. If $h_{k,j} = 1$, then haplotype h_k has the minor allele at SNP j. This notation is summarized in the Index of Notation at the end of this document.

Two different haplotype estimation problems, one combinatorial and the other statistical, are considered in this thesis report. While HaploPool provides estimates for the statistical problem, the algorithms in HaploPool make use of the combinatorial problem several times.

Haplotype Inference. In Section 1.1.3 we alluded to the combinatorial problem of finding a 'good' set of haplotypes that satisfies all the genotypes. For a single pool i let a valid haplotype configuration be a set of $2l$ haplotypes whose allele counts at each SNP sum to the allele counts of a_i. In particular, let a **configuration** c_i be a vector of haplotype counts for pool i. For each haplotype $h_k \in \{0,1\}^m$, c_{ik} gives the number of times haplotype h_k appears in the configuration. Note, that c^i is quite sparse, since $c_{ik} \geq 0$ and $\sum_k c_{ik} = 2l$. In order for the configuration c_i to be **valid**, it must satisfy the following

$$a_{i,j} = \sum_k h_{kj} c_{ik}. \tag{1.1}$$

For a set of n genotypes, find a 'good' collection of valid haplotype configurations $\{c_1, ..., c_n\}$ where c_i is a valid configuration for a particular pool i. In later chapters we will formally define models that precisely characterize what a 'good' collection of configurations is.

Haplotype Frequency Estimation. The n pools represent a sample of haplotypes from a larger population. Estimate the frequencies of the haplotypes that appear in the larger population by using the evidence found in the sample of n pools. For each haplotype $h_k \in \{0,1\}^m$, there is a haplotype frequency $0 \leq f_k \leq 1$. Since these are frequencies, $\sum_k f_k = 1$. In later chapters we will define several likelihood models and, for each model, we will choose the frequencies f_k that maximize the likelihood of the model.

Naïve models and implementations for either of these two problems are likely to have running times which are exponential in the number of SNPs. As discussed in section 1.1.3, a non-pool genotype with H heterozygous sites will have 2^{H-1} valid haplotype configurations. In the case of haplotype frequencies, the state space is also exponential, even when the frequencies are restricted to values which produce valid haplotype configurations. Pool genotypes only increase the space of valid configurations, because there are more symmetry terms. For this reasons, the HAPLOPOOL algorithms focus on genotypes of small pools, where l is typically 1, 2, or 3.

1.3 Prior Work

1.3.1 Non-Pool Genotype Methods

Much work has been done on the haplotype frequency estimation problem for non-pool genotypes. A program named PHASE [27] is the state-of-the-art program for estimating haplotype frequencies from genotypes of individuals. This is a Bayesian approach that utilizes a pseudo-Gibbs sampler to sample from an approximate posterior distribution of the haplotype configurations given the genotypes. PHASE has superior accuracy than other methods and is also quite fast [19].

Statistical algorithms, like PHASE, provide one approach to non-pool haplotype frequency estimation, while other approaches are more combinatorial in nature. One such approach is the perfect phylogeny haplotyping (PPH) formulation first suggested by Gusfield [9]. This formulation relates all the haplotypes in a population by their evolutionary relationships. The PPH makes two assumptions. First, there is no recombination between the haplotypes, so the perfect phylogeny of haplotypes is a tree. Second, each new mutation occurs at a unique site, and each haplotype containing a certain mutant allele must be descended from the haplotype-ancestor where that mutation originated. More details on the PPH formulation are provided in Section 2.1.1 Since its introduction, the perfect phylogeny haplotyping approach has seen many algorithmic innovations that improve the running time required to compute a perfect phylogeny from diploid genotypes [2, 10, 8].

A successful combinatorial haplotype inference method, HAP, was inspired by both the perfect phylogeny haplotyping formulation and greedy algorithmic techniques [10, 12]. This work demonstrated the surprising accuracy of greedy haplotype inference where each iteration of the algorithm selects the most parsimonious haplotype. Halperin and Karp explained the success of the greedy approach in terms of perfect phylogeny haplotyping and the minimum-entropy set cover problem [11].

1.3.2 Pool Genotype Methods

Pooled DNA methods suffer from two caveats. First, as discussed in section 1.1.3 pooling loses information about which genotypes belong to each individual in the pool. Thus, DNA pools lose haplotype frequency information that could be useful for case-control studies of haplotype frequencies. Second, pool allele-frequencies are subject to high error rates with allele-frequency measurement variance being between 0.02 and 0.04 [17]. For this reason, determining accurate allele frequencies from large DNA pools is infeasible.

These two caveats seem to create a barrier that limit the usefulness of pool genotyping. As a result, the primary use of pooled DNA is as a screening procedure for case-control studies. All the cases and all the controls are pooled separately into two large pools. Typically, haplotype frequency estimation is not performed, because too much haplotype information is lost when pooling a large number of samples. Instead of estimating haplotype frequencies, many studies detect SNPs with large deviations in allele frequencies between the case pool and the control pool. Those SNPs identified in this screen are later genotyped on non-pooled samples to validate the findings [3, 24].

When considering the trade-offs between the pool size and the statistical power for case-control studies of the variety mentioned above, Barratt et al assumed that the measurement error rates for allele frequencies equally affect large-pool and small-pool frequencies. They concluded that pools of 50 individuals were optimal. Although the case-control studies they examined used only allele frequencies of single SNPs, rather than haplotype frequencies, their results provide some insight into the experimental costs of DNA pooling [4].

Although these previous methods have carefully considered the trade-offs involved in using large DNA pools, recent work indicates that the allele-frequency measurement error of small pools is less of a concern than previously thought. Beckman et al showed that the error rate for pools of two individuals is comparable to the non-pool genotyping error rate. Using simple clustering algorithms, they were able to accurately correct pool allele-frequencies even in the presence of relatively high error rates [6]. As discussed in Section 1.1.3, a frequency *call* is the frequency predicted by a 'calling' algorithm after it reads the dye intensity of a spot on an oglionucleotide array. Let the error σ be the standard deviation across repeated frequency calls for a single allele of a single SNP. The ability of the clustering algorithms to correct the error depends on whether 2σ is larger than the difference in frequency between allowable frequency calls. For instance, pools of two individuals have five possible allele frequency values (0, 0.25, 0.5, 0.75, and 1) with a frequency difference of 0.25 between neighboring frequency calls. Thus, a calling accuracy of $\sigma < 12.5\%$ will ensure a low rate of incorrect calls.

The haplotype frequency estimation problem for pool-genotype data is less explored than for non-pool genotypes. Methods that exist often suggest creating disjoint pools with each pool containing a small number of unrelated individuals. Pe'er and Beckmann [23] used linear regression to estimate haplotype frequencies from small pools. Three other frequency estimation methods for pools also suggested using small pools: Pools2 [14], EHP [30], and LDPooled [15]. In all three programs, the expectation-maximization (EM) algorithm was suggested to infer haplotype frequencies from small DNA pools. Yang et al [30] was the first to explore the efficiency trade-offs of small pools. Their EM approach, implemented in a program named EHP, was limited in two aspects. First, EHP's running-time limits the size of the data set that can feasibly be processed. Their paper only shows results for haplotypes spanning three SNPs, making it difficult to know whether the pooling efficiency they observed extends to longer haplotypes. Second, they only compared EHP's haplotype frequency results to results obtained by EHP under a different pool size. Evidently it is hard to interpret their results since the performance of their method on genotypes may be inferior to state-of-the-art methods such as PHASE. In particular, it has not been shown that EHP, for pools of size two, is more efficient than PHASE on genotypes.

1.3.3 Our Contribution

The contributions of this thesis report are mirrored by a paper that was joint work between B. Kirkpatrick, C. Santos, R. Karp, and E. Halperin [16]. This section and the remainder of this report are based on that paper.

As far as we are aware, many of the innovations used for non-pool haplotype inference, such as perfect phylogenies have not previously been applied to pool haplotype inference. As a result, our work is the first to suggest using perfect phylogenies for pool haplotype frequency estimation and is the first to treat deviations from the perfect phylogeny model in a formal manner. In addition, we provide a thorough comparison between our method, HAPLOPOOL and the three EM-based haplotype frequency estimation methods that were mentioned above. Since all four of these methods can also process non-pool genotype data, we compare the performance of all four methods on non-pool genotypes to the performance of PHASE on the same non-pool genotypes. This allows for an objective perspective of the efficiency of pooling technologies in terms of the non-pooling technologies that are more commonly used.

Assuming that the sampling and phenotyping costs are less than the genotyping costs, estimating haplotype frequencies from pool genotypes is more cost-efficient than estimating haplotype frequencies from non-pool genotypes. We reach this conclusion from the following result which is presented in more detail in section 3.2. According to our study, genotyping n pools of size 2 is roughly equivalent in cost to the genotyping of n individuals using traditional genotyping, but it is roughly equivalent in accuracy to the genotyping of $1.45n$ individuals using traditional genotyping. In addition, HAPLOPOOL has a faster running time than all four of the other haplotype frequency estimation programs with which we compared (Section 3.2). This means that it is feasible to use HAPLOPOOL for whole-genome haplotype frequency estimates (after partitioning the genome into blocks of 5-25 SNPS). Finally, HAPLOPOOL's speed makes it feasible to consider using a permutation test to estimate the p-value of an association study based on haplotype frequencies.

Chapter 2

Methods

In this chapter we introduce the two algorithms that HAPLOPOOL employs to predict haplotype frequencies from a data set of n genotypes each taken from pools of l individuals. These methods are based on the perfect phylogeny and linear regression. Each algorithm is a stand-alone solution to the haplotype frequency estimation problem, but each algorithm has different assumptions and produces good results for data that fit those assumptions. As a result, the HAPLOPOOL implementation has both algorithms and reports the haplotype frequencies from the algorithm that best fits the data.

To describe how HAPLOPOOL is implemented, we independently introduce the perfect phylogeny model and the regression model in Sections 2.1 and 2.2 respectively. Since the HAPLOPOOL implementation uses properties of those models to decide which frequency estimates are believable, the discussion of how the models are integrated appears in Section 2.3 after the models are introduced. Portions of this chapter are taken from Kirkpatrick et al [16].

2.1 The Perfect Phylogeny

In this section, we use a phylogeny to describe the evolutionary relationships between haplotype sequences. Each haplotype in the phylogeny is either an extant or an ancestral haplotype. The 'oldest' haplotype in the phylogeny is the most recent common ancestor of all the extant haplotypes. There are many possible phylogenies that can describe the relationships between the haplotypes. We use the perfect phylogeny because it places constraints on the phylogenies that are allowed to explain the data. The perfect phylogeny does not allow recombinations, reverse mutations, where a mutation occurs and then reverts back to the ancestral form, or recurrent mutations, where the same mutation occurs in several ancestors.

We introduce a statistical model that infers a perfect phylogeny for pool-genotype data. Gusfield suggested the perfect phylogeny formulation of the haplotype inference problem [9], and he called this formulation the perfect phylogeny haplotyping (PPH) problem. We will first introduce the PPH problem and later discuss the statistical formulation of the perfect phylogeny for pool-genotype data. The statistical model formally treats deviations of the data from a perfect phylogeny. To distinguish between these two conceptual formulations, we will refer to Gusfield's formulation as the PPH problem and the statistical model as the perfect phylogeny model (PPM).

2.1.1 Perfect Phylogeny Haplotyping

As mentioned in Chapter 1, perfect phylogeny haplotyping (PPH) is a formulation of the haplotype inference problem that describes the evolutionary relationships between haplotype sequences. In a phylogeny, more recent haplotypes are generated from ancestral haplotypes by mutations. In this context a *mutation* is the spontaneous acquisition of an allele that differs from the ancestral allele. The ancestral allele is sometimes referred to as *wildtype*, while the more recent allele is *mutant*. When thinking about SNP data, the major allele, or zero-allele, usually represents the wildtype or ancestral allele, while the less frequent one-allele represents the mutant.

A perfect phylogeny on binary characters (in this case SNPs) is defined as a rooted tree having haplotypes as nodes and SNPs as edges. Every edge in the tree, labeled with a SNP, represents a $0 \rightarrow 1$ mutation at that SNP. Note that this tree is not necessarily a binary tree. Since there are many evolutionary events that could violate a tree-like structure, we will introduce the two assumptions that guarantee the phylogeny to be a tree:

1. The number of sites is infinite relative to the genome-wide mutation rate, which allows each site to mutate at most once in the phylogeny.
2. No recombination occurs between haplotypes, and thus each haplotype has a single ancestor.

From property (2), it is easy to see that the phylogeny must be a tree, since each haplotype can only have one ancestor. The definition of the perfect phylogeny tree as *rooted* assumes that there is a most recent common ancestor (MRCA) for the phylogeny. Property (1) tells us that each haplotype containing a mutation descended from the ancestral haplotype where the mutation first occurred. Let each node of the tree represent a haplotype, then the root of the tree is the MRCA of the extant haplotypes. Each edge in the tree represents a mutation that occurred between a **parent-node** and a **child-node**, and is called a **parent-edge** of the child-node. Parent and child relationships in the tree are indicated by two nodes sharing an edge, and the parent node has the shorter path to the root. This figurative use of parent and child should not be confused with genetic relationships such as Mendelian inheritance. Thus, every edge has two adjacent haplotypes which differ by a single SNP, and that SNP is the mutation that labels the edge (Figure 2.1). Property (1) tells us that there is at most one edge in the tree that corresponds to each mutation, and this edge corresponds to the one time in history when the mutation occurred.

Let T be a rooted tree representing a perfect phylogeny. Every edge represents a $0 \rightarrow 1$ mutation at that SNP. Every node, v, in the tree represents an extant haplotype or an ancestral haplotype, h_v. Let d' be the number of nodes or haplotypes in the tree. The tree structure requires that there are $d' - 1$ edges for d' nodes, and, hence, $d' - 1$ is the number of SNPs that mutate in the perfect phylogeny. As a result of property (1), $d' - 1 \leq m$ where m is the total number of SNPs. Let the nodes be numbered $1, ..., d'$ where v is the node number. Let $v = 1$ be the root node, and let the root haplotype consist of the ancestral alleles $h_1 = \vec{0}$. If there is any rooted perfect phylogeny consistent with a data set, then the alleles can be relabeled by swapping the zero- and one-labels

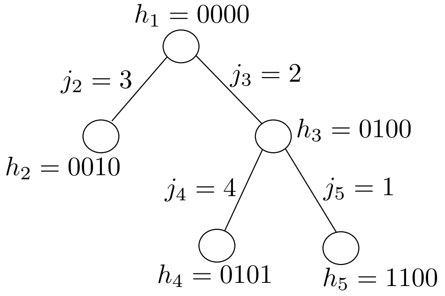


Figure 2.1. **Perfect Phylogeny Haplotyping Example.** In this tree, each node in the tree is labeled with a haplotype and each edge is labeled with the column number of the SNP that mutated along the branch. The root haplotype is the zero vector. Notice that the haplotype 1010 does not appear anywhere in the tree, because its appearance would violate the properties of a perfect phylogeny.

to yield a root of all major alleles [20]. As a result of this, we can restrict our attention to rooted phylogenies where the root consists entirely of major 0-alleles.

For convenience, we number the edges using the node-number of the child-node v which is farthest from the root. This is possible because there are $d' - 1$ edges, so the edges are numbered by v where $2 \leq v \leq d'$. Thus, for every node v in the tree, the parent-edge of v is labeled with SNP j_v. An example of a perfect phylogeny for haplotypes is given in Figure 2.1. For bi-allelic SNPs, the edge labels together with the zero-haplotype root is enough to fully specify the haplotypes that label each node. The path from the root to a node v gives the SNPs that have 1-alleles in the haplotype that labels v. For instance, the path from the root to node 5 has two edges: j_3 and j_5. Thus, the haplotype h_5 has 0-alleles at every SNP, except SNPs 2 and 1 where the haplotype has 1-alleles.

The haplotypes that are induced by the edge labels are said to be **compatible** with the perfect phylogeny tree that induced them. Notice that the haplotype 1010 does not appear anywhere in Figure 2.1, because it is not compatible with the perfect phylogeny. Creating haplotype 1010 from the phylogeny would require either a recombination between h_2 and h_5 (violating property (2)) or a recurrent mutation adjacent to h_2 (violating property (1)).

In this description, we have restricted our discussion to a perspective that is specific to the goal of describing a likelihood model for perfect phylogeny haplotyping. There are some related problems which deserve to be mentioned. There is a great deal of interest in combinatorial algorithms for building perfect phylogenies from data [2, 10, 8]. The most recent algorithms are practical for use on the whole-genome scale. In addition, the perfect phylogeny formulation given here is restricted to bi-allelic data. The phylogenetics research community is also interested in more general formulations of the perfect phylogeny, see *Phylogenetics* by Semple and Steel [25] for details.

2.1.2 Perfect Phylogeny Model

An obvious disadvantage to the combinatorial formulation of the perfect phylogeny haplotyping problem is that there is no way of dealing with errors in the data or deviations of the data from the perfect phylogeny. A statistical model can handle deviations and errors by penalizing perfect

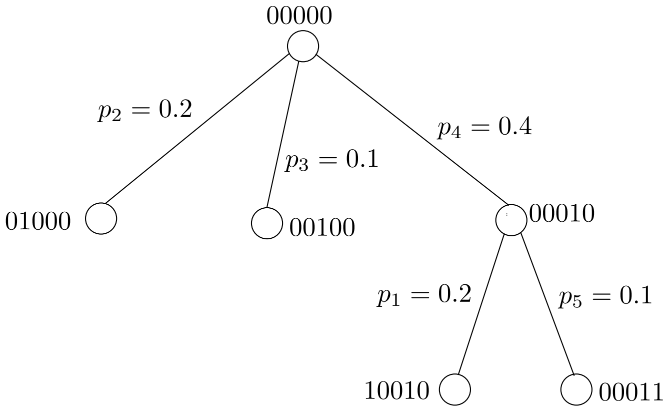


Figure 2.2. **Perfect Phylogeny Model Example.** Each node of this perfect phylogeny is labeled with a haplotype, while every edge is labeled by a SNP and a frequency. The edge corresponding to the mutation at SNP j_v is labeled, here, with the minor allele frequency p_{j_v}. We compute the haplotype frequencies from the allele frequencies. For example, the frequency of haplotype 00010 is computed as $p_4 - (p_1 + p_5)$. Once the haplotype frequencies are computed, the tree gives a model for a specific population of haplotypes.

phylogenies that are not good matches for the data. We introduce a statistical model of the perfect phylogeny that has a penalty for perfect phylogenies that require many recurrent mutations in order to generate the pool-genotype data. Since we are interested in estimating haplotype frequencies, we want to find a maximum likelihood estimate of the haplotypes $\hat{\mathcal{H}}_T = \{h_1, \ldots, h_{d'}\}$ and their frequencies $\hat{\mathcal{F}}_T = \{f_1, \ldots, f_{d'}\}$ from the perfect phylogeny tree that 'best' fits the data. The fit of a tree to the data is described by a likelihood model and the 'best' tree is the one that maximizes the likelihood. In this section, we first introduce the likelihood model and later give a procedure for finding the maximum likelihood perfect phylogeny tree.

The Likelihood of a Perfect Phylogeny Tree. For every perfect phylogeny tree T, we compute the likelihood of the data being generated by that particular tree. In order to do that, we need the tree together with its haplotype frequencies, since they fully specify the likelihood of the data. As mentioned above, the haplotypes $\mathcal{H}_T = \{h_1, \ldots, h_{d'}\}$ of a tree are induced by the nodes of the tree. We can use the tree to estimate the frequencies, $\mathcal{F}_T = \{f_1, \ldots, f_{d'}\}$, of these haplotypes in the following way. For a haplotype $h_v \in \mathcal{H}_T$, let j_v be the SNP corresponding to the parent-edge of h_v, and let $j'_1, \ldots, j'_k$ be the SNP-edges leading to the child haplotypes of h_v. Recall that we defined p_{j_v} as the sample-wide average of the minor-allele frequency in Section 1.2. Since the mutation represented by j_v occurs only once in the tree, p_{j_v} is the total frequency of all the haplotypes in the subtree T_{h_v} (see Figure 2.2). The same holds for each $p_{j'_w}$ and the subtrees rooted at the children of h_v. Therefore, we estimate the frequency of h_v as $f_v = p_{j_v} - \sum_{w=1}^{k} p_{j'_w}$. For the root h_1, we assume that $p_{j_1} = 1$, and thus the sum of the frequencies of all haplotypes in the tree is exactly 1.

The resulting compatible haplotypes $\mathcal{H}_T = \{h_1, \ldots, h_{d'}\}$ and frequencies $\mathcal{F}_T = \{f_1, \ldots, f_{d'}\}$ are now used to compute the likelihood of the tree T. First, if one of the resulting haplotype frequencies is negative, we set the likelihood to be zero. Otherwise, the likelihood of the data given the tree is simply the probability of the data given $\mathcal{H}_T$ and $\mathcal{F}_T$, which will be defined in Equation (2.1), where the data are allowed to deviate from the perfect phylogeny. In order to explicitly write the likelihood function, we define a **configuration** of haplotypes for a pool i to be a vector $c_i = (c_{i,1}, ..., c_{i,d'})$, where $c_{i,v}$ is the count for haplotype h_v in the pool such that $0 \leq c_{iv} \leq 2l$, and $\sum_v c_{iv} = 2l$. The **mutation number** of the configuration c_i with respect to pool i is $mut(c_i, i) = \sum_{j=1}^{m} |a_{i,j} - \sum_v h_{v,j} c_{iv}|$. Informally, $mut(c_i, i)$ is the number of recurrent mutations needed to explain the pool frequencies by the compatible haplotypes in the configuration.

With these definitions, we will define the likelihood function of a perfect phylogeny. Since the individuals are unrelated, we assume Hardy-Weinberg equilibrium (random mating and no selection). This means that each pool is made by drawing $2l$ haplotypes from a shared haplotype-frequency distribution, $f_1, ..., f_{d'}$. The random mating assumption also leads to the pools being independent. Thus, we define the likelihood as a product over pools:

$$L(a_1, \ldots, a_n \mid T) = \prod_{i=1}^{n} \left(\max_{\{\text{configuration } \mathrm{c_i}\}} \epsilon^{mut(c_i,i)} \cdot \prod_{v=1}^{d'} \frac{f_v^{c_{iv}}}{c_{iv}!} \right), \tag{2.1}$$

where ϵ is the probability for a mutation, which we assume is known. The mutation probability is the prior probability of observing a point-mutation in any of the perfect phylogeny haplotypes, and it provides a penalty for trees with haplotypes that deviate a great deal from the data. This probability includes both the probability of a point-mutation actually occurring in the population and the probability of there being a genotyping measurement error which leads to an incorrect allele count. In order to compute the likelihood of the tree T, we must find the most likely configuration for each pool, i.e., the configuration that maximizes the expression $\epsilon^{mut(c_i,i)} \cdot \prod_{v=1}^{d'} \frac{f_v^{c_{iv}}}{c_{iv}!}$.

Dynamic Programming Likelihood Computation. Since there is an exponential number of possible configurations, it is not feasible to try every possible configuration for every possible pool. Therefore, for a fixed tree, we use a linear-time dynamic programming algorithm to find the most likely configuration for every pool and to compute the likelihood of the pool. For every haplotype h_v in the tree T, we define $L_i(h_v, t)$ as the likelihood of pool i restricted to the SNPs in subtree T_{h_v} such that t haplotypes from pool i are assigned to the subtree, i.e., $\sum_{v \in T_{h_v}} c_{iv} = t$. Hence, the most likely configuration for pool i is the one maximizing $L_i(h_1, 2l)$. To compute this function we use bottom-up dynamic programming. For every leaf h_v, if the parent-edge adjacent to h_v corresponds to a SNP j_v, it is easy to see that

$$L_i(h_v, t) = \epsilon^{|a_{i,j_v} - t|} \cdot \frac{f_v^t}{t!}.$$

For every internal node h_v, if j_v is the SNP corresponding to the parent edge of h_v, and $h'_1, \ldots, h'_k$ are the direct children of h_v, then

$$L_i(h_v, t) = \max_{t_0, t_1, \ldots, t_k \geq 0, \sum t_w = t} \left\{ \epsilon^{|a_{i,j_v} - t|} \cdot \frac{f_v^{t_0}}{t_0!} \cdot \prod_{w=1}^{k} L_i(h'_w, t_w) \right\}$$

where t_w is the number of haplotypes assigned to subtree $T_{h'_w}$ and t_0 is the number of haplotypes assigned to haplotype h_v. It is now easy to compute $L_i(h_v, t)$ based on the children's values using a dynamic programming procedure.

Finding the Maximum Likelihood Perfect Phylogeny Tree. In order to find the most likely tree, we implicitly compare all possible trees while constructing the max likelihood tree from top to bottom. Since we are interested in perfect phylogenies, we restrict our search to trees in which the allele frequency of a SNP is larger than the allele frequency of any SNP in its subtree. For simplicity of explaining the perfect phylogeny model, we assume that the m SNPs are ordered by decreasing minor-allele frequency: $p_1 \geq p_2 \geq ... \geq p_m$. Thus, we start with the tree S_0 consisting of the single vertex h_1. We then compute a sequence $S_1, S_2, ..., S_m$, where S_j is a set of perfect sub-phylogenies over the SNPs $1, \ldots, j$. S_{j+1} is obtained from S_j as follows: in all possible ways, we augment each sub-phylogeny in S_j by adding a new leaf and attaching it to the sub-phylogeny by an edge labeled with SNP $j+1$; we compute the likelihood of each resulting sub-phylogeny, and retain the 500 with the highest likelihood. Finally, the highest-scoring phylogeny in S_m is used to obtain the estimated haplotypes $\hat{\mathcal{H}}_T = \{h_1, \ldots, h_{d'}\}$ and their frequencies $\hat{\mathcal{F}}_T = \{f_1, \ldots, f_{d'}\}$.

2.2 Linear Regression

This method is entirely distinct from the PPM method above. Now we exploit linkage disequilibrium to obtain a haplotype frequency estimate. Even though in theory 2^m haplotypes are possible, we see in practice that within a short genomic region there is a limited diversity of haplotypes. We leverage on the fact that SNPs in close proximity are highly correlated, and thus a small number of haplotypes typically cover at least 80% of the population [22].

First, we calculate a set of *candidate haplotypes* based on a variant of the greedy algorithm for haplotype inference [12]. This method iteratively chooses parsimonious candidate haplotypes. Due to LD and limited diversity, we search for a restricted set of candidate haplotypes, $\mathcal{H} = \{h_1, \ldots, h_d\}, (d << 2^m)$, which may explain our set of pools. Once such a set is found, we want to find the frequencies $\mathcal{F} = \{f_1, \ldots, f_d\}$ of the candidate haplotypes in the data. These frequencies are estimated in two steps: local estimation and global estimation. *Local estimation* proceeds by breaking the problem into fragments of many subsets of the data each of which is restricted to several SNPs. These subsets are called 'projections', and the genotypes in them, 'projected' genotypes. Then, we use an Expectation Maximization (EM) algorithm to estimate haplotype frequencies separately for each set of projected genotypes. The *global frequencies* of the candidate haplotypes are estimated over all the SNPs in a block. The global estimates are obtained from local estimates by using them as linear constraints for least squares regression. The regression coefficients are the final estimated haplotype frequencies, $\mathcal{F}$.

This method is both accurate and fast. The accuracy of this method stems from three things: the accuracy of EM for a small number of SNPs, the ability to identify common haplotypes using the greedy method, and the accurate deduction of global haplotype information from the estimates on subsets of SNPs. The speed of the method is due to the number of candidate haplotypes and to the small size of the local estimates. As a result of limited diversity, in most cases we expect to find that d is quite small (no more than 20), and therefore the least squares regression can run efficiently on this small number of candidate haplotypes. Since the EM algorithm for a small number of SNPs is efficient, the performance of our algorithm is not limited by the running time of the EM algorithm.

2.2.1 Candidate Haplotypes from the Greedy Algorithm

The greedy algorithm finds candidate haplotypes for the global estimates of the haplotype frequencies. Informally, the greedy method begins with an initial state representing a set of reliable haplotypes, and it continues by selecting the most parsimonious haplotype to add to that set. To do this, we initialize the greedy algorithm with a set of haplotypes $\mathcal{H}_I$ together with a total ordering on the elements of the initialization set. We will call this relation $\leq_I$ and we can use it to find the maximum $h \in \mathcal{H}_I$. For example, this order could be given by a frequency for each of the initializing haplotypes. Then the frequencies can be compared to each other to determine the order of the haplotypes. If $\mathcal{H}_I = \emptyset$, then this represents a lack of knowledge about the candidate haplotype set. More details about alternative ways of initializing the greedy algorithm are given in Section 2.3.

Once the greedy algorithm is initialized, it finds a haplotype that can be used to explain the maximum number of haplotypes from all the pools, and it adds that haplotype to the set $\mathcal{H}_g$. Then, it finds a haplotype that explains the largest number of the remaining haplotypes in unexplained pools. This haplotype corresponds to the most parsimonious haplotype, in terms of allele counts for all the pools. The algorithm assigns this haplotype to as many pools as possible and adds the haplotype to the set $\mathcal{H}_g$. Then it finds the next most parsimonious haplotype. This continues until each pool has every allele count satisfied by the set of $2l$ haplotypes assigned to that pool. After all iterations of the greedy algorithm are complete, we have set of candidate haplotypes $\mathcal{H} = \mathcal{H}_I \cup \mathcal{H}_g$. This set of haplotypes is the set to which we restrict ourselves in calculating haplotype frequencies in Section 2.2.

A Formal Description of the Greedy Algorithm. In order to explain how the greedy algorithm is initialized, we begin by considering all the allele counts (in every SNP and every pool) as unexplained by inferred haplotypes. Let *counts* be a vector where $counts_{i,j}$ is the count of the unexplained major alleles for locus j in pool i. When the algorithm begins, $counts_i := a_i$. Let the number of remaining unexplained haplotypes for pool i be up_i. In fact, up_i can be viewed as an upper bound on the maximum number of allele counts that a single haplotype can explain, and thus it is initially set to $up_i := 2l$. If $\mathcal{H}_I = \emptyset$, then these initial values for *counts* and *up* do not change before starting the body of the algorithm.

In order to describe how the *counts* and *up* vectors are initialized from the set $\mathcal{H}_I$, we need to introduce some notation. For a haplotype $h \in \{0,1\}^m$, the **multiplicity** of h in pool i is $M(h,i) = \max_t\{\forall j, t + counts_{i,j} - up_i \leq t \cdot h_j \leq counts_{i,j}\}$. Informally, the haplotype h fits $M(h,i)$ times in pool i, since for each j, if $h_j = 1$ then $M(h,i) \leq count_{i,j}$, and if $h_j = 0$ then $M(h,i) \leq up_i - counts_{i,j}$.

At each iteration of the initialization procedure, the algorithm selects the next initializing haplotype from set $h \in \mathcal{H}_I$ such that $h \geq_I h_i$, $\forall\{h_i \in \mathcal{H}_I\}$. Then for every pool i, we use this haplotype h to explain $M(h,i)$ of the haplotypes in the pool. So, we update $counts_i = counts_i - M(h^*,i)h^*$ and $up_i = up_i - M(h^*,i)$. Finally, before beginning the next iteration, we remove h from the initial set of haplotypes $\mathcal{H}_I = \mathcal{H}_I \setminus h$.

Now that *counts* and *up* have been initialized, we consider the iterative body of the greedy algorithm. These iterations will continue until all allele counts are explained by a set of inferred haplotypes. The algorithm first finds the haplotype h^* that maximizes the sum of multiplicities. That is, $h^* = argmax \sum_i M(h,i)$. Then, for pool i, we assign $M(h^*,i)$ of its haplotypes the value h^*, and we update $counts_i = counts_i - M(h^*,i)h^*$, $up_i = up_i - M(h^*,i)$. It is easy to see that an invariant of the algorithm is that up_i and $counts_i$ have non-negative coordinates in each iteration, and that for each $j, up_i \geq counts_{i,j}$ always holds. Furthermore, in each iteration $\sum_i up_i$ decreases, because a haplotype was assigned to at least one pool. The algorithm terminates when $up_i = 0$ for every i.

Within each iteration of the greedy algorithm, finding the most parsimonious haplotype h^* is not an easy task. To quickly identify h^*, we use a branch and bound procedure. We view the haplotypes as the leaves of a balanced binary tree of depth m, where the path to a haplotype determines its value (a left child corresponds to 0 and a right child corresponds to 1). We traverse this tree in a depth-first-search fashion, pruning subtrees based on bounds supplied by two sources. The first bound we consider is given by the sum of multiplicities of the best haplotype found so far. The second bound τ halts the search when an examination of the partial haplotypes corresponding to that branch implies that all the haplotypes h in that branch have $\sum_i M(h,i) < \tau$ for some parameter τ. We start by assigning $\tau = 2nl$ and reduce it gradually until one branch survives (clearly, if $\tau > \sum_i M(h^*, i)$ then none of the branches survive). In practice, the more individuals in the pool, the fewer values of τ that need to be tried before h^* is found, as fewer constraints are imposed on the haplotypes for larger pools.

When larger pools are considered, it is often the case that a given iteration has two optimal haplotypes h^* and g^* such that $\sum_i M(h^*, i) = \sum_i M(g^*, i)$. In order to overcome this, we adjust the algorithm to consider ties by introducing backtracking. Let the sum of multiplicities for haplotypes h^* and g^* be s. The particular loop variables which resulted in finding s will be called $counts^s$ and up^s. First the algorithm finds all explanations of the remaining allele counts $counts_i^s - M(h^*, i)h^*$ and unassigned haplotypes $up_i^s - M(h^*, i)$. Then, the algorithm backtracks to begin again at $counts^s$ and up^s and assigns s of the unexplained haplotypes to g^*.

We note that the greedy algorithm is easily extended to allow for missing data. This is done by augmenting the definition of multiplicity. Assume that a missing allele count for SNP j in pool i is indicated by $a_{i,j} =$ '?'. Then we modify the multiplicity computation to ignore the missing constraints by setting $M(h,i) = \max_t\{\forall j \text{ with } h_j \neq \text{'?'}, t + counts_{i,j} - up_i \leq t \cdot h_j \leq counts_{i,j}\}$. The algorithm then continues as before.

The output of the greedy algorithm is therefore a set of haplotypes, and an assignment of these haplotypes to each of the pools, so that the allele counts in each pool can be explained by the set of haplotypes assigned to that pool. We observe that in practice the greedy algorithm almost always identifies all the common haplotypes, but it sometimes misses rarer haplotypes. However, many of these rare haplotypes are actually a single mutation away from one of the common haplotypes. Therefore, we enhance the greedy algorithm by extending the set of candidate haplotypes to include haplotypes that have hamming-distance one from the most common greedy haplotype. Formally, if h^f is the most frequently assigned haplotype by the greedy algorithm, we extend the set of haplotypes to include the set $\{h \text{ such that } d_H(h, h^f) = 1\}$ where $d_H(h, h^f)$ is the hamming distance between the two haplotypes.

2.2.2 Local Estimates from the EM Algorithm

Once the set of candidate haplotypes $\mathcal{H} = \{h_1, \ldots, h_d\}$ is determined, we can now consider how to compute the frequencies of those candidate haplotypes. When considering pools of size l, the number of possible valid configurations is roughly d^{2l}, where d is the number of candidate haplotypes. Thus, to ensure a reasonable running time for the EM algorithm on pools of more than two individuals, it is essential that the number of haplotypes be relatively small.

To ensure a small number of haplotypes, we focus on the information contained in subsets of the available SNPs. For a subset of SNPs $P \subseteq \{1, \ldots, m\}$, we project the haplotypes in $\mathcal{H}$ onto the columns of P. This results in haplotypes on subsets of the SNPs, or **projected haplotypes**, $\mathcal{H}^P = \{h_1^P, \ldots, h_d^P\}$, where $h_v^P \in \{0,1\}^{|P|}$ is the binary string h_v projected to the set of columns P with $1 \leq v \leq d$. Notice that although we are continuing to use the subscript v, now v indicates some candidate haplotype, rather than a node in a perfect phylogeny tree. Clearly, different haplotypes of $\mathcal{H}$ may have the same projection, and thus, $\mathcal{H}^P$ may contain fewer than d haplotypes. In a similar manner, we obtain projections of the pool allele-count vectors $a_i^P \in \{0,1\}^{|P|}$ for every pool i. The projected haplotype set $\mathcal{H}^P$ together with the projected allele count vectors a_i^P are then used as an input for the EM algorithm. Thus, for each such subset of SNPs, the EM algorithm provides an estimation of the frequencies of the haplotypes of $\mathcal{H}^P$.

For simplicity of presentation, we will describe the EM algorithm applied to the entire set of m SNPs. In order to compute the estimate on projected data, simply take $m := |P|$ and give the EM algorithm the projected genotypes. Once again, we use the vector $c_i = (c_{i1}, ..., c_{id})$ to represent a configuration for a pool i, where c_{iv} is the count for haplotype h_v with $0 \leq c_{iv} \leq 2l$. A configuration is said to be **valid** in pool i if and only if $\sum_v c_{iv} = 2l$ and for every SNP $1 \leq j \leq m$, $a_{i,j} = \sum_j h_{v,j} c_{ij}$. Similar to the likelihood for the perfect phylogeny, assume that random mating and Hardy-Weinberg equilibrium hold. Since there is a shared haplotype-frequency distribution, $f_1, ..., f_d$, the likelihood of a single pool, given a configuration, is a product of the frequencies of each haplotype in the configuration. Since the configurations are disjoint, the likelihood of a pool is taken as a sum over the valid configurations. Since the pools are assumed to be independent, the likelihood function is

$$L(a_1, ..., a_n | f_1, ..., f_d) = \prod_{i=1}^{n} \sum_{\{c_i \text{ valid in pool } i\}} \frac{2l!}{c_{i1}! c_{i2}! \cdots c_{id}!} f_1^{c_{i1}} f_2^{c_{i2}} ... f_d^{c_{id}}.$$

The Expectation Maximization algorithm finds a set of frequencies $\hat{\mathcal{F}} = \{\hat{f}_1, \ldots, \hat{f}_d\}$ that maximizes the likelihood. In each iteration we have an estimate $\hat{f}_1, \ldots, \hat{f}_d$. Then, the EM algorithm alternates between computing an expectation step (E-step) and a maximization step (M-step). The expectation step computes the expected configuration count for each haplotype in pool i: $\beta_v^i = (1/Z) \sum_{valid\ c_i} c_{i,v} Pr[c_{i,v} | \mathcal{F}, a_i]$ where the partition function is $Z = \sum_v \sum_{valid\ c_i} c_{i,v} Pr[c_{i,v} | \mathcal{F}, a_i]$. The full E-step computation is written as

$$\beta_v^i \propto \sum_{\{c_i \text{ valid in pool } i\}} c_{iv} \frac{2l!}{c_{i1}! c_{i2}! \cdots c_{id}!} f_1^{c_{i1}} f_2^{c_{i2}} ... f_d^{c_{id}} \tag{2.2}$$

with the normalization being implied. The maximization step updates each frequency estimate by assigning

$$\hat{f}_v = \sum_{i=1}^{n} \frac{\beta_v^i}{n}. \tag{2.3}$$

The E-step and M-step are performed iteratively until the algorithm converges.

2.2.3 Global Estimates from Linear Regression

Once we have local frequency estimates, which are frequencies for the projected candidate haplotypes, we can compute the global frequencies for the candidate haplotypes. For each given subset P of the columns, the haplotype frequencies of $\mathcal{H}^P$ give a linear constraint on the haplotype frequencies of the haplotypes in $\mathcal{H}$. In particular, for each haplotype $h_u \in \mathcal{H}^P$, we know that $\sum_{v:h_v^P=h_u} x_v = b_u$, where b_u is the local frequency of h_u computed by the EM, and x_v is the (global) frequency of candidate haplotype $h_v \in \mathcal{H}$. For instance, if P contains only one SNP j, then the above constraints imply that the sum of all haplotype frequencies with 1 in SNP j should be the frequency of the minor allele 1 in that SNP.

We use the EM algorithm to compute projected haplotype frequencies of all single SNPs, all pairs of SNPs, a set of $\binom{m}{2}$ random triples of SNPs, and a set of $\binom{m}{2}$ random quadruples of SNPs. Each of these runs gives a set of linear constraints. We use this strategy, because in practice it gives the finest-grained estimate of projected haplotypes that reliably increases the accuracy of our algorithm. Using longer projected haplotypes tends to hurt the running time without providing more accuracy. In addition, we use the linear constraint that the sum of all frequencies is exactly 1. Thus, we end up with a $\{0,1\}$ matrix C with d columns, and a vector b which corresponds to the projected haplotype frequencies. The resulting haplotype frequencies could then be computed by solving the set of equations $Cx = b$ for the frequencies $\mathcal{F} = \{x_1, x_2, ...x_d\}$. A cartoon of some constraints is given in Figure 2.3.

Unfortunately, due to errors in the frequency estimates for the different sets of SNPs, the different solutions are inconsistent, and therefore no solution exists to the equation $Cx = b$. We thus consider the least squares solution for this equation, i.e., we find $x \geq 0$ such that $\|Cx - b\|_2$ is minimized. We note that a similar approach was taken by [4, 23], where the equations corresponded to single SNPs, and the setting involved large DNA pools.

The above least squares procedure treats all constraints as if the components of the vector b are distributed according to a standard Gaussian. However, clearly, some constraints are more informative than others, and some are more error-prone than others. For instance, the constraints imposed by single SNP projections are more accurate than the constraints imposed by pairs of SNPs. In general, we find that the accuracy of the EM algorithm declines when the number of SNPs gets larger. It is therefore useful to give more weight to the constraints that are based on haplotypes projected over smaller numbers of SNPs. In particular, we can rewrite the objective function as $x^* = argmin_x \sum_u w_u \left(b_u - \sum_v C_{uv}x_v\right)^2$, where w_u represents the weight given for the u^{th} constraint. In particular, each w_u is the inverse of the variance expected of the residual, and it serves to standardize the error. If $w_u = 1$ for all u then this is simply the least squares formulation describe above. For an efficient practical implementation, we found that we could fix the weights as $w_u = 3m$ for single SNP projections and $w_u = 1$ for projections of two or more SNPs.

2.3 HaploPool

Both methods for estimating the haplotype frequencies have different strengths. As a result, we can combine the two algorithms into a single implementation that performs well on most data. Before introducing how the two algorithms are integrated into HaploPool, we will discuss the strengths of each model.

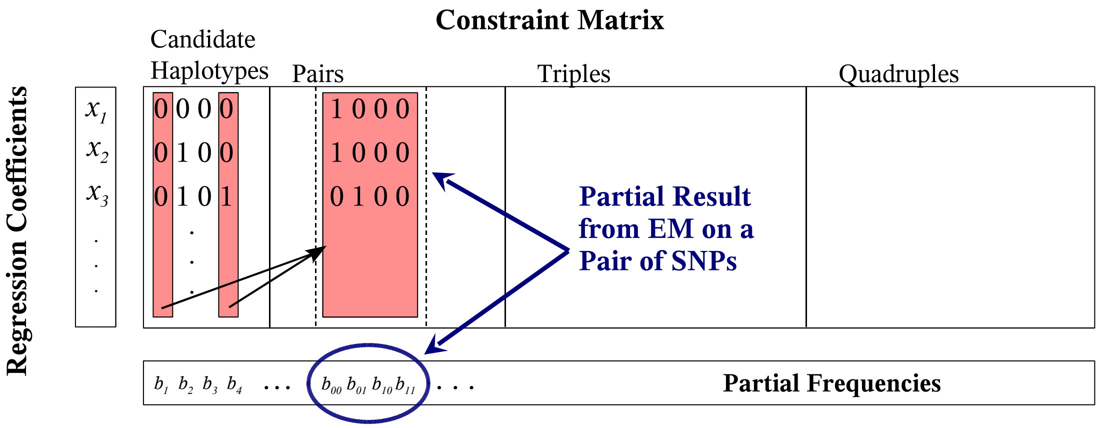


Figure 2.3. **Regression Matrix.** This is a cartoon of the regression matrix, the partial (or local) frequencies and the regression coefficients. Every candidate haplotype is listed as a row in the 'candidate haplotypes' section of the constraint matrix. Notice that the actual C matrix is the transpose of the one in the picture, because a row in the picture corresponds to a column of C. The constraints on single SNPs come from the allele frequencies, and each column in the candidate haplotypes section of the matrix corresponds to a single SNP. The constraints that fall in the section of the matrix labeled 'pairs' come from the EM results on projected SNP-pairs. Every four columns in the pairs section correspond to the four pair-haplotype frequencies computed by the EM. For example, b_{00} is the pair-haplotype frequency for the haplotype 00 on the pair of SNPs under consideration, which are indicated in red in the candidate haplotype section of the matrix. The haplotype frequencies for every candidate haplotype containing 00 at the respective SNP-pair are constrained to sum to b_{00} by the one-entries in the first column of the red pairs block. Similar constraints hold for b_{01}, b_{10}, and b_{11}. In the triples and quadruples portion of the matrix, there are respectively 8 and 64 columns (local haplotypes) devoted to each local EM frequency estimate.

When the maximum likelihood perfect phylogeny tree has been found, we have both an estimate of the haplotype frequencies and the number of recurrent mutations required to generate the data from the prefect phylogeny model. When the tree needs a small number of recurrent mutations to explain all of the pools ($\sum_i mut(c,i) < 2$), we observe that the resulting model gives a more accurate estimate $\hat{\mathcal{F}_T}$ for the haplotype frequencies than the regression method. However, when more recurrent mutations are needed, evidently the set of haplotypes in that region cannot be reasonably explained by the perfect phylogeny model.

When both the candidate haplotypes and the EM projection estimates are accurate, the regression method accurately estimates haplotype frequencies. The regression is most accurate when the candidate haplotypes inferred by the greedy method contain all the true common haplotypes. In practice the greedy algorithm almost always identifies all the common haplotypes, but it sometimes misses rarer haplotypes. The order in which the haplotypes are considered by the greedy algorithm may affect the number of rare haplotypes we obtain. We also observed that the EM algorithm produces fairly accurate estimates on short projected haplotypes, but is prone to errors as the number of SNPs in the projection grows.

In order to exploit the strengths of these two methods, HaploPool begins by computing the maximum likelihood perfect phylogeny model and obtaining the number of recurrent mutations needed to generate the data from that model. If the number of mutations is small, $\sum_i mut(c,i) < 2$, then the perfect phylogeny haplotypes, $\hat{\mathcal{H}_T}$, and their frequencies, $\hat{\mathcal{F}_T}$, become the final haplotype frequency estimates of HaploPool. If the number of mutations is at least 2, then HaploPool uses the perfect phylogeny haplotypes to initialize the greedy algorithm $\mathcal{H}_I = \hat{\mathcal{H}_T}$ and the ordering of the haplotypes is given by their perfect phylogeny frequencies. The greedy algorithm then yields a set of candidate haplotypes $\mathcal{H} = \hat{\mathcal{H}_T} \cup \mathcal{H}_g$. Following that, the EM and linear regression proceeds exactly as described. These two execution paths through HaploPool are the most useful, and they are represented in the flow diagram in Figure 2.4 as the first right branch (yes-branch at decision node 'Use Perfect Phylogeny?'). The decision node labeled 'Perfect Fit to Tree?' is answered yes if $\sum_i mut(c,i) < 2$.

The other execution path that allows HaploPool to only execute the regression algorithm (and skip the perfect phylogeny model entirely) was created to compare the accuracy of the two algorithms. When the perfect phylogeny model is not used, we initialize the greedy algorithm with the empty set $\mathcal{H}_I = \emptyset$ that represents a lack of knowledge about the haplotype set. Then following the completion of the greedy algorithm, the candidate haplotypes for the regression are $\mathcal{H} = \mathcal{H}_g$. This decision is represented in the HaploPool Flow Diagram (Fig. 2.4) as left branch at the first decision node.

These three versions allow us to test the effect of the perfect phylogeny model on the accuracy of the algorithm. As discussed in the results of Chapter 3, we observe that the version in which we use only the perfect phylogeny outperforms the other versions as long as there is a small number of recurrent mutations across all the pools. When the number of recurrent mutations gets larger, the combination of perfect phylogeny with greedy gives the best results. Thus, HaploPool chooses which algorithm to use for the final frequency estimate based on the number of recurrent mutations needed for the perfect phylogeny.

HaploPool Flow Diagram

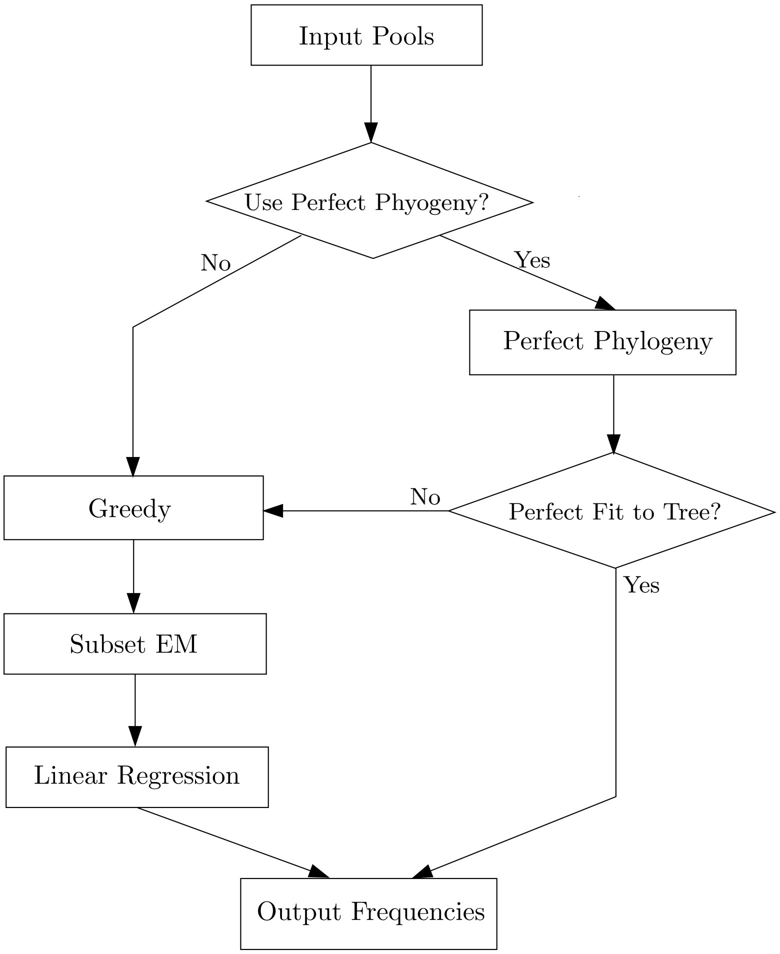


Figure 2.4. **HaploPool Flow Diagram.** At the core of the HAPLOPOOL algorithm are the four procedures located in the rectangular boxes and described in the Methods section. There are three possible paths from the input to the output in this flow diagram. We compare all three in the Results section. We found that the best strategy is to always use the perfect phylogeny and to let the algorithm decide whether the data fit the perfect phylogeny. When there is a small mutation number, the haplotype frequency estimates produced by the perfect phylogeny are very good, so the algorithm returns that estimate.

Chapter 3

Results

For pool-genotype data, we compared the accuracy of HaploPool to three state-of-the-art algorithms for haplotype frequency estimation from pool data. The three programs, LDPooled [15], EHP [30] and Pools2 [14], all used the EM algorithm to estimate haplotype frequencies. Pools2 actually estimated the genotype frequencies, and we used PHASE to estimate the haplotype frequencies from these genotypes. For non-pool genotype data, we also compared our results to the widely-used phasing method PHASE [27].

We simulated haplotypes and pools, in order to have a ground-truth to which the estimated frequencies could be compared. We carefully describe these simulations in Section 3.1. In the following Section 3.2, we discuss how increasing the pool size results in better accuracy when the number of pools is held constant. This leads to the questions of how efficient pooling is, and what sample cost is required for better accuracy (also Section 3.2). There are two important considerations about the robustness of these results: (1) noise in the genotype calls, and (2) various block properties either assumed by the simulation or a result of the frequency estimates, (Sections 3.3 and 3.4, respectively). Finally, Section 3.5 discusses the accuracy of HaploPool and discusses the PPM model compared with the regression model.

3.1 Simulation

In order to do the comparisons, we needed a gold-standard of haplotype frequencies against which we could compare the haplotype frequency estimates produced by each program. To obtain the gold-standard, we took phased haplotypes from the HapMap data set and generated additional haplotypes using recombination. Then we randomly selected from those haplotypes to create individuals and pools. The details of the simulation are discussed in this section.

Simulated Data Sets. To evaluate the performance of HaploPool we examined two data sets for which estimates of the haplotype frequencies were already available. These two data sets were derived by simulation from the phased haplotypes of the individuals in the CEU population from HapMap [28] phase 1. The CEU data was composed of thirty mother, father, child trios that were phased using PHASE [27]. The first data set consisted of SNPs from the first 9 Mb of Chromosome 19. The second data set contained two of the ENCODE regions, ENm010 and ENm013, which were each 500 Kb regions of Chromosome 7. For our study, we took the phased haplotypes for the sixty unrelated parents as the starting point to simulate unrelated individuals. For each data set, Chr19 and ENCODE, we simulated a total of 600 unrelated haplotypes using the Li and Stephens [18] recombination model to simulate haploid gametes from recombined haplotypes already in the data set. The haplotype frequencies of these 600 unrelated haplotypes made up our *gold-standard* population haplotype frequencies against which we compared all the phasing methods. From the haplotype frequencies of the unrelated haplotypes, we drew (with replacement) the diploid individuals which made up the pool or genotype samples which were given as input to each phasing algorithm.

The chromosomal regions of each data set were partitioned based on physical distance into 10kb blocks (more partitioning schemes are discussed in Sec. 3.4). In order to consider a more realistic case where the SNP density was less than one SNP every 300 bases, we retained every second SNP in the ENCODE regions. These simulated data sets had a variety of block sizes ranging from 2 to roughly 45 SNPs.

Measures of Accuracy. In order to compare the accuracy of frequency estimation between the different methods, we compared each method to the gold-standard population frequencies from the observed counts of the haplotypes in the set of 600 haplotypes generated by our simulations. We compared the predicted haplotype frequencies, f_i, from each data set of 60 pools to the gold-standard population level haplotype frequencies, g_i. We used two natural measures: the l_1 and χ^2 distances. Let the l_1 distance between two distributions f and g be defined as $l_1(f, g) = \sum_{i=1}^{d} |f_i - g_i|$. The χ^2 distance between the two distributions is simply the result of the χ^2 statistic where g is the expected distribution, i.e., $\chi^2(f, g) = \sum_{i=1}^{d} \frac{(f_i - g_i)^2}{g_i}$ where d is the number of gold-standard haplotypes.

3.2 Efficiency of Pooling over Non-Pooling

When holding the number of pools constant, the accuracy of haplotype frequency estimates improved with increasing pool size. This was observed for all the pool-genotype programs, although HAPLOPOOL achieved the best accuracy and running time. Since PHASE's non-pool haplotype frequency estimates were superior to those of all other programs, this observation led to the question of whether pool estimates with HAPLOPOOL were efficient when compared with the non-pool haplotype frequency estimates produced by PHASE. This section discusses these observations and uses the results of a simulation to quantify the efficiency of HAPLOPOOL relative to PHASE.

The results for the ENCODE regions are given in Figure 3.1. This plot illustrates two points. First, HAPLOPOOL consistently performed more accurately than Pools2 and LDPooled, while being as accurate as EHP. HAPLOPOOL was also much faster than the other computational methods for inferring haplotype frequencies from pools. Second, the most accurate estimation of the haplotype frequencies was achieved by using HAPLOPOOL on pools of size four. In fact, as the pool size grew, the accuracy improved, provided that there were no errors in the genotyping. This is especially interesting, since even though PHASE gave slightly better results than HAPLOPOOL for genotype

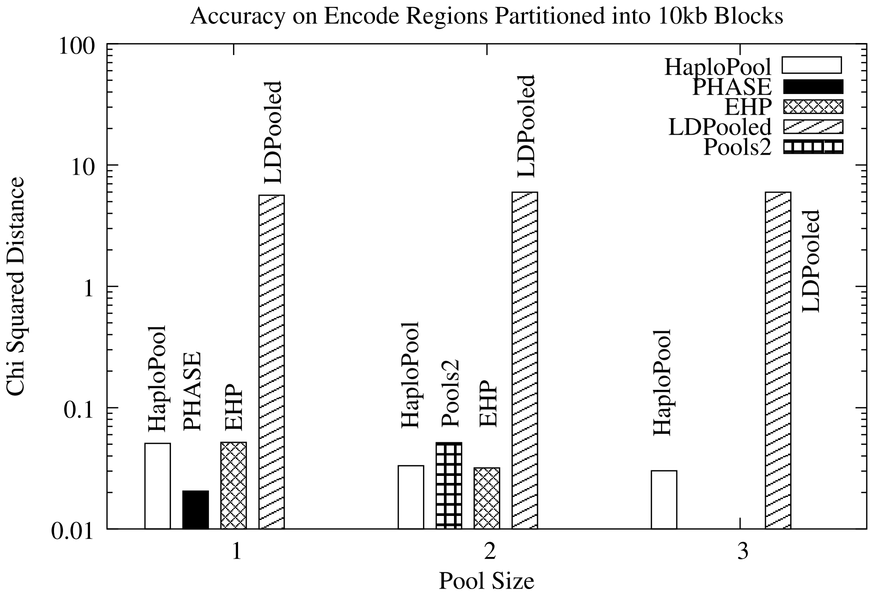


Figure 3.1. **Accuracy of Haplotype Frequency Estimates on Encode Regions (10kb Partitions).** This figure demonstrates a clear improvement in accuracy with with larger pool sizes. After simulating a large population to obtain the gold-standard frequencies for comparison, we partitioned the regions into blocks of limited diversity using 10kb physical distance. Each program ran on a random sample of 60 pools and returned frequency estimates. The χ^2 distance was computed from these frequencies and the gold-standard frequencies. Note that EHP appears only in this figure due to a running time in excess of a week on a 4-core 1.6GHz Intel Xeon Server.

data, it was beneficial to use HaploPool on pool data. For the same number of genotyping experiments with pools, we obtained more accurate information about the population from which the sample was drawn, and thus an association study would have gained power with no further genotyping costs.

In order to quantify the efficiency of using DNA pools and HaploPool we measured the increase in number of genotyping experiments needed by traditional genotyping in order to achieve the same accuracy as HaploPool. In other words, we compared the scenario in which we use HaploPool to estimate the haplotype frequencies from n pools of two samples each, with the scenario in which we use PHASE to estimate the haplotype frequencies from αn genotypes. As shown in Figure 3.2 these two scenarios provided similar levels of accuracy for the haplotype frequencies when $\alpha \approx 1.45$. In other words, we found empirically that the accuracy of a study that genotypes $2n$ individuals using n DNA pools of two samples each, is equivalent to the accuracy of a study that genotypes $1.45n$ individuals using non-pool genotyping. Since the cost of genotyping a pool is close to the cost of genotyping an individual, we saw that, in the second scenario, the cost of genotyping has increased by close to 45%. Again, we compared the cost of genotyping either pooled or non-pooled DNA, because we assumed that the cost of genotyping exceeds the cost of sampling and phenotyping.

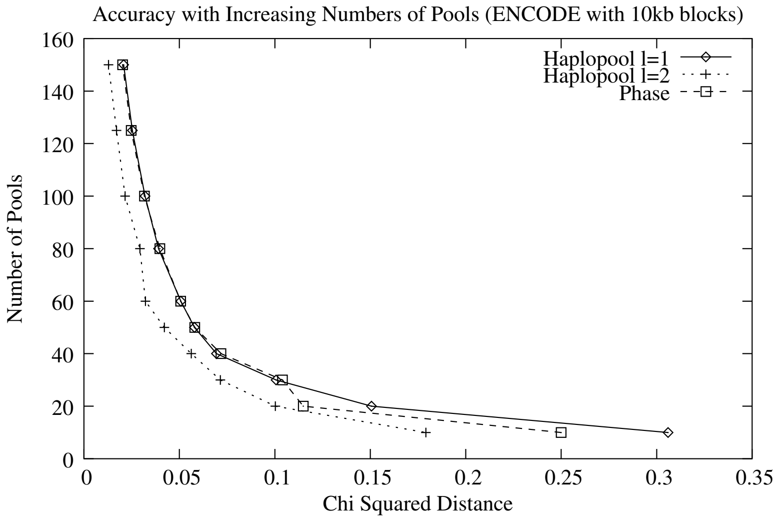
Running Times Three of the programs, HaploPool, PHASE, and LDPooled were run in Linux on an AMD Athlon 64 with a 3 GHz processor and 1000 MB. Pools2 is a Windows program and was executed on a Pentium 1.6 GHz with 512 MB. This program could not be directly compared with those of the other programs since Pools2 estimated genotype frequencies. We ran PHASE (for 9.91 seconds on the AMD) on the Pools2 genotype frequencies to obtain the haplotype frequency estimate. It is clear that Pools2 was slower than HaploPool due to the 2-stage estimation, regardless of the platform it ran on. Although EHP does not appear in the plot (due to infeasible running time), our other tests indicate that EHP was slower than the other methods.

3.3 Effect of Noise

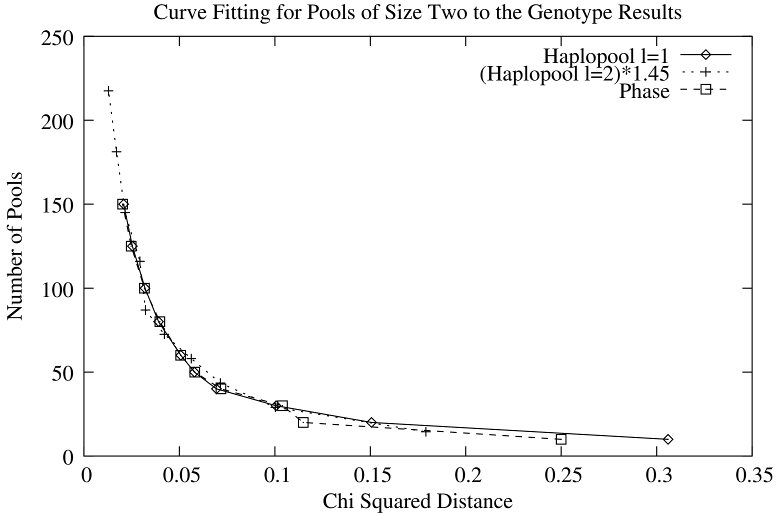
Even though the above results seem very promising, a justified objection to the use of DNA pools may be that the erroneous-call rates and no-call rates for DNA pools of two, three, or four individuals are higher than the rate for standard genotyping. There are two types of errors to consider: missing calls, where a genotype call is completely unknown, and erroneous calls, where a genotype was mistakenly called with the wrong frequency. In order to characterize the effect of these errors, we simulated missing and erroneous genotype calls with different amounts of error. We then computed haplotype frequencies with PHASE and HaploPool, and compared those estimates.

The erroneous-call and no-call rate for standard genotyping varies according to the platform, DNA quality and maintenance, and other factors, but it is reasonable to expect it to be on the order of $1\% - 2\%$ missing data and $< 1\%$ error rate. In [6] it was shown that for the Sequenom platform, the erroneous-call and no-call rates are on that order of magnitude for pools of two individuals.

The effect of missing data was simulated by randomly masking out each allele count independently with probability p. We measured the performance of the algorithms for values of p ranging from 0.01 to 0.05 as shown in Figure 3.4. For these instances, we compared the results of Hap-



(a)



(b)

Figure 3.2. **The Number of Input Pools Affects Accuracy.** Plot (a) shows the accuracy of HaploPool and PHASE on the Encode data set (partitioned by physical distance) as the number of pools increase. In (b), we see an adjusted curve for HaploPool's results for $l = 2$. The y-values for this curve were multiplied by 1.45. We observe that the three curves are nearly identical. Thus, one would have to genotype 45% more individuals in order for PHASE to achieve the same accuracy as HaploPool.

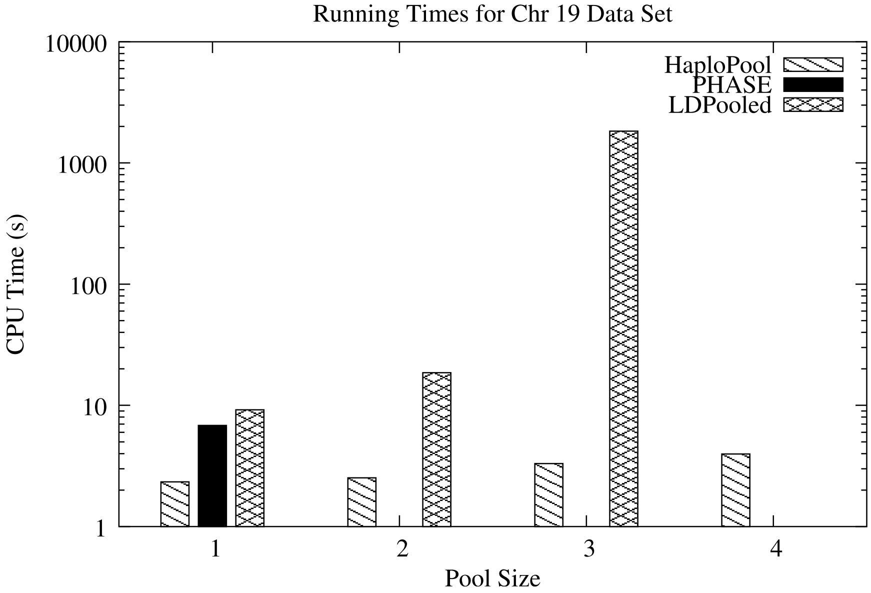


Figure 3.3. **Running Times on Chr 19 Data Set.** Three of the programs, HaploPool, PHASE, and LDPooled were run in Linux on an AMD Athlon 64 with a 3 GHz processor and 1000 MB. Since the running times of LDPooled were fairly slow, the y-axis was plotted with a log scale. Due to uncontrollable circumstances, Pools2 and EHP do not appear here, but it was clear from our tests that they were slower than the other programs.

loPool and PHASE as the other methods did not cope well with such data sets (LDPooled did not accept missing data as input, and Pools2 was too slow to run on these instances). Our study shows that the missing data rate did not affect the results substantially, and that even with 5% missing data and a fixed genotyping budget the accuracy of HaploPool with pools of size two was considerably better than the accuracy of PHASE on genotype data with no missing data.

We simulated erroneous genotyping calls by adding Gaussian errors with a standard deviation σ to each called allele frequency. Let the correct allele frequency be a_{ij} where i is the pool and j is the SNP, and let $\hat{a}_{ij}$ be the miscalled frequency. Then we picked the value $\hat{a}_{ij}$ from the distribution $\hat{a}_{ij} = a_{ij} + x$ where $x \sim \mathcal{N}(0, \sigma)$. Figure 3.5 shows the results of these simulations for σ ranging from 0 to 0.06. Note that traditional genotyping was not affected by these values of σ, as the difference in frequency between the possible observations of frequencies $(0, 0.5$ and $1)$ was much larger than the standard deviation. Notably, Figure 3.5 shows that as long as the standard deviation of the allele frequency call was smaller than 0.05, estimating haplotype frequencies from pools of size 2 was advantageous over non-pool genotyping.

3.4 Robustness to Varying Block Conditions

Since every result shown previously was computed after splitting a genomic region into blocks, a natural question is whether haplotype frequency estimates are robust to the properties of a block. First, we duplicated the above simulations using several block partitioning methods, and the results were indeed robust to the partitioning method. Second, we found that blocks with a

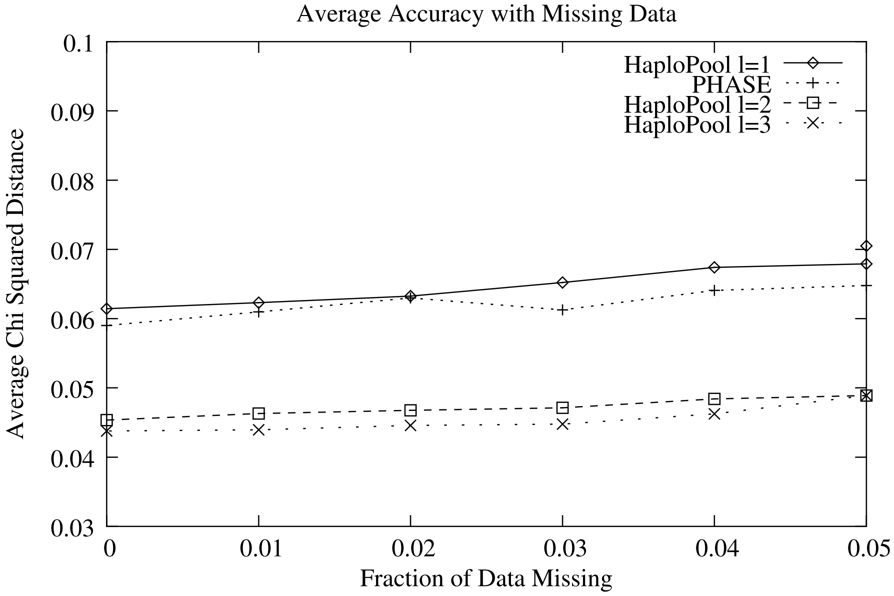


Figure 3.4. **Missing Data in Chr 19 Data Set.** Here we see the effect of missing data on the accuracy of the haplotype frequency estimates as the fraction of missing calls was increased. HaploPool was robust when presented with missing calls at rates seen on existing genotyping platforms. Note that HaploPool, $l = 2$, performed better with data missing at a rate of 5% than PHASE did with no missing data.

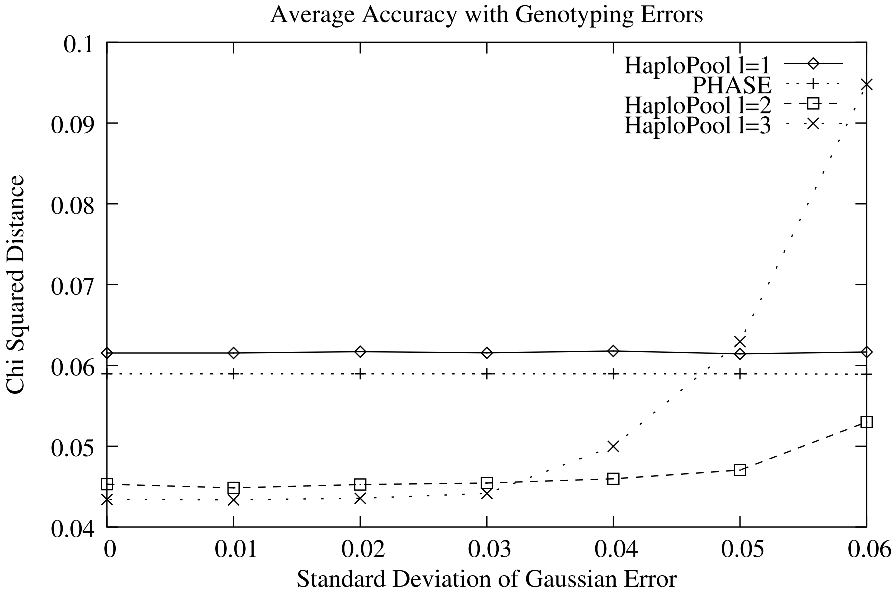


Figure 3.5. **Genotyping Errors in Chr 19 Data Set.** Erroneous genotyping calls had a greater effect on accuracy when the pools were of a larger size. The estimates were unaffected by the error when σ was smaller than the distance in frequency between calls that could be mistaken for each other. Since genotypes have a frequency leeway of 0.5, the calls, and thus the accuracy, were unaffected. Pools of size 2 provided the advantage of increased accuracy while remaining robust to bad calls when $\sigma < 0.05$.

limited diversity of haplotypes (i.e. few haplotypes with large haplotype frequencies) had better haplotype frequency estimates than blocks with greater diversity.

HapMap Based Partitions. To see whether our results were robust to the method used for block partitioning, we independently used three partitioning methods. Each partitioning method was executed as a pre-processing step on the data in the HapMap database. When running an actual study, one could use the SNPs in HapMap to partition the genome into blocks. Thus, human genotype data can be partitioned using the HapMap data.

As already mentioned the first partitioning method was based on a physical distance 10kb and retained half of the SNPs in each block. For the second partitioning method, we used Hap [10] to split each HapMap region into blocks such that each block had at most five haplotypes that account for 90% of the population. Third, we split the regions into blocks using Haploview's 4-gamete partitioning [5]. The number of SNPs in the blocks partitioned by Haploview ranged from 3 to 45, with 90% of the blocks having had 20 or fewer SNPs. Hap produced smaller blocks, with lengths between 3 and 10 SNPs.

The results for Chromosome 19 (partitioned by Hap) were very similar to the results discussed above and are given with running-times in Table 3.1. The Hap partitioning of the ENCODE regions yielded similar results (data not shown), with HAPLOPOOL having performed slightly better than PHASE on genotype data. For this same data set we again performed a simulation for varying numbers of pools (data not shown) and had very similar results to Figure 3.2.

Pool Size	**Algorithm**	**Average l_1**	**Average χ^2**	**CPU Time (s)**
1	HAPLOPOOL	0.148	0.062	**2.34**
1	PHASE	**0.144**	**0.059**	6.82
1	LDPooled	0.175	0.779	9.18
2	HAPLOPOOL	**0.121**	**0.046**	**2.52**
2	Pools2	0.158	0.086	-
2	LDPooled	0.136	0.662	18.63
3	HAPLOPOOL	**0.112**	**0.044**	**3.31**
3	LDPooled	0.118	0.321	1832.12
4	HAPLOPOOL	**0.106**	**0.043**	**3.97**
4	LDPooled	-	-	-

Table 3.1. **Accuracy on the Chr 19 data set for pools of varying sizes.** Hap [10] split this data set into 410 blocks ranging in length from 3 to 10 SNPs. Each program ran with varying pool sizes and 60 pools input for each run. We list the average l_1 and χ^2 distances between the estimated frequencies and the gold-standard frequencies, and the bold numbers indicate the best performance for a given pool size. The discrepancy for LDPooled between χ^2 and l_1 distances follows from the fact that for a few blocks LDPooled performed very poorly and that the χ^2 distance is sensitive to this.

Limited diversity indicated accurate frequency estimates. Both PHASE and HAPLOPOOL produced frequency estimates with varying degrees of accuracies over all the blocks. It might be useful to characterize the circumstances under which we can have high confidence in our

Pool Size	Algorithm	Average l_1	Average χ^2
1	HaploPool	0.118	0.0481
1	PHASE	0.119	0.0493
1	LDPooled	**0.117**	**0.0477**
2	HaploPool	0.104	0.0320
2	Pools2	-	-
2	LDPooled	**0.101**	**0.0293**
3	HaploPool	**0.0861**	**0.0243**
3	LDPooled	-	-

Table 3.2. **Accuracy and pool sizes for blocks with $>$ 90% coverage.** These are the average l_1 and χ^2 distances for the 22 blocks whose results have high coverage. The high coverage sets for HaploPool ($l = 1, 2, 3$), LDPooled $l = 1$ and PHASE were very similar, and we took the intersection. Here, bold indicates the best performance among the programs for each specified pool size. EHP does not appear due to infeasible running time.

predictions. This characterization is also applicable when the haplotype frequencies are predicted from traditional genotype data, and not only from DNA pools.

We observed that limited diversity can serve as a predictor of the accuracy of haplotype frequency estimations. All the methods (HaploPool, PHASE, LDPooled, and Pools2), were more accurate in blocks for which the set of common haplotypes (with frequency at least 5%) accounted for more than 90% of the population (referred to as 90% *coverage* in Table 3.2). As can be seen in Figure 3.6, for the blocks with lower coverage, the haplotype frequencies were estimated poorly by HaploPool. The same figure shows a similar trend when using PHASE on the same data. In particular, as shown in Figure 3.7, for the haplotype frequencies estimated for the blocks generated by Haploview, the advantage of using pools was not substantial unless we restricted ourselves to the blocks where there was limited diversity.

Thus, when performing haplotype based analysis in association studies, we recommend using the HapMap database to devise a partitioning for the genomic regions under consideration. After collecting data and estimating haplotype frequencies, we recommend focusing haplotype-based statistical tests on blocks for which the coverage is at least 90%. Since blocks with less coverage tend to have less reliable haplotype estimates, p-values obtained from them could be inflated and misleading. Rather than risk false-negatives due to haplotype frequency estimation, we recommend using allele-frequency statistics on low coverage blocks. These recommendations are universal, and they apply regardless of whether the study uses pooled or non-pooled DNA and regardless of which phasing program is used.

3.5 Analysis of HaploPool

Although HaploPool was designed to solve the haplotype frequency estimation problem, we also implemented a method for assigning haplotypes to pools by using the haplotype frequency estimates. The accuracy of these assignments is discussed below. In addition, we compared the haplotype frequency estimation accuracy of the perfect phylogeny model to the regression model.

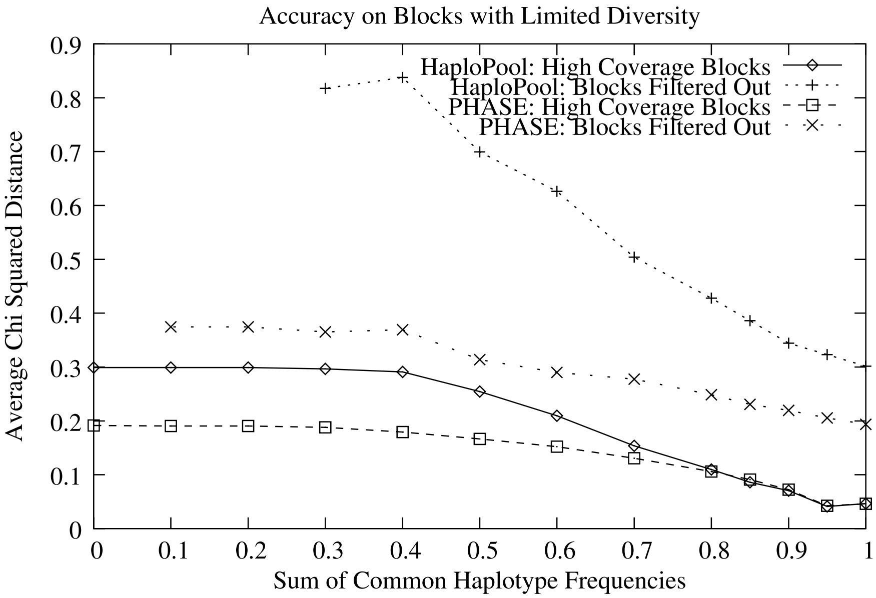
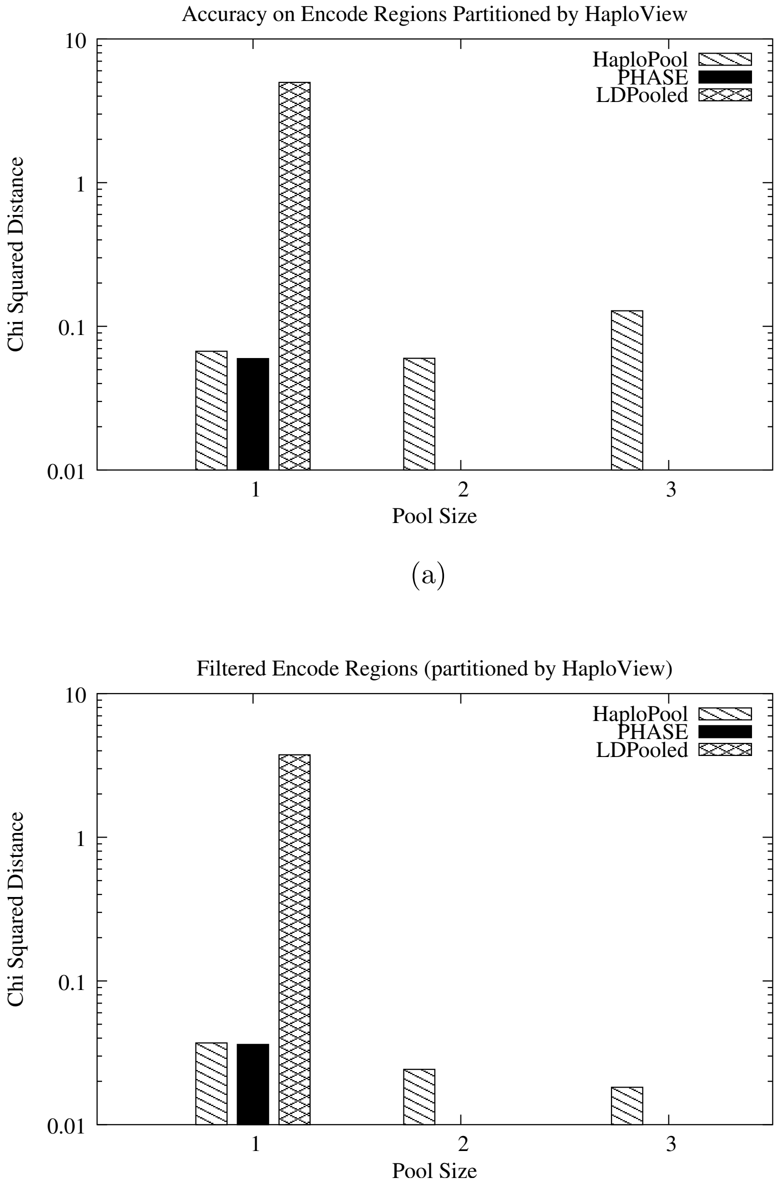


Figure 3.6. **Effect of Haplotype Diversity on Genotype Accuracy.** Blocks whose HaploPool and PHASE estimates contained haplotypes of limited diversity were more accurate than blocks with more diversity. Here limited diversity means that common haplotypes dominate the estimated frequencies. The lines marked with pluses and stars give the accuracy of HaploPool and PHASE respectively when given non-pool genotypes on the blocks with diversity limited by the threshold on the x-axis. The x- and box-marked lines are the accuracies for each method on the blocks that were filtered out by the coverage threshold. The data used here was obtained by creating blocks of 10 SNPs each from the first 9 Mb of Chr 19.



(a)

(b)

Figure 3.7. **Accuracy of Haplotype Frequency Estimates on Encode Regions Partitioned by Haploview.** The data shown here is from the coding regions of Chromosome 7. Haploview partitioned the data by adding SNPs to a block as long as the observed gametes (in all column pairs) had frequency greater than 0.01. For each block, we simulated a large population for the gold-standard frequencies, and then drew samples of 60 pools from the population for the input data. Plot (a) shows the χ^2 distance between each programs' estimates and the gold-standard estimates. Plot (b) gives the results only on blocks of limited diversity, where the common haplotypes accounted for more than 95% of the estimates. In both (a) and (b), the distances were plotted on a log-scale for the y-axis, because LDPooled performed poorly on some blocks and this was reflected in it's χ^2 value. LDPooled also failed to yield estimates for pools of size 2 or 3, probably because of the block lengths, which ranged from 3 to 45 SNPs. EHP does not appear due to infeasible running time.

	HaploPool without perfect phylogeny		**HaploPool**	
Pool Size	**Average l_1**	**Average $\chi 2$**	**Average l_1**	**Average $\chi 2$**
1	0.1295	0.0491	0.1302	0.0502
2	0.0978	0.0302	0.0961	0.0294
3	0.0901	0.0266	0.0853	0.0242
4	0.0897	0.0325	0.0774	0.0218

Table 3.3. **The Perfect Phylogeny on Encode Data Set Partitioned by Hap.** We compared the performance of HAPLOPOOL with and without the perfect phylogeny. The addition of the perfect phylogeny contributed a 5% increase in l_1 accuracy for pools of size 3 and 14% for pools of size 4.

From the results shown below, we found that the perfect phylogeny model improved the accuracy of frequency estimates from pooled data.

Haplotype Assignments for Pools. Our method is capable of predicting the haplotypes that comprised each input pool. This is done by using the estimated frequencies to choose the maximum likelihood haplotype configuration for each pool. In the case of multiple individuals per pool, haplotypes are determined for the whole pool, rather than for each individual. To determine haplotype prediction error, we counted the haplotypes in each prediction which did not occur in the gold-standard input data. On the data set which was a 10kb partitioning of two ENCODE regions of Chr. 7, our method correctly assigned haplotypes to genotype data with an average error rate of 1.3% and standard deviation of 2.6%. Pools of size two had slightly more errors with a mean error rate of 2.6% and a standard deviation of 4%.

Perfect Phylogeny Model and the Regression Method. We measured the effect of the perfect phylogeny model on the accuracy of our method. We compared the following three variants of our algorithm. First, we used the HAPLOPOOL algorithm, as described in the Methods section; that is, we use the perfect phylogeny and the greedy algorithm to estimate a set of plausible haplotypes $\mathcal{H} = \hat{\mathcal{H}}_T \cup \mathcal{H}_g$, and we then use the EM and linear regression to estimate the haplotype frequencies of $\mathcal{H}$. In order to compare this to the perfect phylogeny model, we compared this to the variant in which we used the perfect phylogeny to estimate the haplotypes $\hat{\mathcal{H}}_T$ and their frequencies $\hat{\mathcal{F}}_T$, as described in the Methods section, but without using the greedy algorithm and the other extensions of the algorithm. Finally, we used the combination of EM and linear regression on the set of haplotypes $\mathcal{H}_g$ produced by the greedy algorithm, but without using the perfect phylogeny model at all. As shown in Table 3.3, the use of the perfect phylogeny as an initial step for the generation of the set $\mathcal{H} = \hat{\mathcal{H}}_T \cup \mathcal{H}_g$ is especially beneficial when two or more individuals are used for each pool. In these cases, we achieve an improvement of 14% in the l_1 statistic as compared to the variant in which the set $\mathcal{H}_g$ is generated by the greedy algorithm alone. Furthermore, as can be seen in Table 3.4, if we focus our attention on blocks in which no recurrent mutations were needed, i.e., blocks for which the perfect phylogeny can fully explain the data, then the most accurate algorithm is simply the variant that only uses the perfect phylogeny.

	Perfect Phylogeny		**HaploPool without Perfect Phylogeny**		**HaploPool**	
Pool Size	**Avg** l_1	**Avg** $\chi 2$	**Avg** l_1	**Avg** $\chi 2$	**Avg** l_1	**Avg** $\chi 2$
1	0.1081	0.0389	0.1078	0.0358	0.1079	0.0361
2	0.0792	0.0209	0.0864	0.0245	0.0825	0.0213
3	0.0683	0.0163	0.0813	0.0221	0.0713	0.0162
4	0.0676	0.0195	0.0795	0.0238	0.0668	0.0153

Table 3.4. **Blocks with No Mutations: The Perfect Phylogeny on the Encode Data Set Partitioned by Hap.** On blocks which perfectly fit the perfect phylogeny, we compared the performance of the three algorithms, as described in Figure 2.4, that is: HAPLOPOOL, HAPLOPOOL without the perfect phylogeny, and the perfect phylogeny alone. The measurements shown are the average χ^2 and l_1 distances for pools of sizes ranging from one to four. For these blocks, the perfect phylogeny frequency estimates were the most accurate, followed by the frequencies estimated using the perfect phylogeny haplotypes and HAPLOPOOL.

Chapter 4

Discussion

In this thesis report, we advocate using DNA pools of a small number of individuals when performing case-control association studies with haplotype frequencies. To this end, we introduced HaploPool, a novel method for inferring haplotype frequencies from small disjoint pools of unrelated individuals. We provided simulation results that demonstrate, for a fixed number of genotype experiments, the accuracy of HaploPool relative to the objective standard of traditional genotyping.

The effect of simulated errors suggests that pools of two individuals provide a good balance between increasing accuracy and decreasing robustness to error (Figure 3.5). It appears that such small pools are robust to realistic values of genotyping errors. Clearly, the effect of error rate on HaploPool's accuracy depends heavily on the specific genotyping platform, and we believe that benchmarking these methods on different platforms is essential. Since errors exhibit a large influence on pools of size greater than two, we conclude that previous, large-pool, methods are not competitive with traditional genotyping for haplotype frequency estimation.

Although small-pool techniques are most directly applicable for conducting case-control studies using haplotype frequencies, small pools can also be used to analyze data with a quantitative phenotype. Quantitative traits can be analyzed by grouping the individuals into pools of similar characteristics. For example, if the distribution of a trait is normal, individuals falling into the same standard deviation can be grouped into pools. The idea is that individuals with similar quantitative values might have similar genetic factors that are correlated with the expression of the trait. For this reason, the use of small pools is advantageous since small pools can be chosen to minimize the variance of the quantitative phenotype within each pool.

HaploPool has two algorithms built into it. The first algorithm is based on the perfect phylogeny model, and the second is a regression analysis that aggregates frequency predictions for subsets of SNPs. Since the perfect phylogeny model works well when the data fit a perfect phylogeny, and the regression analysis works well the data do not fit a perfect phylogeny, these two algorithms are combined into a single implementation.

Although perfect phylogeny haplotyping has been used in the past for phasing [10], deviations from the model were treated in a less formal manner than in the present paper, and previous algorithms could not extend easily to DNA pools. The model used here, although designed to work with DNA pools, can also be used for phasing traditional genotype data. However, we observed here that the perfect phylogeny improves the accuracy of predictions more significantly for DNA pools. Pools allow more options for which alleles can be paired together in the same haplotype, so the constraints given by the perfect phylogeny model have more influence on the estimate.

The results in this paper illustrate that as opposed to common perceptions, DNA pools can be used to obtain more accurate haplotype frequency estimates than traditional genotyping methods using an equal number of genotyping experiments. Indeed, we show that by using pools of two or three individuals, one can obtain better frequency estimates than are available from performing the same number of genotyping experiments using traditional genotyping. Since pools of size two and three use the same genotyping technology, the cost for running the pooling experiments is roughly equivalent to running normal genotype experiments. There is a small increase in the cost due to quantizing the DNA samples, but as noted in [6], this cost is negligible compared to the cost of genotyping. In [6], it is reported that, in their hands, this cost is \$5 per sample. Hence, according to our study, genotyping n pools of size 2 is roughly equivalent in cost to genotyping n individuals using traditional genotyping, but the haplotype frequency estimates are roughly equivalent in accuracy to those of $1.45n$ individuals using non-pool genotyping.

Here, we assume that the per-person cost of genotyping dominates the cost of sample collection and phenotyping, and that the cost of genotyping largely determines the cost of a study. If sample collection is more expensive than genotyping, as in the cases of rare diseases or diseases that are difficult to assay, DNA pooling may no longer be cost-efficient. We use the accuracy results mentioned in the preceding paragraph to make a rough estimate of when pooling is cost-effective. Assuming that the cost of genotyping is equivalent for non-pool and pool genotyping (i.e. entirely neglecting the \$5 per sample quantization cost), the cost of genotyping is efficient when the cost of $0.45n$ genotypes exceeds the cost of $0.55n$ samples. This is because the non-pool case (requiring less samples) would need to perform 45% more genotype experiments to achieve the same accuracy as pooling. Yet, the sample-size cost of the completing all n pools in the pool experiment is 40% more samples in pools than the non-pool case. So, we have that 0.82 genotypes must be more expensive than collecting a single sample. With the cost of genotyping being roughly \$0.002 to \$0.006 per SNP for the 100,000 SNP array, sample collection costs must exceed \$164 to \$495 per person before pooling becomes less cost-effective than non-pool genotyping.

Following from these computations, we can see that pooling is cost-effective in many scenarios which involve human genetics and model organism genetics. For human diseases which are not too rare or difficult to assay, the sampling and phenotyping costs can be less than the genotyping costs, particularly when genotyping 100k or more SNP. In addition, many model organisms are grown cheaply in the laboratory, and sampling and phenotyping are very inexpensive. Indeed for some new model organisms a SNP chip may not exist, and more expensive genotyping technologies must be used. Thus, pooling strategies might be very valuable to the model organism communities.

Previous results by Yang et al [30] showed that DNA pooling experiments are more efficient than genotyping on blocks of two and three SNPs. However, the results did not establish the efficiency of DNA pooling relative to an objective standard such as PHASE, the state-of-the-art non-pool method. We also observe that their findings are highly dependent on the haplotype distribution, and blocks with many SNPs or a large diversity of haplotypes (such as in Fig. 3.7) have infeasible running times.

The novel results offered here show that the accuracy of pooling is competitive with traditional genotyping even in the presence of errors. Our comparison with previous methods shows that the other methods for haplotype frequency estimation from pools are probably not competitive with PHASE. Both the improved running time and the better accuracy obtained by our method over previous approaches made this conclusion possible. In particular, the running time of HAPLOPOOL makes it feasible to estimate whole-genome haplotype frequencies, after partitioning the genome.

Bibliography

[1] K. G. Ardlie, L. Kruglyak, and M. Seielstad. Patterns of linkage disequilibrium in the human genome. *Nature Reviews Genetics*, 3, 2002.

[2] V. Bafna, D. Gusfield, G. Lancia, and S. Yooseph. Haplotyping as perfect phylogeny: A direct approach. *Journal of Computational Biology*, 10(3-4):323–340, 2003.

[3] LF Barcellos, W Klitz, LL Field, R Tobias, AM Bowcock, R Wilson, MP Nelson, J Nagatomi, and G. Thomson. Association mapping of disease loci, by use of a pooled DNA genomic screen. *American Journal of Human Genetics*, 61(3):734–47, 1997.

[4] BJ Barratt, F Payne, HE Rance, S Nutland, JA Todd, and DG Clayton. Identification of the sources of error in allele frequency estimations from pooled DNA indicates an optimal experimental design. *Ann Hum Genet*, 66:393–405, 2002.

[5] J.C. Barrett, B. Fry, J. Maller, and M.J. Daly. Haploview: analysis and visualization of ld and haplotype maps. *Bioinformatics*, 2005.

[6] K.B. Beckman, K.J. Abel, A. Braun, and E. Halperin. Using DNA pools for genotyping trios. *Nucleic Acid Research*, 34(19):e129, 2006.

[7] CS Carlson, MA Eberele, L Kruglyak, and DA Nickerson. Mapping complex disease loci in whole-genome association studies. *Nature*, 429(6990):446–52, 2004.

[8] Z. Ding, V. Filkov, and D. Gusfield. A linear-time algorithm for perfect phylogeny haplotyping. *Journal of Computational Biology*, (2):522–553, 2006.

[9] Gusfield. Haplotyping as perfect phylogeny: Conceptual framework and efficient solutions. In *Proceedings of the 6th Annual International Conference on (Research in) Computational (Molecular) Biology*, 2002.

[10] E. Halperin and E. Eskin. Haplotype reconstruction from genotype data using imperfect phylogeny. *Bioinformatics*, 20(12):1842–49, 2004.

[11] E. Halperin and R. Karp. The minimum-entropy set cover problem. In *Proceedings of the 31st International Colloquium on Automata, Languages and Programming, Bologna, Italy*, 2004.

[12] E. Halperin and R.M. Karp. Perfect phylogeny and haplotype assignment. In *RECOMB*, 2004.

[13] L. H. Hartwell, L. Hood, M. L. Goldberg, A. E. Reynolds, L. M. Silver, and R. C. Veres. *Genetics: From Genes to Genomes*. McGraw-Hill, New York, 2nd edition, 2004.

[14] J. Hoh, F. Matsuda, X. Peng, D. Markovic, M.G. Lathrop, and J. Ott. SNP haplotype tagging from DNA pools of two individuals. *BMC Bioinformatics*, 4(14), 2003.

[15] T. Ito, S. Chiku, E. Inoue, M. Tomita, T. Morisaki, H. Morisaki, and N. Kamatani. Estimation of haplotype frequencies, linkage-disequilibrium measures, and combination of haplotype copies in each pool by use of pooled DNA data. *American Journal of Human Genetics*, 72:384–398, 2003.

[16] B. Kirkpatrick, C. Santos Armendariz, R. M. Karp, and E. Halperin. HaploPool: Improving haplotype frequency estimation through DNA pools and phylogenetic modeling. *Bioinformatics*, 23:3048–3055, 2007.

[17] S. Le Hellard, S. J. Ballereau, P. M. Visscher, H. S. Torrance, J. Pinson, S. W. Morris, M. L. Thomson, C. A. M. Semple, W. J. Muir, D. H. R. Blackwood, D. J. Porteous, and K. L. Evans. SNP genotyping on pooled DNAs: comparison of genotyping technologies and a semi automated method for data storage and analysis. *Nucl. Acids Res.*, 30(15):e74–, 2002.

[18] N. Li and M. Stephens. Modeling linkage disequilibrium and identifying recombination hotspots using single-nucleotide polymorphism data. *Genetics*, 165(4):2213–33, 2003.

[19] Jonathan Marchini, David Cutler, Nick Patterson, Matthew Stephens, Eleazar Eskin, Eran Halperin, Shin Lin, Steve Qin, Goncalo Abecassis, Heather Munro, and Peter Donnelly. A comparison of phasing algorithms for trios and unrelated individuals. *American Journal of Human Genetics*, 78:437–450, 2006.

[20] F.R. McMorris. On the compatibility of binary qualitative taxonomic characters. *Bulletin of Mathematical Biology*, 39, 1977.

[21] Gilean A. T. McVean, Simon R. Myers, Sarah Hunt, Panos Deloukas, David R. Bentley, and Peter Donnelly. The Fine-Scale Structure of Recombination Rate Variation in the Human Genome. *Science*, 304(5670):581–584, 2004.

[22] N Patil, AJ Berno, DA Hinds, WA Barrett, JM Doshi, CR Hacker, CR Kautzer, DH Lee, C Marjoribanks, DP McDonough, BT Nguyen, MC Norris, JB Sheehan, N Shen, D Stern, RP Stokowski, DJ Thomas, MO Trulson, KR Vyas, KA Frazer, SP Fodor, and DR Cox. Blocks of limited haplotype diversity revealed by high-resolution scanning of human chromosome 21. *Science*, 294(5547):1719–23, Nov 23 2001.

[23] I. Pe'er and J.S. Beckmann. Resolution of haplotypes and haplotype frequencies from SNP genotypes of pooled samples. In *RECOMB '03: Proceedings of the seventh annual international conference on Research in computational molecular biology*, pages 237–246, New York, NY, USA, 2003. ACM Press.

[24] R. Plomin, L. Hill, I. W. Craig, P. McGuffin, S. Purcell, P. Sham, D. Lubinski, L. A. Thompson, P. J. Fisher, D. Turic, and M. J. Owen. A genome-wide scan of 1842 DNA markers for allelic associations with general cognitive ability: A five-stage design using DNA pooling and extreme selected groups. *Behavior Genetics*, 31(6):497–509, 2001.

[25] C Semple and M Steel. *Phylogenetics.* Oxford University Press, 2003.

[26] P. Sham, J. S. Bader, I. Craig, M. O'Donovan, and M. Owen. DNA pooling: a tool for large-scale association studies. *Nature Rev. Genet.*, pages 862–871, 2002.

[27] M. Stephens, N. Smith, and P. Donnelly. A new statistical method for haplotype reconstruction from population data. *American Journal of Human Genetics*, 68:978–989, 2001.

[28] The International HapMap Consortium. The international HapMap project. *Nature*, 426:789–796, 2003.

[29] David G. Wang, Jian-Bing Fan, Chia-Jen Siao, Anthony Berno, Peter Young, Ron Sapolsky, Ghassan Ghandour, Nancy Perkins, Ellen Winchester, Jessica Spencer, Leonid Kruglyak, Lincoln Stein, Linda Hsie, Thodoros Topaloglou, Earl Hubbell, Elizabeth Robinson, Michael Mittmann, Macdonald S. Morris, Naiping Shen, Dan Kilburn, John Rioux, Chad Nusbaum, Steve Rozen, Thomas J. Hudson, Robert Lipshutz, Mark Chee, and Eric S. Lander. Large-Scale Identification, Mapping, and Genotyping of Single-Nucleotide Polymorphisms in the Human Genome. *Science*, 280(5366):1077–1082, 1998.

[30] Y. Yang, J. Zhang, J. Hoh, F. Matsuda, P. Xu, M. Lathrop, and J. Ott. Efficiency of single-nucleotide polymorphism haplotype estimation from pooled DNA. *Proceedings of the National Acadamy of Science*, 100:7225–7230, 2003.

Index of Notation

Notation	**Description**
a_{ij}	the allele count, or number of 1-alleles, observed at SNP j in pool i
c_i	For a particular pool i, the configuration, c_i is a vector of haplotype counts such that for each haplotype, h_k, c_{ik} gives the count of that haplotype.
f_k	the frequency f_k of the k-indexed haplotype h_k
h_k	$h_k \in \{0,1\}^m$, where $h_{k,j} = 1$ when the haplotype has the minor allele at SNP j
i	the index of a pool such that $1 \leq i \leq n$
j **or** j_v	the index of a SNP such that $1 \leq j \leq m$. In some cases it is necessary to index j by a second variable v.
l	the number of diploid individuals pooled together
m	the number of SNPs in a data set
n	the number of pools in a data set
p_j	the sample-wide allele frequency of the 1-allele for SNP j: $p_j = \frac{1}{2nl} \sum_i a_{ij}$
T	a rooted perfect phylogeny tree with each node representing a haplotype and each edge representing a SNP. The root is the all-zero haplotype.
valid c_i	When $a_{i,j} = \sum_k h_{kj} c_{ik}$, a configuration is *valid* for pool i.

Index