The Pennsylvania State University

The Graduate School

Department of Statistics

DENSITY ESTIMATION AND MODAL BASED METHOD FOR HAPLOTYPING AND RECOMBINATION

A Dissertation in

Statistics

by

Xianyun Mao

© 2010 Xianyun Mao

Submitted in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

December 2010

The dissertation of Xianyun Mao was reviewed and approved* by the following:

Bruce G. Lindsay

Willaman Professor of Statistics

Head of the Department of Statistics

Dissertation Adviser Chair of Committee

Naomi S. Altman

Associate Professor of Statistics

Yu Zhang

Assistant Professor of Statistics

Stephen W. Schaeffer

Associate Professor of Biology

*Signatures are on file in the Graduate School.

Abstract

Genetic problems such as haplotype inference and recombination analysis are rarely studied using nonparametric models. We present here some new methods based on kernel density estimation and a modal expectation-maximization (MEM) method for analyzing genetic data. We also use a degree of freedom (DOF) calculation for bandwidth selection and diagnostics.

For the problem of inferring haplotypes from genotypes, we construct a likelihood function that depends on the unknown haplotype density. We then apply a likelihood EM to a naive initial estimator to create an updated density that has higher likelihood. The density is then used to find the most likely haplotype pairs for any genotype. The performance of the method is tested on simulated data and small sets of real data. To improve the performance of our method for large data (~1,000 individuals, 10,000 sites), we develop degrees of freedom (DOF) as a diagnostic tool to partition large data. We then use MEM to solve each partition and to merge the solutions. We show that the new method yields comparable performance to available methods both in time and in accuracy. In a similar fashion, we can define a density estimator for binary sequences (haplotypes) in the presence of recombination and mutation. With the new density, one can estimate the probability of recombination for given sites.

Table of Contents

List of Tables

List of Figures

Acknowledgments

I am deeply indebted to my advisor, Dr. Bruce Lindsay for his invaluable guidance, support and patience in the past years. His dedication to research and his students gave me strength to complete the work and will always remind me of how a professor should work and teach. Deepest gratitude are also due to the members of the dissertation committee. I thank Dr. Yu Zhang and Dr. Stephen Schaeffer for the inspiring conversations over some of topics related to my work. I thank Dr. Naomi Altman not only because of her assistance to the dissertation work but also because it was she who helped and encouraged me to join this program and start studying statistics five years ago.

I has my deepest gratitude to Dr. Mark Shriver who has supported and inspired me in research for many years. He also offered tremendous help when I decided to study statistics. I want to also thank Dr. Kateryna Makova for her inspiration to me on bioinformatics.

I would like thank all my graduate friends, Zhe Chen, Yijia Feng, Yeojin Chung, Sham Bhat, Huei-wen Teng, Andreas Artemiou, Lu Zhang and many others. I especially need to thank Jianping Sun for many useful discussions over my projects. I would like also thank all members of the Statistics department for such a wonderful program.

Lastly, I would like to thank my family for all their love and encouragement. For my parents who raised me and supported me in all my pursuits. And most of all for my loving, supportive, and patient wife Nan.

Chapter 1

INTRODUCTION

1.1 Motivation

The main goal of my dissertation is to study biological questions using statistical methods, such as likelihood methods based on expectation-maximization (EM) and the mixture model. The biological questions of the most interest are mutation and recombination. These two mechanisms are the main sources of short term genetic variability in a simple population. Other evolutionary forces such as natural selection, gene flow and genetic drift interact with one another and alter genetic variability to shape evolution.

One direct benefit from studying these biological questions is to understand evolution better. By far, coalescence theory is the best known working theory in evolution studies. It provides a retrospective model of evolution and infers the most recent common ancestors (MRCA) of given population samples. In a broad view, statistical methods and tools are widely used in molecular evolution and facilitate advances in phylogeny, testing for selection and studying fossil records. On the other hand, new methods with extensive statistical tools have also been proposed to study evolution (Chen, 2003; Chen and Lindsay, 2006). Such methods offer new ways to study evolution and likely reply to some unanswered questions. In general, the development of new techniques allows us to get a better picture of evolution

history as well as the relationship between present individuals.

It is important that a better understanding of short term evolution leads to the improvement of disease mapping methods. Coalescence theory and its extensions have been used to find disease loci (Morris, et. al 2002, Browning 2006). New methods incorporating the mixture model and density estimation may offer powerful new analytical tools to add to coalescence theory. Additionally, the new methods may be useful to haplotype inference, which is a key question in statistical genetics.

Our focus here is on methods that are applicable to data sequences with long lengths, but are still relatively quick to compute. We will also consider whether methods with higher statistical accuracy but more computational expense are reasonable competitors.

1.1.1 Basic concepts and terminologies in genetics

We should introduce some basic concepts in genetics in order to make the materials in this dissertation more understandable and readable to general audience.

- DNA is short for deoxyribonucleic acid, which retains most of the genetic information of an organism. A DNA molecule is made of two strings of simple units called nucleotides, whose common types are labeled A, T, G and C. In cells, DNA molecules are organized into chromosomes. We often refer to DNA molecules as DNA sequences. Techniques in molecular biology such as sequencing, and DNA microarrays are often used to extract the information about DNA sequences.

- Most animals are diploid, meaning they have two homologous copies of each unique chromosome. One of the copies is from the father and the other copy is from the mother. What actually happens is that during a process called meiosis, a diploid organism produces cells with only one copy of the unique chromosomes, called gametes (sperm or ova). One gamete then fuses with another gamete to reproduce sexually.

- The diploid structure creates a so-called haplotype-genotype problem. Most of the

available techniques in molecular biology usually extract the combined information from both homologous copies simultaneously. In this thesis, we call the combined information, the **genotype**. It is also important to know the haploid information, usually known as the **haplotype** because the analysis of haplotypes, e.g., the International HapMap Project, is a major part of human disease studies.

- Recovering the haplotypes from the genotype is complicated by mutation and recombination.

- Mutation in biology is referred to as the change to the DNA material of an organism. It occurs in many forms and due to various reasons, both internal ones and external ones.

- Recombination is the exchange of DNA sequences between paired chromosomes. The purpose of recombination is to align the homologous chromosomes for cell division in meiosis. Recombination also comes in many forms; it is mostly often referred to as chromosomal crossover or crossover. The recombination rate is known to be highly heterogeneous in the human genome.

1.1.2 Biological questions

Biological processes such as recombination and mutation have intrigued researchers for decades. While the physical nature of these processes are constantly investigated, they are also studied statistically and analytically to uncover how they impact life forms and fuel evolution. The research fields include gene mapping, phylogenetic analysis, haplotype inference and others.

Mutation occurs at a fairly low rate, for instance about 10^{-8} mutations per base pair per generation for eukaryotes like human (Roach, et al. 2010). But given the massive size of human genome, the number of mutations for each generation is substantial. There is also

heterogeneity of the mutation rate in different genome regions. Some regions are prone to mutation while others are more stable.

The most common genetic data is called single nucleotide polymorphism (SNP) data. A SNP is a site with single base variation in coding between individuals on a DNA sequence. These sites are very abundant in the human genome, estimated at about six million. There are usually two letter-code variants for a SNP site, called alleles. One might code one allele as 1 and the other one 0. The sequences of SNP data then can be coded as binary sequences. The formal definition of SNP is based on a threshold for the minor allele frequency, which may be set as .05, .01 or .005 (NHGRI).

Estimating the recombination rate is far more complicated. After the genetic map (the order of genes or DNA sequences on a chromosome) has been constructed, the recombination rates between neighboring sites can be estimated along with the physical map (the nucleotide sequences of chromosomes). The population recombination rate can be estimated from population genetic data (Kindahl 1994;Hey and Kliman 2002; Dib, et al. 1996; Kirsch et al. 2000). These studies are based on familial information. What we are interested in is population-based, and hence involves deeper ancestry. We call it *archaic* recombination. Like the mutation rate, the recombination rate between sites is heterogeneous; the regions with high recombination rates are called recombination hot-spots. Other factors affect the landscape of recombination, i.e., telomeres and centromeres.

1.1.3 Coalescence model and individual clustering

Coalescence theory, first formalized by Kingman (1982), is a retrospective model of neutral evolution. The original idea is to depict the relationship between observed samples by finding their most recent common ancestor (MRCA). In the simplest version, it does not assume recombination, natural selection or population structure. The more advanced models do offer relaxed assumptions (Kaplan, et al. 1988; Neuhauser and Krone, 1997; Mohle and Sagitov, 2001; Morris, et al. 2002; Hellenthal and Stephens, 2006). The result of a coalescence model

can be represented by a phylogenetic tree. The relationship of the individuals can be viewed as the ways they shared MRCA. Coalescence models also provide the estimates of coalescence time and other terms. One of the unrealized goals of this thesis is to replicate the clustering effect of MRCA in which individuals (individual sequences) descend from the same ancestors are clustered together.

The relationship between individuals has been studied in many other ways. One can use distance based methods such as principal coordinate analysis (PCoA) (Gower, 1966), principal component analysis (PCA) (Pearson, 1901) or multidimensional scaling (MDS) (Torgerson, 1952) . If a certain genetic model can be defined as clustering rules or be treated as probabilistic models, one can also use model-based methods. One good example is the software STRUCTURE (Pritchard et al., 2000). The software is based on a model which assigns each individual to one of a mixture of groups; one assumes Hardy-Weinberg equilibrium (HWE) within each group. The membership of each individual may be modeled as an hidden binary (0 or 1) or fractional [0-1] variable. Then the HWE assumption can be used to construct a likelihood function. The EM algorithm to maximize this likelihood starts with a random choice of individual membership; the membership assignments are then updated in each iteration. STRUCTURE is often used to cluster individuals, find individual affinity and conduct gene mapping analysis when the phenotypic data is provided.

1.1.4 Applications in biomedical research

Evolution shapes humans and our biological environments; that includes the diseases we carry. There is obvious social impact from locating the genes which cause a certain disease. Research on these questions is known as gene mapping. Two common research topics are association studies and linkage studies. While the genetic factors for diseases with high inheritance have been easily uncovered, methods with more statistical power are needed to look for the genetic cause of common diseases with low heritability. Statistical clustering based methods can improve the available mapping methods and produce useful results. On

the other hand, coalescence models as well as metrics used in evolution are already recognized as ways to improve the statistical powers of gene mapping studies. One can easily see the benefits of studying evolution.

A particular kind of population has captured a lot of attention in gene mapping studies recently. It is known as the admixed population, such as found in the United States in African American and Latino subpopulations. The admixed populations are the descendants of the mix of two or more distinct groups (parental populations). It is assumed that the disease-causing variants of a gene originated from just one of the parental populations. In the admixed populations, gene regions across genome should be characterized by the origins from the parental populations. Furthermore, the gene with the disease-causing variants should manifests the characteristics of its origin from the parental populations or be linked to a region that manifest such characteristics. One of the characteristics often used is the ancestral state or the admixture proportion of the site. Thus, one can conduct gene-mapping studies by looking at these characteristics.

1.2 Statistical aspects of clustering in genetics

Suppose one has SNP data for some samples. If the haplotype sequences of the samples are known, one can recode the data as binary sequences. The frequencies of different sequences can be counted and aligned in the plot such as Figure 1.1b. The methods described in the previous sections have been used to cluster individual haplotype sequences and construct tree-like phylogeny/structures.

1.2.1 Kernel density estimation and mutation kernel

The idea of smoothing has a long history in statistics. The data histogram in Figure 1 is one version of smoothing. To summarize the systemic features of the binary data described in the previous subsection, one may try to smooth it. In addition to a histogram, the smoothing

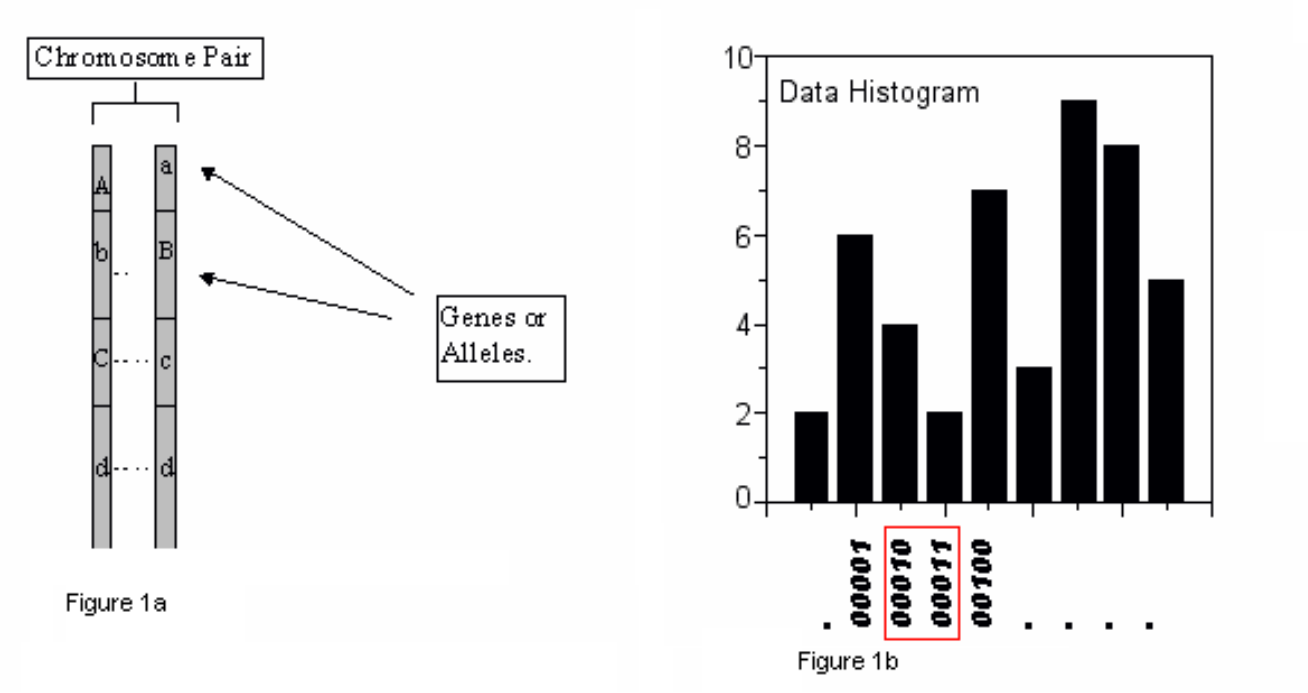


Figure 1.1: binary sequences and smoothing technique

techniques include but are not limited to local averaging, kernel smoothing (kernel estimator) and Fourier series.

A kernel density estimator is often defined as following. If x_1, x_2, ... x_n is an independent and identically-distributed sample of a random variable, then the kernel density approximation to the underlying probability density function is

$$f_h(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right), \tag{1.1}$$

where K is some kernel and h is a smoothing parameter called the bandwidth. This definition is based on continuous data; we will need to modify it for binary sequence data.

One of the key elements in the smoothing technique is the kernel function. For continuous data, there are many choices of a kernel function such as the normal kernel. In the case of binary sequences, there is an alternative called the mutation kernel (Chen and Lindsay,

2006). A binary sequence of length L is

$$x = (x(1), ..., x(L))$$

where each x_i is 0 or 1. The mutation kernel for two binary sequences x and y is defined as

$$M_p(x, y) = \prod_s m_p(x(s), y(s)) = \prod_s p^{|x(s)-y(s)|}(1-p)^{1-|x(s)-y(s)|} = (\frac{p}{1-p})^{\sum |x(s)-y(s)|}(1-p)^L \tag{1.2}$$

Here $x(s)$ and $y(s)$ are the states (0 or 1) for sequences x and y at site s; p is called the mutation rate. One can vary p to create a bandwidth parameter. One can also transform p to $\tau = -\log(1-2p)$. The mutation kernel can be deemed as the Markov kernel for a mutation model. In this model, τ corresponds to the total time of evolution.

Suppose x_1, x_2, ...x_n are the observed binary sequences. The density estimator of binary sequence, ϕ based on the mutation kernel (3.1) is

$$f_k(\phi) = \frac{1}{n}\sum_{i=1}^{n} M_p(\phi, x_i). \tag{1.3}$$

As we will show in Chapter 2, one may use (1.3) to obtain a phylogenetic tree-like structure by conducting the data smoothing sequentially in increasing τ and clustering the points by the modes of the density. As the smoothing increases, the number of modes decreases, creating a complete tree structure. For other statistical applications, one could also choose a good tuning parameter to obtain a density estimator for inference.

1.2.2 Mixture model and clustering by mode

The mixture model arises when the population has distinct population subgroups, each being some fraction of the total population. The data becomes a mixture when the subgroup identities are not known, or are "hidden". One can see examples of mixture models everywhere. One interesting feature of haplotype data is that it presents many cases of hidden variable

problems. For example, recombination breaks paired parental strings and allows DNA material exchange. In that process, determining which part of a binary sequence comes from the father or the mother is a hidden variable problem. Secondly, determining the two haplotypes of a genotype is a question of hidden information. Let us define a population of binary sequences, which act as the pool of binary sequences. The haplotype distribution comes from sampling in this population. Then the genotypes may be viewed as the mixture of two binary sequences from this sequence pool.

Last, but not the least, a sample of individuals can often be considered as a mixture of individuals from different populations.

1.2.3 Expectation-Maximization and initial starting values

Given a likelihood function $L(\theta)$, with parameter θ, the Expectation-Maximization algorithm (Dempster, Laird and Rubin 1977) is used to find the value of the parameter θ that maximizes $L(\theta)$ when some required data is hidden. This value maximizing the likelihood function is called the maximum likelihood estimator. The EM algorithm was formalized by Dempster, et al (1977). It consists of two steps, the expectation (E) step and the maximization (M) step. It is especially useful in situations that are known as missing hidden data problems. For example haplotype inference is a hidden data problem. The observed data are the genotypes and the hidden data are the two haplotypes for each genotype. A probabilistic model for the haplotype inference is needed.

Given a model, the likelihood EM (LEM) takes a starting density and updates it to a one step estimator of the haplotype distribution. The modal EM (MEM) can be used to find the modes of the density function estimated by LEM. The use of both EM methods in the same problem means we must take care to distinguish which results stem from maximizing the likelihood (LEM) and which stem from maximizing a density function (MEM).

1.2.3.1 Starting density

To initiate the LEM algorithm, we need a initial guess of the density, a starting density. Our interest in genetics problems will also lead us to consider the so-called independent starting density. Creating density estimators from LEM steps was proposed in Chung and Lindsay (2009). The diffusion kernels here includes the normal kernel and the mutation kernel. The logic for attacking the problem this way is as follows: We can think of the starting value as representing our initial ignorance, either as the null hypothesis or as a prior. In the subsection, we will show how one LEM step updates the starting density to a new density with higher likelihood.

1.2.3.2 Continuous likelihood EM steps: LEM1

It can be shown in the **continuous case** that the likelihood EM works (LEM) as follows. Given initial continuous density $\pi(\phi)$, the latent density, the mixture model density is

$$f(x;\pi) = \int k(x;\phi)\pi(\phi)\ du(\phi). \tag{1.4}$$

Here $du(\phi)$ stands for the uniform measure on the $\phi-$space, eg du for the normal kernel, and counting measure for discrete densities. Given a random sample x_1,...,x_n from (A.2), with unknown model parameter π, the likelihood function becomes

$$L(\pi; X = x_1, x_2, ..., x_n) = \prod \int k(x_i;\phi)\pi(\phi)\ du(\phi). \tag{1.5}$$

The idea of LEM estimation is that one can take a current estimator $\pi_{old}(\phi)$ and update it to a new estimator $\pi_{new}(\phi)$ that has higher likelihood using the EM algorithm. One LEM step from a current estimator π_{old} is obtained from the update step as:

$$\pi_{new}(\phi) = \pi_{old}(\phi)\Delta(\phi),$$

where

$$\Delta(\phi) = n^{-1} \sum \frac{k(x_i;\phi)}{f(x_i;\pi_{old})}.$$

We note here the relationship between $\Delta(\phi)$ and the mixture gradient $\Delta(\phi) - 1$. Under mild conditions, there is a nonparametric maximum likelihood estimator for model (A.2) (Vardi and Lee, 1993; Lindsay, 1995). The NPMLE estimator is a discrete distribution and the mixture gradient completely determines whether we have reached the solution. However, one appealing feature of the continuous approach is that if one could iterate to infinity with continuous π, ignoring computational questions, the solution would converge to the discrete NPMLE. It has been shown the covergence is weak (Chung, 2010). Practically speaking, it makes sense to use the continuous approach initially so as to get good starting values for the discrete approach (which are otherwise hard to find).

1.2.3.3 Modal EM (MEM)

The Modal EM (MEM) method is an approach to find the modes of a density estimator using the EM concept. Note that we use LEM when we apply the EM algorithm to a likelihood. When applied to another objective function, like a density, we call it MEM.

The basic idea is this: given any density that can be expressed in the form

$$k(\phi) = \sum_a \prod_b K(\phi; a, b)$$

where a and b are arbitrary summation and multiplication indices, one can find a mode of the density nearest a chosen starting value ϕ_0 by iteratively maximizing until convergence

$$\phi_{t+1} = \arg\max_{\phi} \sum_a w_t(a) \sum_b \{\log K(\phi, a, b)\}$$

where the weights $w_t(a)$ are determined by the previous value of ϕ_t as follows:

$$w_t(a) = \frac{\prod_b K(\phi_t; a, b)}{\sum_a \prod_b K(\phi_t; c, b)}$$

(Li, Ray and Lindsay, 2007). This algorithm is very similar to an EM algorithm for a mixture model, except that in the latter one is maximizing a likelihood, not a density. To find all the modes, or to construct a modal tree, one needs to select the starting values judiciously. The details of MEM will be further explained in the next chapter.

1.3 Research in haplotype inference

As described in the previous subsection, one may treat the haplotype data as binary sequences codes as 0,1. In practice, one often observes genotype data, which is the combination of two binary sequences. Genotypes can be coded as 0, 1 and 2 corresponding to the sum of the two binary sequences (haplotypes). The two true haplotypes for sites with genotype of 0 or 2 are unambiguous while those with 1 are ambiguous in that we need differentiate haplotype 1 from haplotype 0. Thus, for a given genotype sequence the set of possible haplotype pairs has $2^{(m-1)}$ elements, where m is the number of ambiguous sites. Inferring haplotypes based on genotypes is a important task in statistical genetics and bioinformatics because genotype data is much cheaper to collect. There are many proposed methods of haplotype inference. For simplicity, I put them into two categories, EM based methods and the others. Most of the methods assume either that there is a distribution of haplotypes from which the inferred haplotypes are sampled or that the haplotypes of some individuals depends on those of others.

1.3.1 EM algorithm and other statistical methods

One of the early methods for haplotype inference is an EM method proposed by Excoffier and Slatkin (1995). Their definition differ from ours. They defined the genotypes not having known haplotype information as "phenotype" and define the genotypes with known pairing of haplotypes as "genotype". Obviously the second kind of data, "genotype", can provide us with population information of value for the "phenotype" data. We will use our definitions in the following.

The model assumes the observed haplotypes follow an unknown multinomial distribution. The parameters of interest are the haplotype frequencies, p_1, p_2, ...p_k,where k is the number of possible haplotypes. The likelihood function of the haplotype frequencies given the counts of the phenotypes is

$$L(p_1, p_2, ..., p_n) = a_1 \prod_j (\sum_i P(h_{ik}h_{il}))^{n_j}$$

where n_j as the count of the phenotype G_j, $\{h_{ik}h_{il}\}$ is the set, indexed by i, of haplotypes that form G_j, $P(h_k h_l) = (p_k)^2$ if $k = l$ and $P(h_{ik}h_{il}) = 2p_k p_l$ if $k \neq l$. The algorithm starts with the initial guess of the haplotype frequencies and uses them to calculate the genotype frequencies (expectation step). Then the expected genotype frequencies can be used to estimate the haplotype frequencies (maximization step). One can see the method does not directly identify the haplotype pairs for genotypes. However, the haplotypes of the individual samples are inferred from the results of haplotype frequencies.

This method is the first elaborated haplotype inference (HI) method based on likelihood function. It also can be viewed as a Bayesian method (maximum posterior density) whose prior is uniform. In the case of hundreds of sites in the data, the method will have trouble finding the global maximum because there are multiple disconnected peaks on the likelihood surface. It is suggested the algorithm be run with different starting points to reveal the existence of multiple peaks and to possibly find a global maximum. Such implementation may improve the reliability but certainly increases computational cost.

In the past few years, researchers have explored and presented Bayesian methods with different priors such as the Dirichlet distribution (Niu, et al. 2002, 2005; Xing, et al. 2004) or the distribution derived from the infinite sites assumption (Stephens and Donelly, 2003). The later methodology is known as PHASE. Most of the Bayesian methods also incorporate the Gibbs sampler in the calculation. More recently, a few fast and flexible HI methods have been proposed (Scheet and Stephens, 2006; Kimmel and Shamir, 2005). The fastPHASE method (Scheet and Stephens, 2006) uses maximum likelihood with a hidden Markov-chain (HMM) to estimate the haplotype clusters as well as the locations of switch between clusters for every individual in the samples. The haplotype clusters are the states in a hidden Markov Chain. The shorter subsequenes of the data are analyzed by PHASE method. The new program is much faster with slightly less accurate results compared with the existing methods.

1.3.2 Other methods of Haplotype inference

Another set of HI methods may be called combinatorial methods (Gusfield and Orzack, 2005). One class of methods is rule-based. One starts with the genotype data and proposes an objective function for optimization. Here a function means a rule or a feature of the data that needs to be optimized. For instance, Clark (1990) proposed the following inference rule: start with the sequences with zero or one ambiguous site and infers the haplotypes for them. This inferred haplotypes are called the initial resolved haplotypes. Then, one may resolve another genotype by choosing one of the initial haplotypes and inferring the opposite haplotype given the genotype sequence. Then this newly inferred haplotype is added to the resolved category if it is not already included. Continue in this fashion to create a complete set of resolved haplotypes.

Clark's method may produce different solutions depending on the ordering of the sequences (Clark, 1990). Therefore, one should reorder the entries many times and solve each reordered dataset before the *best* solution is reported. It is computationally expensive and it doesn't guarantee a globally best solution unless all the possible orderings are analyzed. Of

course, finding the "best" solution stated above also involves a rule or a objective function.

There are two other approaches worth mentioning. One is called Pure Parsimony Haplotype. The idea is to find a solution of inferred haplotypes that minimizes the total number of distinct haplotype sequences. The other is called the Perfect Phylogeny Haplotype (PPH) problem. The term "Perfect Phylogeny" means the tree structure of a set of binary sequences rooted by a single binary sequence fits a coalescent model. The computation of the Clark method, Pure Parsimony, PPH and other methods relies more on the algorithms stemmed from the computer science field than on those from statistics. Studies have shown that recombination affects the performance of the methods. It is also suggested that for data with a low recombination rate, the Pure Parsimony model yields results as accurate as PHASE does. Note that PHASE was regarded one of the best haplotype inference methods in terms of accuracy (Scheet and Stephens, 2006; Kimmel and Shamir, 2005). On the other hand, the original idea of PPH comes from the assumption of coalescence without recombination.

GERBIL (Kimmel and Shamir, 2005) partitions long sequences into smaller subsequences and infer haplotype in the subsequences. Recall that an important step of the fastPHASE algorithm is to infer haplotypes locally, which is a partition strategy to some extent. Here we think the partition of long sequences has dual roles. First, the computation for the haplotype inference problem is easier on shorter subsequences. More importantly, the partition strategy in GERBIL and the local clustering of haplotypes in fastPHASE takes recombination into account. Thus we will also propose a new partition strategy based on density estimation.

1.4 Recombination

Recombination is an important element in evolution. Researchers either study it directly to compute fine-scale recombination rates or include it in a model to assess population structure, infer phylogeny and conduct gene mapping. The data used in the first case is DNA sequence alignments (tabulated haplotype sequences). Most of the ideas in this field

rely on the combination of phylogenetic tree structure and a probability transition matrix for the recombination rate between sites or a hidden Markov model (HMM) for the dependence among the sites.

It is natural to introduce the hidden Markov Model. A HMM is a statistical model involving hidden states and observed states. The word *hidden* is interchangeable with *latent*. Both mean that the observed variables arise in a simple way from an underlying random process that is "hidden" from observations or "unobserved". A good example is the haplotype inference problem where the genotype is observed while the haplotypes are hidden. In HMM, the observed states only depend on the hidden states and the hidden states are modeled as a Markov process. A few new approaches have been proposed such as the multiple change-point model (MCP) of Suchard et al. (2003), the dual multiple change-point model (DMCP) of Minin et al. (2005), and the phylogenetic factorial hidden Markov model (PFHMM) of Husmeier (2005).

1.5 Other statistical approaches

The complexity of the modeling and the computation in the biological problems compels us to seek new methods. It may be a method leading to easy implementation, an estimator that is unbiased and/or with less variance, an algorithm speeding up the computation or a method that provides improved statistical inference. Three methods are discussed in this section, composite likelihood, Monte Carlo likelihood and dynamic programming.

1.5.1 Composite likelihood

The first composite likelihood method was proposed by Besag (1975) who called it pseudo-likelihood. The beauty of the idea is to construct a likelihood-like objective function with less computational complexity (than the full likelihood function) that one can use when the full likelihood function or Bayesian methods are computationally intractable. As defined by

Lindsay (1988), a composite likelihood is any product of marginal or conditional likelihoods constructed so as to have simpler computation than the full likelihood. This approach has been applied to spatial data (Stein, Chi and Welty ,2004), sparse data (Liang and Yu, 2003) and genetics. In the last case, composite likelihood methods have been used to improve the studies of evolution (Zhu and Bustamante, 2005; Moore and Stevens, 2008; Wright, et al., 2008) and gene mapping (Devlin, et al., 1996; Andersen, 2004). In the case of haplotype inference, one could construct the marginal densities for adjacent sites to build a composite likelihood function.

1.5.2 Monte Carlo likelihood and Gibbs sampling

Monte Carlo likelihood (Geyer and Thompson, 1992; Thompson, 1994) can also provide computational advantages when the exact solution to a likelihood function or a Bayesian problem is infeasible. It is tightly related to Monte Carlo Markov Chain (MCMC). The Monte Carlo likelihood uses MCMC to estimate likelihood ratios and provide expectation of parameters. It provides a competitor to composite likelihood methods although one might hope for a synergistic combination of the two.

One realization of MCMC is through the Gibbs sampler (Geman et al. 1984; Tanner and Wong, 1987; Gelfand and Smith, 1990). Suppose one have a set of parameters $\Theta = \theta_1, ..., \theta_n$ and the marginal distribution of $\theta_1|(\theta_2, ..., \theta_n); ....; \theta_n|(\theta_1, ..., \theta_{n-1})$ are known. Then one may set some initial value for Θ and update $\theta_1, ..., \theta_n$ one by one using the marginal distribution with the most recently updated values of $\theta_1, ..., \theta_n$.

- Start with $\Theta^0 = \theta_1^0, ..., \theta_n^0$
- For Θ^1:

  Generate θ_1^1 from $\pi(\theta_1^0|(\theta_2^0, ..., \theta_n^0)$;

  Generate θ_2^1 from $\pi(\theta_2^0|(\theta_1^1, ..., \theta_n^0)$;

  ...

Generate θ_n^1 from $\pi(\theta_n^0|(\theta_1^1, ..., \theta_{n-1}^1)$;

This sequence results in one update of the Markov Chain. Now repeat to generate $\Theta^2, ..., \Theta^M$

1.5.3 Dynamic Programming

Dynamic programming (Bellman, 1957) is often in mathematics and computer science to help find solutions to complex problems by breaking a large problem into small subproblems. It has been widely used in bioinformatics, computational genetics such as local sequence alignment (Smith-Waterman algorithm, Smith and Waterman, 1981). The basics of a dynamic problem often involves identifying subproblems, solving subproblems, storing the solutions and the calculations for subproblems and then using them to solve a larger overlapping problem. In the example of the Smith-Waterman algorithm for local sequence alignment between two sequences, one starts by specifying a scoring system for similarity between the elements on the sequences. One then compares subsequences of all possible length and optimizes the overall similarity measure between the two sequences. Here the subproblems are the alignment on the subsequences and the larger overlapping problem emerges when one compares the solutions from different partitions of the same sequence.

Chapter 2

Clustering binary sequences, a preface

In this chapter I present a new modal method for clustering binary sequences and extend it to a genotype model. Our method is based on assigning observations to the modes of a nonparametric density estimator. The input for this method will be a set of n binary sequences $x_1, ..., x_n$. In the DNA context we can think of this as haplotype data. The output will 1) provide a tree-like structure for clustering the sequences; 2) provide starting value for nonparametric maximum likelihood method; and 3) can be used to compare two sets of data.

Clustering of binary sequences using tree-like structures has been extensively studied in molecular evolution. Many algorithms and tools have been developed and are often referred to as phylogenetic methods. There are two main categories of methods, distance based methods and sequence-based methods.

The distance based methods such as neighbor joining (NJ), minimum evolution (ME) and the least squares (LS) method rely on distance matrices generated using some distance measure between sequences. The sequence-based methods are mainly model-based methods or rule-based methods like the maximum parsimony (MP) or the maximum likelihood (ML) method. However, the comparison of our new method with these methods may not be of much use since the phylogenetic methods are used to study evolution in macro scale, like

speciation, that involve millions of years of changes. The statistical confidence in the clusters are usually assessed by bootstrap in these methods. The kind of data of interest in this thesis is a population sample from a single species. The low diversity in binary sequences from a single species and a large number of sequences cause some difficulty for the phylogenetic methods; for example there are many interior branches with length zero (Nei and Kumar, 2000).

The work of clustering binary sequences in a purely statistical setting was pioneered by Chen and Lindsay (Chen, 2003 and Chen and Lindsay, 2006). This chapter serves as a stepping stone for the later chapters in which we formulate the models for genotype-haplotype and recombination.

2.1 Clustering individuals by MEM

Binary sequences can be viewed as high dimensional data where the number of dimensions is determined by the number of ambiguous sites. An illustration of a three dimension dataset is presented. Each axis represents a binary site and there are two possible values for each site, 0 and 1. In the three-dimension case, the data mass may be at the vertexes of the cube as the empty dots (Figure 1). The observed sequence 101 is the filled dot at $(1, 0, 1)$.

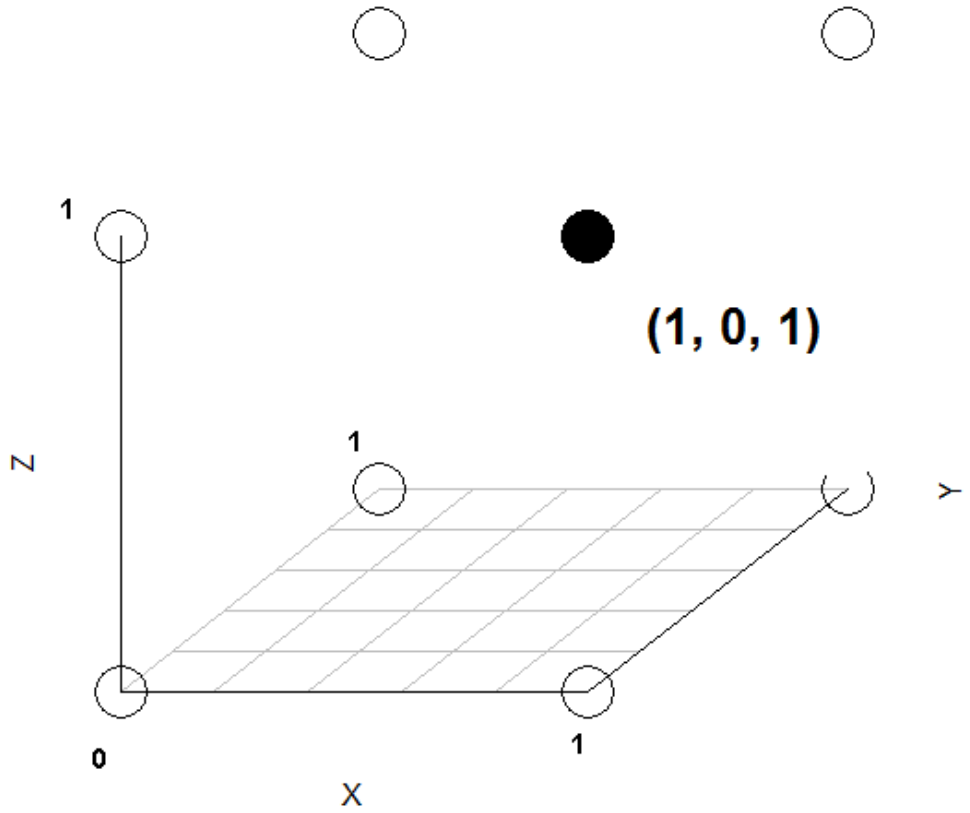


Figure 2.1: A illustration of clustering binary sequence

The modes of binary sequence data will be the sequences with the (locally) highest frequency in a kernel density estimate. For instant, a dataset consisting of only one sequence will have a single mode, which is the sequence. However, datasets with many different sequences could have multiple modes in the density estimator, where the location of each mode is determined by the data as well as the bandwidth τ in the kernel function. In this thesis we will construct kernel density estimators based on the data and use the modes of these densities to identify data structure. We use the mutation kernel (Chen and Lindsay, 2006) as the basic element for the density function estimator for our haplotype, genotype-haplotype and recombination models.

For a set of n binary sequences $\{x_1, ..., x_n\}$ and a fixed bandwidth τ, we define the modes of the density estimator to be $\zeta_1, ..., \zeta_k$. We will say data sequence x_i is assigned to the jth data cluster if the *nearest* mode to x_i is ζ_j. Here *nearest* will mean that an MEM algorithm

started at x_i and converges to ζ_j.

2.2 Updating haplotype density via Likelihood EM (LEM)

We consider a initial estimator for the haplotype density $\pi(\phi)$ to be

$$\gamma(\phi) = \pi_0(\phi) = \prod_s \gamma_s^{\phi(s)}(1-\gamma_s)^{1-\phi(s)}. \tag{2.1}$$

This density $\pi_0(\phi)$ is an independence density over sites, with the probability of 1 varying according to γ_s. The parameter γ_s can be set as the proportion of $1'$s at site s. Since we assume the 0-1 status on each site is independent, we also call this initial density, the $'$independent$'$ initial density. Using one step LEM update, one first gets the gradient function

$$\Delta(\phi) = n^{-1} \sum_i \frac{M(x_i, \phi)}{\sum_{\phi_1} M(x_i, \phi_1)\gamma(\phi_1)}$$

where

$$M(x_i, \phi) = \prod_s m(x_i(s), \phi(s)), \tag{2.2}$$

$$m_p(x, \phi) = p^{|x-\phi|}\overline{p}^{1-|x-\phi|}$$

and p is the mutation rate of the single nucleotide ($\overline{p} = 1-p$). Here we assume p is the same for every site.

Now the independence of γ's means we can break the sum in the denominator by site and get

$$\sum_{\phi_1} M(x_i, \phi_1)\gamma_s(\phi_1) = \prod_s \{m(x_i(s), 0)(1-\gamma_s) + m(x_i(s), 1)\gamma_s\} = \prod_s D_s(x_i(s)).$$

One can rewrite the gradient function as

$$\Delta(\phi) = n^{-1} \sum_i \prod_s \frac{m(x_i(s), \phi(s))}{D_s(x_i(s))}.$$

Therefore,

$$\begin{aligned} \pi_1(\phi) &= \gamma(\phi)\Delta(\phi) \\ &= n^{-1} \sum_i \prod_s \frac{m(x_i(s), \phi(s))}{D_s(x_i(s))} \gamma_s^{\phi(s)} (1-\gamma_s)^{1-\phi(s)} \\ &= n^{-1} \sum_i \prod_s h_s(x_i(s), \phi(s)) \end{aligned} \tag{2.3}$$

The new kernel function h_s defined implicitly in (2.3) is a updated version of the mutation kernel introduced in the first chapter. If one elects to use $\gamma_s = \frac{1}{2}$ for all s, which is the uniform starting density

$$\gamma(\phi) = \pi_0(\phi) = \prod_s \frac{1}{2}^{\phi(s)} \frac{1}{2}^{1-\phi(s)} = \prod_s \frac{1}{2}, \tag{2.4}$$

then the updated kernel function

$$D_s(x_i(s)) = m(x_i(s), 0)(1 - \frac{1}{2}) + m(x_i(s), 1)\frac{1}{2} = \frac{1}{2}$$

and

$$\begin{aligned} h_s(x_i(s), \phi(s)) &= \frac{m(x_i(s), \phi(s))}{D_s(x_i(s))} \gamma_s^{\phi(s)} (1-\gamma_s)^{1-\phi(s)} \\ &= \frac{m(x_i(s), \phi(s))}{\frac{1}{2}} \frac{1}{2} \\ &= m(x_i(s), \phi(s)) \end{aligned}$$

is the original mutation kernel (2.2). Thus the LEM1 step for the uniform starting leads to

$$\pi_1(\phi) = n^{-1} \sum_i \prod_s m(x_i(s), \phi(s)). \tag{2.5}$$

We expect the estimator in (2.3) will provide more meaningful results than (2.5) because the independent starting density (2.1) is more informative than the uniform starting density (2.4).

2.3 General form of MEM algorithms

In this section, I will describe how to find the modes of a kernel density estimator using the modal EM algorithm.

Recall the basic idea of MEM: given any density that can be expressed in the form

$$k(\phi) = \sum_a \prod_b K(\phi; a, b)$$

where a and b are arbitrary summation and multiplication indices, one can find the mode of the density nearest a chosen starting value ϕ_0 by iteratively maximizing until convergence

$$\phi_{t+1} = \arg\max_\phi \sum_a w_t(a) \sum_b \{\log K(\phi, a, b)\}$$

where the weights $w_t(a)$ are determined by the previous value of ϕ_t as follows:

$$w_t(a) = \frac{\prod_b K(\phi; a, b)}{\sum_a \prod_b K(\phi_t; a, b)}$$

(Li, Ray and Lindsay, 2007). We will later see the iterations of this algorithm are very easy to compute in our problem.

To find all the modes, or to construct a modal tree, one needs to select the starting values

judiciously. We will use each of the n data points x_i as starting values.

Reminder: The MEM algorithm just described is similar to a EM algorithm for a mixture model. We call the latter one LEM because it is maximizing a likelihood while the MEM maximizes a density.

2.4 MEM for binary sequences

Our kernel density estimator for haplotypes using the mutation kernel is

$$\hat{f}_\tau(x) = n^{-1} \sum_i \prod_s h_{s\tau}(x_i(s), x(s))$$

where $h_{s\tau}(x_i(s), x(s)) = \frac{m_\tau(x_i(s),x(s))}{D_s(x_i(s))} \gamma_s^{x(s)} (1-\gamma_s)^{1-x(s)}$; i and s are the sequence and site indices, respectively. The $\hat{f}_\tau$ has the general form of a kernel density estimator as in equation (1.1). The h_s function is the updated version of equation (1.2), $m_\tau(x_i(s), x(s))$. One can think of this as a mixture model, where we have n sub populations mixed together. We draw an observation $\underline{x}$ from population i with probability $\frac{1}{n}$. Conditionally, given the population index i, the observation $\underline{x}$ is a mutated version of x_i, based on independent mutations over sites.

The MEM algorithm is as follows. Given x_1, the first estimate of the mode ζ_1 that is nearest to x_1 is

$$\zeta_1^{<1>} = \arg\max_\zeta \sum_i w^1(x_i) \sum_s \log h_{s\tau}(x_i(s), \zeta_1(s) = x_1(s))$$

Since the sites are conditionally independent, we can update each site separately. The update for site s for mode ζ_1 simplifies to

$$\zeta_1^{<t>}(s) = \arg\max_\zeta \sum_i w^t(x_i) \log h_{s\tau}(\zeta_1(s), x_i(s))$$

where the weights for x_i are

$$w^t(x_i) = \frac{\prod_s h_{s\tau}(\zeta_1^{<t-1>}(s), x_i(s))}{\sum_j \prod_s h_{s\tau}(\zeta_1^{<t-1>}(s), x_j(s))}.$$

(Note: the denominator of $w^t(x_i)$ is not needed in the maximization as it is constant over i.)

This maximization step is done by comparing

$$\sum_i w^t(x_i) \log h_{s\tau}(\zeta(s), 0)$$

and

$$\sum_i w^t(x_i) \log h_{s\tau}(\zeta(s), 1).$$

Technically speaking, if $\sum_i w^t(x_i) \log h_{s\tau}(\zeta(s), 1) = \sum_i w^t(x_i) \log h_{s\tau}(\zeta(s), 0)$, both 1 and 0 are equally likely. This situation is very rare. If it occurs, our algorithm leaves the state of $\zeta_1^{<t>}(s)$ unchanged from its previous state $\zeta_1^{<t-1>}(s)$.

The general algorithm will be

1) Given current iterate value $\zeta_1^{<t-1>}$, start with the first site of the first sequence and calculate the weights $w^t(x_i)$, which depend on the initial value and the original data sequences;

2) For site s of mode $\zeta_1^{<t>}$, calculate the MEM objective function for $\zeta_1^{<t>}(s) = 1$ or 0

$$\sum_i w^t(x_i) \log h_{s\tau}(\zeta(s), 1) \tag{2.6}$$

and update $\zeta_1^{<t>}(s)$ to the states (0 or 1) depending on whether function (2.6) is $<$ or $>$ $\frac{1}{2}$.

3) move to the next site of the first sequence and repeat part 2)

4) Once every site of the first sequence is updated, then one may view the resulting updated sequence $\zeta_1^{<t>}$ as a step towards the mode ζ_1 from the data point x_1. One may update $\zeta_1^{<t>}$ using 2)-3) until no sites in $\zeta_1^{<t>}$ changes in moving to $\zeta_1^{<t+1>}$.

5) After finding the mode for x_1, one may search for the mode for the second sequence x_2 and repeat (1)-(4).

Note that since we are maximizing over a discrete space, the algorithm stops in finite time. No stopping rule is needed.

2.5 The choice of bandwidth

We here present an method of choosing bandwidth for density estimators based on degrees of freedom. We use quadratic distance (Lindsay, et. al., 2008) to calculate the degrees of freedom of kernel density functions. Statisticians often use degrees of freedom to assess model complexity. Hastie and Tibshirani discussed the degrees of freedom of a smoother in the context of nonparametric regression (1990). Extensive discussion about this matter can also be found in Cleveland and Devlin (1988). Extensive discussion about using degrees of freedom concept in variable selection can be found in Ye (1998). More recently, degrees of freedom was used to assess the number of nonzero coefficients in the lasso estimates (Efron, et al., 2004; Zou, Hastie and Tibshirani, 2007). Statistician also studied the statistical properties of degrees of freedom in other contexts such as support vector regression (Gunter and Zhu, 2007) and a model selection method using CAP penalties (Zhao, Rocha and Yu, 2006).

2.5.1 Quadratic distance and degrees of freedom (DOF)

We will introduce a method based on quadratic distance to choose a range for the bandwidth, τ. We will first introduce the theoretical background of quadratic distance (Lindsay et al. 2008) and then the degrees of freedom calculation. Then we will lay out a basic strategy for choosing τ by determining the degrees of freedom of the π_1 estimator.

2.5.1.1 Quadratic distance (background)

Let $\hat{F}$ be the empirical CDF ($eCDF$) with mass $\frac{1}{n}$ at each x_i. Let G be the true CDF. Let $K(x, y)$ be a symmetric nonnegative definite kernel function. Then let

$$\begin{aligned} d(\hat{F}, G) &= \int\int K(x,y)(d\hat{F}(x), dG(x))(d\hat{F}(y), dG(y)) \\ &= K(\hat{F},\hat{F}) - K(\hat{F},G) - K(G,\hat{F}) + K(G,G) \end{aligned}$$

where $K(A,B) = \int\int K(x,y)dA(x)dB(y)$.

This is the empirical distance between $\hat{F}$ and G. And it is nonnegative valued. In the setting of testing the goodness of fit of models, one could think of $d(\hat{F}, G)$ as a test statistic for $\{H_0 : \text{true CDF} = G\}$. More details about the asymptotics of DOF are given in Lindsay, et. al. (2008)

In the case of kernel functions, $DOF(K)$ is defined to be

$$\frac{tr_G(K)^2}{tr_G K^2} \tag{2.7}$$

where $tr_G(K) = \int K(x,x)dG(x)$ and $tr_G(K^2) = \int\int K^2(x,y)dG(x)dG(y)$.

2.5.1.2 The use of DOF

We will first construct an appropriate $K(\underline{x}, \underline{y})$ to represent the distance of our estimator from the true value.

We have our kernel estimator as

$$\hat{\pi}(\phi) = \frac{1}{n}\sum_i k(\underline{x}_i, \phi)$$

where $H_{s\tau}(\underline{x}, \phi) = \prod_s h_{s\tau}(x(s); \phi)$ (equation 2.3) can be such *kernel* function.

The $\hat{\pi}$ is an unbiased estimator of

$$\pi^*(\phi) = E_x k(\underline{x}_i, \phi).$$

We consider the quadratic distance for density estimator π:

$$d(\hat{\pi}, \pi^*) = \sum_{\phi} (\hat{\pi}(\phi) - \pi^*(\phi))^2$$

Now reverse the orders of integration in the $d(\hat{\pi}, \pi^*)$ by integrating over ϕ first, we will get

$$d(\hat{\pi}, \pi^*) = \int \int K(\underline{x}, \underline{y}) d(\hat{F} - G)(x) d(\hat{F} - G)(y)$$

where the kernel K is defined below.

In the discrete case, the kernel in the DOF calculation of a density estimator, $n^{-1} \sum_i k_\tau(\underline{x}_i, \phi)$ with k_τ being the kernel of the density, is

$$K(\underline{x}, \underline{y}) = \sum_{\phi} k_\tau(\underline{x}, \phi) k_\tau(\underline{y}, \phi)$$

Thus for haplotype density, π_1 in (2.3) the kernel is

$$K(\underline{x}, \underline{y}) = \sum_{\phi} \prod h_\tau(x(s), \phi(s)) \prod h_\tau(y(s), \phi(s)) = \prod_s r_\tau(x(s), y(s))$$

where $r_\tau(x(s), y(s)) = h_\tau(x(s), 1) h_\tau(y(s), 1) + h_\tau(x(s), 0) h_\tau(y(s), 0)$. The r_τ kernel takes on four values that can be tabled.

To assess the degree of smoothing, τ, we can estimate $DOF(G)$ by

$$\widehat{DOF}(G) = DOF(\hat{F})$$

The numerator of 2.7 is

$$\begin{aligned} tr_G(K) &= \int K(\underline{x},\underline{x})dG(\underline{x}) \\ \text{which is estimated by} \quad G &= \hat{F} \\ &= \frac{1}{n}\sum_i K_\tau(\underline{x}_i,\underline{x}_i) \end{aligned}$$

In the denominator,

$$K_\tau^2(\underline{x},\underline{y}) = \prod_s r_\tau^2(x(s),y(s))$$

and

$$tr_G(K^2) = \int\int K^2(\underline{x},\underline{y})dG(\underline{x})dG(\underline{y})$$

and using $G = \hat{F}$ gives

$$= \frac{1}{n^2}\sum_i\sum_j K_\tau(\underline{x}_i,\underline{x}_j)$$

Here

$$DOF(\hat{F}) = \frac{[\sum K_\tau(\underline{x}_i,\underline{x}_i)]^2}{\sum\sum K_\tau^2(\underline{x}_i,\underline{x}_j)}. \tag{2.8}$$

This function can be quickly calculated over different ranges of τ in order to find a reasonable range. In higher dimensions, τ must be larger to attain the same DOF.

Note that the DOF presented here is uncentered. We did not present and pursue the centered version of DOF, $CDOF$ because its calculation is unstable.

2.6 Results

We use a dataset of 18 sequences of West African samples (Yoruba) and 34 sequences of European American samples from the International HapMap Project (2003). The length of the sequences is 22.

2.6.1 DOF calculation

The DOF calculation for this set is summarized in Table 2.1. Note that the number of modes that have been inferred are correlated with the DOF, although they are not equal.

p, the mutation rate	DOF	exact number of nodes/modes by MEM
.1	8.05	11
.12	3.71	10
.16	2.11	8
.18	1.54	7
.255	1.5	4
.26	1.44	3
.27	1.39	1

Table 2.1: DOF calculation for 52 sequences of length 22

2.6.2 Tree structure

The modes $x_i \to \zeta_i$ found by the MEM method are themselves binary sequences. Some of the modes may be identical. The sequences sharing the same mode $\zeta_i = \zeta_j$ are clustered together. They might be considered to share a common ancestor, ζ_i. Furthermore, one may use the modes ζ_i from a small bandwidth p as the input data for MEM with a larger bandwidth. To do so in such a hierarchical fashion, one may collapse a large number of sequences sequentially until one sequence remains. And the resulting clustering and branching of data sequences and the modes forms a tree-like structure.

I analyzed a dataset of 52 binary sequences with 22 sites using the MEM (Figure 2.2). The tree structure showed there were two clusters in the samples. Within the larger cluster,

there was a subcluster entirely of European American haplotypes except YRI2. In the small cluster, some YRI (Yoruba) samples also formed a subcluster.

2.7 Summary and discussion

In this chapter we have shown the basics of using likelihood and density estimation to cluster haplotypes. The rest of the dissertation uses a similar recipe.

2.7.1 Applications of MEM

We consider the practical uses of the modal clustering method. The tree structures for the same samples using different parts of the genome may be compared to produce a consensus tree (Adams, 1972 and Carpenter, 1988). With the consensus tree, One was able to study the variation of the MEM method as a tree structure estimator (Chen, 2003). On the other, one may use the tuning parameters as the estimator of mutation rate or time. Similarly one may compare the tree structures for different parts of the genome to provide insights about evolution.

2.7.2 Bootstrapping

Like many phylogenetic methods, the MEM method could use some form of bootstrap method (Felsenstein, 1985) to provide inference for the tuning parameters or to check consistency of the tree structure. Due to the low diversity in the data samples and the potential large number of sequences, the distances between the binary sequences are small and the exact tree structure can be easily affected by the inclusion/exclusion of a few sequences. So the objectives for bootstrap resampling in the MEM case may be different from those in the case of phylogenetic method. In the latter, one commonly bootstraps on the sites used. In our case, neighboring sites are dependent, and it may be wiser to bootstrap on the X's. One may focus more on the general trend in the tree structure such as topologies and major

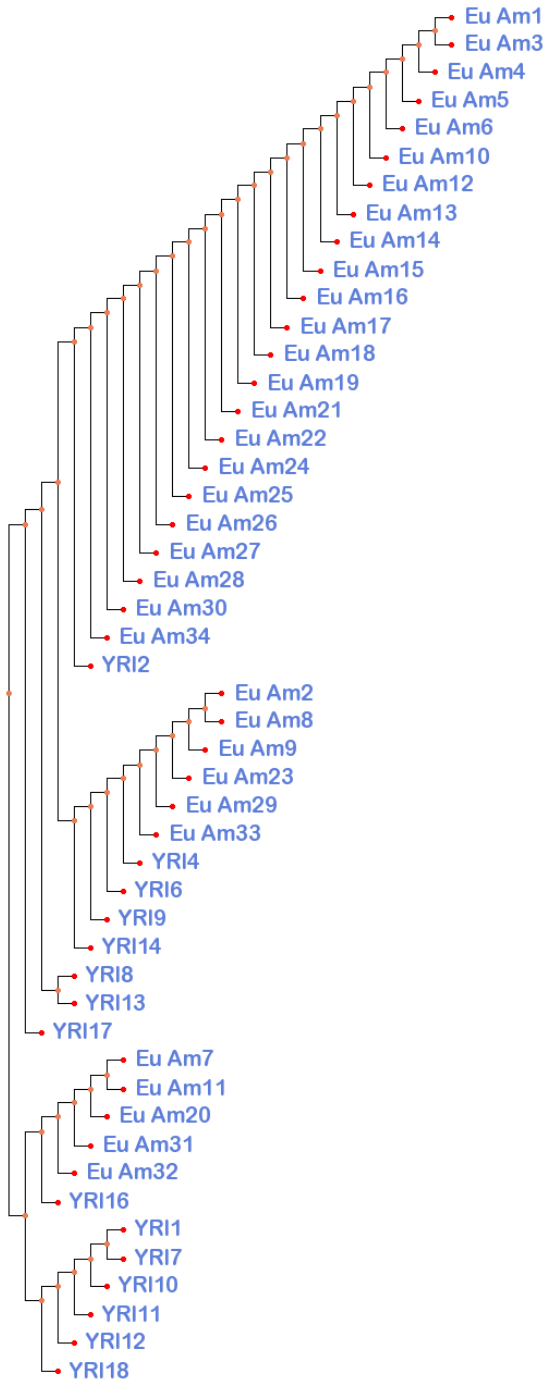


Figure 2.2: Tree structure using the MEM method. Tree is generated by Phyfi (Fredslund, 2006).

branches instead of on the branch lengths.

Chapter 3

A Modal based Haplotype Inference method for linked single nucleotide polymorphisms

Abstract

Motivation:

Haplotype inference as an important aspect of genetic studies has been well studied. There are many currently available methods for haplotype inference but most of them are based on parametric models. Nonparametric models rely on less stringent assumptions. For the haplotype inference problem, we will model the observed genotype data as a mixture of mutated ancestral haplotypes. A kernel density estimator for the haplotype distribution can then be constructed using some recent developments in nonparametric statistics.

Results:

An objective function is proposed for inferring the individual haplotypes within a genotype for datasets with linked single nucleotide polymorphisms. The objective function is the conditional probability of the haplotype pair given the observed genotype, as calculated using the kernel density estimator. The underlying model allows all the sites to be dependent.

One uses the density estimator for the distribution of haplotypes to find the most probable haplotype pairs for any given genotype. There are multiple modes in the haplotype objective function, so a strategy for using multiple starting values is developed. Analysis on both simulation data and real data shows the performance of the new method equals PHASE on short sequences.

3.1 Introduction

One of the early methods for inferring haplotypes from genotype data is an EM method proposed by Excoffier and Slatkin (1995). The model assumes the observed haplotypes follow an unknown multinomial distribution. The parameters of interest are the haplotype frequencies, p_1, p_2, ...p_k, where k is the number of possible haplotypes. The EM algorithm starts with an initial guess of the haplotype frequencies and the expectation (E) step uses them to calculate the genotype frequencies. Then the expected genotype frequencies can be used to estimate the haplotype frequencies (in the maximization step). The method has trouble finding the global maximum if there are multiple disconnected peaks on the likelihood surface. The authors recommended that the algorithm be run with different starting points to reveal the existence of multiple peaks and to possibly find a global maximum. Such implementation may improve the reliability but certainly increases computational cost.

In the past few years, researchers have presented Bayesian methods for haplotype inference with different priors such as the Dirichlet distribution (Niu, et al. 2002, 2005; Xing, et al. 2004) or the distribution derived from the infinite sites assumption (Stephens and Donelly, 2003). The later methodology is known as PHASE. PHASE is one of the best haplotype inference methods in terms of accuracy (Scheet and Stephens, 2006; Kimmel and Shamir, 2005).

There are also many nonstatistical methods for haplotype inference, such as Clark's method (1990), the Pure Parsimony Haplotype model (Gusfield, 2003) and the Perfect Phy-

logeny Haplotype model (Gusfield, 2002). Some of these methods fall into the realm of computer science. Reviews on the nonstatistical methods for haplotype inference can be found in Bonizzoni, et al. (2003) and Gusfield and Orzack (2005).

Recombination and the decay of linkage disequilibrium in long genotype sequences are among the main obstacles to inferring haplotypes in large datasets. The fastPHASE program (Scheet and Stephens, 2006) and GERBIL (Kimmel and Shamir, 2005) along with earlier methods (Niu et al., 2002; Stephens and Donnelly, 2003) were implemented with means to account for recombination and the decay of linkage disequilibrium.

In this paper, we present a genotype-to-haplotype inference method that is based on creating a density estimator for the ancestral haplotypes. We use this density to find the most likely pair of haplotypes that have generated the given genotype. The density estimator is derived by building a mixture model for the data, and then applying the EM to the model likelihood.

Kernel density estimation (KDE) is an important tool of nonparametric statistics. Nonparametric methods have been used previously in haplotype inference (Xing, et al. 2007) as well as in other topics of genetics studies, such as linkage studies (Koller, 1999), the association of multiple genes with human disease (Shaid, et al., 2005) and estimating the effects of quantitative trait loci (Fine, 2004). However, the application of KDE in genetics studies is mostly limited to nonparametric regression (Beaumont, Zhang and Balding, 2002) and neural networks for analyzing metabolic data (Holmes, et. al. 2001).

Finite mixture models, which can be expressed in different forms under different settings (Everitt and Hand, 1981; McLachlan and Basford, 1988; Lindsay, 1995), have deep roots in statistical genetics (Zeng, 1994; Churchill and Doerge, 1994; Bailey and Elkan, 1994; Pritchard, et. al. 2000) including some of the haplotype inference methods (Excoffier and Slatkin, 1995; Stephens et al., 2001; Stephens and Donnelly, 2003 and Scheet and Stephens, 2006). A mixture model has two key elements, the component densities and the weights associated with these components. Chen and Lindsay (2006) introduced a nonparametric

method for clustering binary sequences, in which an ancestral mixture model of binary sequences (haplotype sequences) was used. The ancestral mixture model used by Chen and Lindsay is a finite mixture model whose component densities were called mutation kernels.

We will use the same mixture model of haplotypes as Chen and Lindsay but will create additional assumptions to model the observed genotype data. Given the genotype model, a density estimator of ancestral haplotypes can be constructed from the likelihood using a likelihood expectation maximization (LEM) method. The statistical properties of the LEM method have been studied by Chung and Lindsay (2010). One can use the LEM method to construct an estimator for the conditional probability of a pair of haplotypes generating the observed genotype. This conditional density is then maximized to find the most likely haplotype pair for the genotype. We carry out the maximization using the modal expectation maximization (MEM) algorithm (Li, Ray and Lindsay, 2007). The highest found mode determines the estimated haplotype pair.

3.2 Methods

First, we will briefly describe the mutation kernel used by Chen and Lindsay (2006). The mutation kernel is defined on haplotype sequences. Then we will introduce a new kernel function for genotype sequences, the genotype mutation kernel, which is derived from the haplotype mutation kernel. We will construct a density estimator of the ancestral haplotype distribution using the genotype kernel. We will also derive the conditional probability that a particular haplotype pair generated the genotype sequence. We will then describe the Modal Expectation Maximization (MEM) algorithm that can find the maximizer of the conditional probability. Four strategies of choosing starting values for the MEM algorithm will also be described.

3.2.1 Mutation kernel

A binary sequence of length L is

$$x = (x(1), ..., x(L))$$

where each $x(s)$ is 0 or 1. The mutation kernel for two binary sequences x and y is defined as

$$M_p(x, y) = \prod_s m_p(x(s), y(s)) = \prod_s p^{|x(s)-y(s)|}(1-p)^{1-|x(s)-y(s)|} = (\frac{p}{1-p})^{\sum |x(s)-y(s)|}(1-p)^L \tag{3.1}$$

Here $x(s)$ and $y(s)$ are the states (0 or 1) for sequences x and y at site s and p is the single nucleotide mutation rate. We will use p as a tuning parameter to control the level of smoothness of the density estimators.

3.2.2 Overview of the genotype-haplotype model

We assume there exists an unknown distribution of ancestral haplotype sequences, μ, whose density is denoted by $\pi(\mu)$. The haplotype sequences are recorded as strings of binary variables, 0 or 1, of length L. We observe n genotype sequences, $x_1, ..., x_n$, each of length L, where at each site s, the genotype $x_i(s)$ is the sum of two latent haplotypes, $h_{i1}(s)$ and $\overline{h}_{i1}(s)$ and so takes on values 0, 1 or 2. We further assume that the haplotype sequences h_{i1} and $\overline{h}_{i1} = h_{i2} = x_i - h_{i1}$ arose by mutation from two ancestral haplotype sequences, μ_{i1} and μ_{i2}. Our random mating assumption is that these two ancestral haplotypes were drawn independently from the ancestral haplotype density, $\pi(\mu)$. The relationship between h_{i1}, $\overline{h}_{i1}$ and x_i is summarized in (3.2).

$$x_i(s) = \begin{cases} 0 & h_{i1}(s) = \overline{h}_{i1}(s) = 0 \\ 1 & h_{i1}(s) = 1 \ , \ \overline{h}_{i1}(s) = 0 \text{ or } \ h_{i1}(s) = 0 \ , \ \overline{h}_{i1}(s) = 1 \\ 2 & h_{i1}(s) = \overline{h}_{i1}(s) = 1 \end{cases} \tag{3.2}$$

where s is a site on x_i.

The sites on x_i with genotype=1 will be called **ambiguous**. Any genotype sequence x_i having at most one ambiguous site will be called a **trivial** genotype because there is a unique pair of haplotype sequences that generate such a genotype. We call the x_i's with two or more ambiguous sites **nontrivial**.

The ancestral mixture model for the genotype sequences, x's, uses the haplotype density, $\pi(\mu)$, and a component (or kernel) density, $g(x, \mu_1, \mu_2)$. Here μ_1 and μ_2 represent the two unknown ancestral haplotypes. We construct a kernel function $g_p(x(s); \mu_1(s), \mu_2(s))$, the **genotype mutation kernel**, that carries the dual effects of the mutations and the missing haplotype data. It is the conditional probability of $x(s)$ taking on value 0, 1 or 2, given the known haplotype ancestors $\mu_1(s)$ and $\mu_2(s)$. It has 3x2x2=12 possible values that can be constructed in a table for repeated calculations. Here m_p is the mutation kernel; a and b are 0 or 1.

$$\begin{aligned} g_p(0; a, b) &= P(H_1 = 0, H_2 = 0 | \mu_1 = a, \mu_2 = b) \\ &= m_p(0, a) m_p(0, b) \\ g_p(2; a, b) &= P(H_1 = 1, H_2 = 1 | \mu_1 = a, \mu_2 = b) \\ &= m_p(1, a) m_p(1, b) \\ g_p(1; a, b) &= P(H_1 = 1, H_2 = 0 | \mu_1 = a, \mu_2 = b) + P(H_1 = 0, H_2 = 1 | \mu_1 = a, \mu_2 = b) \\ &= m_p(1, a) m_p(0, b) + m_p(0, a) m_p(1, b) \end{aligned} \tag{3.3}$$

We then construct the mixture density of genotype sequence x as follows,

$$f(x;\pi) = \sum_{\mu} g_p(x;\mu_1,\mu_2)\pi(\mu_1)\pi(\mu_2). \tag{3.4}$$

One can subsequently write the likelihood function for the ancestral mixture model given a random sample genotype sequences, x_1,...,x_n from (3.4), as follows,

$$L(\pi; X = x_1,x_2,...,x_n) = \prod_i \sum_{\mu} g_p(x_i;\mu_1,\mu_2)\pi(\mu_1)\pi(\mu_2). \tag{3.5}$$

Note that the parameter of interest is $\pi(\mu)$, the density of ancestral haplotypes.

Our methodology requires an initial guess of $\pi(\mu)$ in the likelihood (3.5). In this paper, we will use the independent starting density, π_0 as the starting value of π,

$$\pi_0(\mu) = \prod_s \gamma_s^{\mu(s)}(1-\gamma_s)^{1-\mu(s)} \tag{3.6}$$

where γ_s's represent the probability of 1 at site s. We call it the independent starting density because this density corresponds to drawing each $\mu(s)$ independently using $\Pr(\mu(s)=1) = \gamma_s$. We set $\Pr(\mu(s)=1)$ to be the observed relative frequency of haplotype 1 at site s.

3.2.3 The density of the ancestral haplotypes for the genotype-haplotype model, $\pi_1(\mu)$

We update $\pi_0(\mu)$ (3.6) to $\pi_1(\mu)$ by the LEM method. The derivation of $\pi_1(\mu)$ is given in Appendix A. The $\pi_1(\mu)$ density is guaranteed to have higher likelihood in (3.5) than π_0.

The estimated ancestral haplotype density for the genotype-haplotype model, $\pi_1(\mu)$ is

$$\pi_1(\mu) = n^{-1}\sum_i \prod_s r_s(x_i(s),\mu(s)). \tag{3.7}$$

where the kernel r for site s is defined by:

$$\begin{aligned} r_s(0,\mu) &= \frac{m_p(0,\mu)}{A_s}\gamma_s^\mu(1-\gamma_s)^{1-\mu} \\ r_s(2,\mu) &= \frac{m_p(1,\mu)}{B_s}\gamma_s^\mu(1-\gamma_s)^{1-\mu} \\ r_s(1,\mu) &= \frac{1-A_s m_p(0,\mu)-B_s m_p(1,\mu)}{2A_sB_s}\gamma_s^\mu(1-\gamma_s)^{1-\mu} \end{aligned} \tag{3.8}$$

and

$$\begin{aligned} A_s &= \gamma_s m_p(0,1)+(1-\gamma_s)m_p(0,0) \\ B_s &= \gamma_s m_p(1,1)+(1-\gamma_s)m_p(1,0). \end{aligned}$$

Here γ_s is the observed frequency of haplotype 1 at site s and m_p is the mutation kernel.

3.2.4 Objective function for haplotype inference

Given a genotype sequence x_i, one might wish to infer the two unseen haplotypes, h_1 and $\overline{h}_1$ that generated the genotype. We will use the probability $\Pr(H_{i1}=h_{i1}, H_{i2}=\overline{h}_{i1}|x_i;\pi_1)$ to construct an objective function for this problem. Based on our mixture model, we construct the conditional probability of the haplotype pair, h_{i1} and $\overline{h}_{i1}$ given the data, using the density estimator π_1 described above. First note that h_{i1} and $\overline{h}_{i1}$ are under the constraint of $\overline{h}_{i1}=x_1-h_{i1}$, so there is indeed only one variable between h_{i1} and $\overline{h}_{i1}$. Maximizing this conditional probability is equivalent to maximizing

$$f(h_{i1}) = \sum_{\mu_1}\sum_{\mu_2} M_p(h_{i1},\mu_1)M_p(\overline{h}_{i1},\mu_2)\pi_1(\mu_1)\pi_1(\mu_2). \tag{3.9}$$

Here M_p is the mutation kernel between two haplotype sequences and π_1 is given in (3.7).

In order to calculate (3.9), it is best to write the π_1 terms as in (3.7), and then change

the orders of summation in μ and x to get:

$$n^{-2} \sum_j \sum_k \prod_s k_4(h_1(s), x_j(s)) k_4(\overline{h}_{i1}(s), x_k(s)) \tag{3.10}$$

where the new k_4 function is expressed as

$$k_4(h, x) = m(h, 0) r_s(x, 0) + m(h, 1) r_s(x, 1). \tag{3.11}$$

Note that k_4 has 2 x 3 possible arguments, corresponding to $\{0,1\} \times \{0,1,2\}$. To increase computational efficiency, these six values can be calculated and put in a table, then reused throughout an analysis. The kernel density estimator of the haplotype pair, h_{i1} and $\overline{h}_{i1}$, (3.10) then requires a summation of n^2 terms. The derivation of formula (3.10) is shown in Appendix B.

3.2.5 Modal EM (MEM)

We describe here the Modal EM (MEM) algorithm for finding modes on a density function of ϕ. Given any density that can be expressed in the form

$$k(\phi) = \sum_a \prod_b K(\phi; a, b),$$

where a and b are arbitrary summation and multiplication indices, one can find a mode of the density nearest a chosen starting value ϕ_0 by iteratively maximizing until convergence

$$\phi_{t+1} = \arg\max_{\phi} \sum_a w_t(a) \sum_b \{\log K(\phi, a, b)\},$$

where the weights $w_t(a)$ are determined by the previous value of ϕ_t as follows:

$$w_t(a) = \frac{\prod_b K(\phi_t; a, b)}{\sum_c \prod_b K(\phi_t; c, b)}$$

(Li, Ray and Lindsay, 2007).

We can find the modes of (3.9) using the MEM algorithm as follows: for a given x_i and the current update of h_{i1}, $h_{i1}^{<t>}$, let

$$h_{i1}^{<t+1>} = \arg\max_{h_{i1}} n^{-2} \sum_j \sum_k w^{<t>}(x_j, x_k) \sum_s \log\{k_4(h_{i1}(s), x_j(s)) k_4(\overline{h}_{i1}(s), x_k(s))\} \quad (3.12)$$

where

$$w^{<t>}(x_j, x_k) = \frac{\prod_s k_4(h_{i1}^{<t>}(s), x_j(s)) k_4(\overline{h}_{i1}^{<t>}(s), x_k(s))}{\sum_j \sum_k \prod_s k_4(h_{i1}^{<t>}(s), x_j(s)) k_4(\overline{h}_{i1}^{<t>}(s), x_k(s))}. \quad (3.13)$$

Recall that $h_{i1}^{<t>} = x_i - \overline{h}_{i1}^{<t>}$. A computational saving of (3.12) is that the variables of interest $h_{i1}(s)$ are each found in a separate summand, and so can be optimized separately.

The update of a particular site s on haplotype h_{i1} is

$$h_{i1}^{<t+1>}(s) = \arg\max_{h_{i1}} n^{-2} \sum_j \sum_k w^{<t>}(x_j, x_k) \log\{k_4(h_{i1}(s), x_j(s)) k_4(\overline{h}_{i1}(s), x_k(s))\}. \quad (3.14)$$

The numerically difficult part of this maximization is the calculation of the weights $w^{<t>}$. However, once calculated, maximization over $h_{i1}(s)$ can be done by comparing the two possible values, 0 and 1.

We considered two approaches for executing the above MEM algorithm. The main difference is in how the weights, $w^{<t>}(x_j, x_k)$ in (3.13) were calculated.

- Whole length MEM (wMEM): the procedure described above will be called the whole length MEM because we update all the $h_{i1}(s)$ using a single set of weights. The computational cost of solving for one genotype sequence is $n^2 L$.

- Reweighted MEM (rMEM): An alternative method is to use the MEM for one $h_{i1}(s)$ at a time, proceeding from **left to right**. In this case, one can update the MEM weights for $h_{i1}(s)$ using the new values $h_{i1}(t)$ determined for sites $t = 1, ..., s-1, s+1, ..., L$. This idea of updating weights for every site is similar to the idea of Expectation Conditional Maximization (ECM) in which one maximizes the EM likelihood function over one set of parameters while holding the remaining parameters fixed. The rMEM method has a computational cost of n^2L^2 but as we shall see, tends to climb to higher modes on the objective function than the wMEM algorithm provided that both algorithms are given the same starting value.

Starting value strategies

We considered four strategies for selecting starting values for the haplotype, h_{i1}. Note that one needs to select starting values only for the nontrivial individuals. The starting values below are listed in order of increasing complexity. The more complex methods are also more effective.

- Naive Starting values: We choose the starting value of h_{i1} by letting the starting value $h_{i1}^{<0>}(s) = 1$ for $x_i(s) = 1$ for all the sites, s, on h_{i1}.

- Trivial-individual based starting values: We start by separating the trivial individuals from the nontrivial individuals. From the trivial individuals we can construct a list of solved haplotypes. This idea was proposed originally by Clark (1990).

- Neighbor starting values:

    1. First we create a matrix of **partially** known haplotype sequences based on genotypes x_i, ..., x_n. If the genotype at a site $x_i(s) = 2$, the corresponding **partial** haplotype $h_1(s) = 1$. A genotype of 0 implies the **partial** haplotype is 0. For the genotype of 1, we write haplotype $h_1(s) = x$, where x means unknown. This

procedure creates a list of the **partial** haplotypes (PH), which look like .

$$PH = \begin{cases} 1111\text{x}1000\text{xx}1100101 & ...00\text{x}110010 \\ 1\text{x}111\text{x}1000\text{x}1100101 & ...0011\text{x}0110 \\ 1111\text{x}1000\text{x}11001\text{x}01 & ...0\text{x}0110010 \\ ..... & ..... \end{cases} \tag{3.15}$$

2. To create a starting value for an individual i with genotype x_i, we identify the locations where $x_i(s) = 1$. We label the locations from the left as $a_1, a_2, ...$

3. Next we count the number of solved haplotypes (1,0), (0,0), (0,1) and (1,1) on the pair of first two sites (a_1, a_2) based on the PH list. We denote the counts as n_{10}, n_{00}, n_{01} and n_{11}. If $n_{10} \times n_{01} > n_{11} \times n_{00}$, then the genotype (1,1) on sites (a_1, a_2) seems more likely to have arisen from the haplotype pair (0,1)×(1,0) than from the pair (1,1)×(0,0). Otherwise (1,1)×(0,0) pair seems more likelihood. If it is a tie, we randomly choose between (0,1)×(1,0) and (1,1)×(0,0).

4. Then we repeat step 3 for the pair of sites (a_2, a_3), now picking a pair of haplotypes, either (0,1)×(1,0) or (0,0)×(1,1). To make a single pair of haplotypes for (a_1, a_2, a_3), we match on values at site a_2. For example, if the pairs at (a_1, a_2) are (x,0)×($\bar{x}$,1) and the pairs at (a_2, a_3) are (0,y)×(1,$\bar{y}$), then the new pairs are (x,0,y)×($\bar{x}$,1,$\bar{y}$).

5. We repeat steps 3 and 4 for all adjacent pairs of (a_j, a_{j+1}) to get the starting values of all the ambiguous sites for individual i.

- Pseudo-kernel-estimator based strategy: We will let h_1^*, h_2^*, ... h_n^* be the partial haplotypes in PH (3.15). We can think of the x's as being missing values in a haplotype model (Appendix C). If one treats the x values as missing-at-random data (and they are not), then one can construct a kernel density estimator for the haplotype density

as follows: Let M_i be the set of indices of the missing values for h_i^* and let M_i^c be the indices of the observed values. The pseudo kernel density for the haplotype model with the missing values is

$$\pi_1^*(\phi) = n^{-1} \sum_i \prod_s (h_p(h_i^*(s), \phi(s)))^{\mathbf{I}(s \in M_i^c)} (\frac{1}{2})^{\mathbf{I}(s \in M_i)}. \tag{3.16}$$

where the h_p kernel is defined in Appendix C. Here we let the kernel function value for the missing values be $\frac{1}{2}$, which is equivalent to assuming the probability of observing 1 at the missing site to be the same as that of observing 0. To construct an initial value for inferring the two haplotypes corresponding to x_i, we start with the partial haplotype h_i^*. Given an individual with partial haplotype h_i^*, we let Φ_i be the set of all binary sequences ϕ that match h_i^* at all unambiguous sites. Here we encounter a second optimization problem, namely to choose haplotype $\phi_i \in \Phi_i$ such that

$$\phi_i = \arg max\{\pi_1^*(\phi), \phi_i \in \Phi_i\}.$$

The solution to the optimization problem becomes the pseudo-kernel-estimator based starting value.
We consider the following algorithm to find a mode of $\pi_1^*(\phi)$ with the constraint that $\phi_i \in \Phi_i$.

1. We start with the first partial haplotype in (3.15) and maximize (3.16) over ϕ = 1111(0,1)1000xx1100101..., where $(0, 1)$ replaces the x notation on the first site with missing value. We use $h_p(h_i^*(j), \mathrm{x}) = \frac{1}{2}$ for the missing sites except the first one. The maximization literally means we calculate (3.16) for ϕ_1 = 111111000xx1100101... and ϕ_2 = 111101000xx1100101.... We choose between ϕ_1 and ϕ_2 with the higher value of the objective function (3.16).

2. Suppose we choose 1 for the first missing value. When we move to the second

missing value, we condition on the result from step 1. For the second missing value, we maximize (3.16) over $\phi = 111111000(0,1)x1100101...$ and repeat step 1.

3. We can repeat step 2 for all missing values in PH. This is the **left to right** calculation. One might also do it in a **right to left** fashion by starting with the first missing value on the right.

Thus, we have four strategies of choosing starting values of h_{i1}, the naive strategy, the trivial-individual strategy, the neighbor strategy and the pseudo-kernel-estimator strategy.

Combining results from different starting values

The MEM algorithm may reach different modes from different starting values due to the multimodality of the density estimator (3.10). So we select the best mode, namely the one that optimizes the objective function (3.9) by calculating the heights of different modes and picking the highest mode. Note that the trivial-individual strategy may not be applicable for longer genotype sequences. The reason is that the relative frequency of trivial individuals is smaller for longer sequences; the number of trivial individuals is associated with the number of trivial individual based starting haplotypes, which means we may not find any proper starting haplotypes for some of the nontrivial individuals. For data analysis, we indeed combine the trivial-individual strategy and the naive strategy. In the simulation study we explore the efficacy of using several different sets of initial values and combining them.

3.2.6 Simulation Data and Real Data

We examined the effectiveness of our methodology using a simulation study and a real data set. In the simulation study, we created the ancestral density $\pi(\phi)$ by first randomly generating two ancestral haplotype sequences of different lengths L =8, 16 and 32. The two sequences were generated by (3.6) with all $\gamma_s = .5$. We gave each ancestor weight .5. We then randomly drew from the ancestor haplotype density 100 times. We then created six datasets

of 100 haplotype sequences by mutating each site with the probabilities, P_{sim} = .005, .01, .05, .1, .15 and .2. We call the mutation probabilities (P_{sim}) the mutation scenarios. These generated haplotypes were then randomly paired to form n=50 genotypes. In all, there were $3 \times 6 = 18$ combinations for L and P_{sim}. We then created 5 sets of 100 haplotype sequences for each combination. In order to evaluate the role of the tuning parameter in our estimator, we used five different choices for p, namely .05, .1, .2, .3 and .4, in the MEM algorithm.

For the real data, we downloaded the genotype sequences of Yoruba people on chromosome 1 from HapMap release 2 (HapMap, 2003 and 2006). The genotype data was of 60 individuals and 1,600 single nucleotide polymorphisms (SNP). A single nucleotide polymorphisms is the genetic marker that differs between individuals on single nucleotide. SNP data can be transformed into binary sequences by coding one of variant of each site as 0 and the other as 1. For each genotype sequences, the haplotype pair of the genotype sequences are known based on familial information. To account for recombination and the decay of linkage disequilibrium in real genotype data, we divided the SNPs into smaller subsequences in three different ways. The three partitions of the 1,600 SNPs were 50 subsequences of 32 sites, 100 subsequences of 16 sites and 200 subsequences of 8 sites. The idea of partitioning long sequences into shorter subsequences for haplotype inference problem was adopted by PL-EM (Qin, Niu and Li, 2002) and PHASE. In the latest version of PHASE, the number of sites in the partitioned subsequences is between 6 and 8. Since we do not have a partition-ligation algorithm, the comparison between our method and PHASE is fair only when we use short sequences.

3.2.7 Measurement of haplotyping error and PHASE

To compare the performance of our method and PHASE's, we used err_{site}, which is the ratio of the number of incorrectly phased sites to the number of non-trivial heterozygous sites. The number of incorrectly phased sites is obtained by comparing the estimated haplotype pairs (by MEM or PHASE) to the known haplotypes (for the simulation study) or the phased

haplotypes (for the real data). The err_{site} is the same as the IGP measurement used in Marchini, et al. (2006).

We used the -MS option in the Phase program, which corresponded to using the method of not inferring recombinations. We kept all the other parameters in PHASE as default.

3.3 Results

3.3.1 The simulated data

In Table 3.1 and the tables in Appendix D, we compare the performance of various MEM algorithms with different tuning parameters, p = .05, .1, .2, .3, and .4, and the PHASE program on the simulated data. We used the four starting value strategies. We also combined the results of the four starting value strategies by choosing the highest mode. If the global maximum of the objective function is indeed the best estimator based on the haplotype density estimator, then the method of combining the results from different starting values should be closer to the maximum, and hence the best method. The comparison of method was based on the total error rate, err_{site}. The first five columns provide a detailed comparison of various starting value strategies; the sixth column presents results from the rMEM, which seems to perform well using the naive starting value, and the seventh column provides the results from PHASE for comparison. We tabulated the error rate by the lengths of the sequences (L), the mutation scenarios (P_{sim}) and the haplotype inference methods being used. Only the results for the tuning parameter p=.1 are reported in the Table 3.1, as the results were very robust to this parameter (see Appendix D). The results of the MEM algorithms using the other levels of tuning parameter, p are in Appendix D.

This study provided evidence that the wMEM method with multiple starting values (column 5, Table 3.1) was the best method among all the wMEM. The rMEM method in column 6 was comparable to the wMEM method with multiple starting values. The better performance of the wMEM method with multiple starting values and the rMEM method with

naive strategy point to the importance of allocating extra effort to seek the best solution to the optimization problem. However, wMEM with multiple starting values should be regarded as a better approach because of computational efficiency. The wMEM method with multiple starting values completed the analysis of all $5 \times 18 = 90$ data sets with p=.1 for less than 2 hrs while rMEM required 8 hrs. Both algorithms were executed on a single node server (Sun SunFire v20z) with 2 AMD Opteron 250 2.4 GHz Processors and 8 GB memory. Last but not least, the wMEM algorithm with multiple starting values performed slightly better than PHASE in column 7.

The effect of the tuning parameters on the MEM algorithm with multiple starting values is small when we compare Table 3.1 with the tables in Appendix D. Without further analysis on the choice of tuning parameter, we chose to use $p = .1$ for the wMEM algorithm with multiple starting values for the analysis of the real data.

3.3.2 Analysis of the real data

We analyzed three partitions of the raw data, the 50 subsequences each with 32 sites, 100 each with 16 sites and 200 each with 8 sites. The data sets were analyzed by the wMEM algorithm with multiple starting values for the tuning parameter, $p = .1$ as well as PHASE. The err_{site} is summarized in Table 3.2. The wMEM algorithm with multiple starting values was equivalent to PHASE for length 8, but showed signs of being inferior for length 16 and 32, although the difference between wMEM and PHASE was not of statistical significance. It did appear that there was some deterioration in relative performance as sequence length increased.

3.4 Discussion

We restricted our analysis to short sequences because the underlying model assumes the haplotype pairs are the descendants of the ancestral haplotypes by mutation but not by

	Simulated mutation rate	1	2	3	4	5	6	7
L=8	.005	0.003 (0.003)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0.002 (0.002)
	.01	0.007 (0.007)	0.001 (0.001)	0.001 (0.001)	0.001 (0.001)	0.001 (0.001)	0.001 (0.001)	0.001 (0.001)
	.05	0.097 (0.023)	0.073 (0.018)	0.143 (0.026)	0.067 (0.016)	0.073 (0.018)	0.073 (0.018)	0.088 (0.013)
	.1	0.248 (0.036)	0.099 (0.012)	0.206 (0.024)	0.100 (0.015)	0.096 (0.011)	0.094 (0.010)	0.123 (0.009)
	.15	0.292 (0.016)	0.218 (0.009)	0.271 (0.012)	0.200 (0.008)	0.210 (0.009)	0.206 (0.015)	0.204 (0.021)
	.2	0.296 (0.019)	0.230 (0.010)	0.287 (0.013)	0.232 (0.010)	0.229 (0.011)	0.233 (0.007)	0.246 (0.015)
L=16	.005	0.278 (0.069)	0.004 (0.003)	0.004 (0.003)	0.004 (0.003)	0.004 (0.003)	0.004 (0.003)	0.003 (0.002)
	.01	0.086 (0.065)	0.015 (0.004)	0.011 (0.004)	0.015 (0.004)	0.015 (0.004)	0.015 (0.004)	0.021 (0.004)
	.05	0.203 (0.062)	0.093 (0.016)	0.105 (0.014)	0.090 (0.017)	0.091 (0.017)	0.090 (0.018)	0.107 (0.022)
	.1	0.289 (0.028)	0.158 (0.017)	0.171 (0.022)	0.134 (0.018)	0.138 (0.017)	0.138 (0.016)	0.168 (0.016)
	.15	0.313 (0.009)	0.271 (0.009)	0.273 (0.007)	0.226 (0.011)	0.226 (0.010)	0.241 (0.014)	0.246 (0.007)
	.2	0.340 (0.012)	0.324 (0.011)	0.326 (0.008)	0.278 (0.013)	0.280 (0.013)	0.280 (0.015)	0.276 (0.010)
L=32	.005	0.407 (0.024)	0.017 (0.008)	0.016 (0.008)	0.017 (0.008)	0.017 (0.008)	0.017 (0.008)	0.016 (0.006)
	.01	0.350 (0.072)	0.021 (0.003)	0.018 (0.003)	0.020 (0.003)	0.021 (0.003)	0.021 (0.003)	0.015 (0.002)
	.05	0.350 (0.019)	0.094 (0.021)	0.106 (0.021)	0.093 (0.022)	0.093 (0.021)	0.093 (0.021)	0.094 (0.018)
	.1	0.358 (0.011)	0.211 (0.016)	0.212 (0.017)	0.190 (0.010)	0.190 (0.010)	0.193 (0.009)	0.220 (0.013)
	.15	0.338 (0.013)	0.294 (0.020)	0.307 (0.011)	0.281 (0.020)	0.281 (0.020)	0.279 (0.020)	0.288 (0.012)
	.2	0.374 (0.008)	0.340 (0.007)	0.335 (0.007)	0.322 (0.013)	0.322 (0.013)	0.332 (0.010)	0.324 (0.012)

Table 3.1: Comparison of wMEM with different starting values, rMEM and PHASE under the simulation scheme with the tuning parameter, $p = .1$. For each cell, the first number is the average err_{site} rate and the second number is its standard error. 1 means wMEM with the naive starting value; 2 means wMEM with the naive starting value and the trivial individual starting value; 3 means wMEM with the neighbor starting value; 4 means wMEM with the missing data based starting value; 5 means the highest modes among 1, 2, 3 and 4; and 6 means the rMEM algorithm with the naive starting value; 7 means the result from PHASE.

	wMEM with multiple starting values	PHASE
50 subsequences of 32 sites	.075 (.007)	.063 (.007)
100 subsequences of 16 sites	.044 (.004)	.038 (.004)
200 subsequences of 8 sites	.025 (.002)	.024 (.002)

Table 3.2: Performance of wMEM algorithms with multiple starting values for the tuning parameter, $p = .1$. The first number is the average err_{site} among the subsequences. The number in the parenthesis is the standard error.

recombination. For the simulation study, where we only allowed mutations, our method performed slightly better than PHASE on the simulated data regardless of the sequence length.

There are well documented recombination events in the real data (The International HapMap Consortium, 2003). The deteriorating performance of our method on longer subsequences in the real data can be partially explained by the existence of recombination. Recombination should create **new** ancestral haplotypes due to the recombination between the original ones. Because recombination is rare, the new ancestral haplotypes due to recombination are also of low relative frequency. For models not designed for recombination such as ours, a recombination event may be treated as multiple mutation events, which is a parsimonious solution under the model assumption. Furthermore, the fact that our method has equivalent performance to PHASE in sequences of length 8 coincides with the default partition size in PHASE, 6-8 sites per segment. This coincidence suggests our method, once integrated with a proper partition-ligation algorithm, may be a good alternative to the currently available statistical methods of inferring haplotypes for longer sequences.

One main contribution of our method is to model haplotype sequences under the framework of kernel density estimation. One potential advantage of using a density estimator

for inferring haplotypes over the available parametric mixture models is the model flexibility provided by the tuning parameter, p. One can adjust the smoothness of the density estimator by adjusting the tuning parameter. We did not fully investigate how to choose a good tuning parameter for a given data set. But we acknowledged the choice of p affects the performance of our method (in Table 3.1 and the tables in Appendix D). We propose to investigate the use of degrees of freedom (DOF) or cross-validation for selecting a good tuning parameter.

The novelty of our method also lies in the proposed starting value strategies. Additional analysis shows the MEM algorithm with multiple starting values has equivalent performance to an enumerative algorithm that evaluates the conditional probability of haplotype pair on all the possible points (data not shown).

We have focused on here developing a modal based haplotype inference method. But the main underlying motivation for our work is the potential use of our method in disease studies. Our method is based on a density estimator of ancestral haplotype distribution. If we focus on the ancestral haplotypes themselves instead of the haplotype pairs that generates the genotype sequences, we can use the modes on the density of the ancestral haplotype sequences to conduct association studies. Using haplotype sequences under density estimation framework instead of using individual genetic markers, such as SNPs, not only increases power (Zaykin, et. al. 2002) but also provides more flexibility to the analysis. This will be our future direction.

3.5 Conclusion

We approached the haplotype inference problem under the framework of density estimation. We constructed an objective function for haplotype inference using the density of ancestral haplotypes; We adapted the Modal EM (MEM) method to maximize the objective function by finding its modes. We designed two version of MEM, the whole length MEM (wMEM) method and the reweighted MEM (rMEM) method. The rMEM method could find higher

modes than the wMEM method if they were assigned with the same starting value but rMEM required more computational time. We also used four strategies of choosing starting values for the MEM algorithm; the performance of the starting value strategies varied. We found the best overall strategy for inferring haplotype pair from genotype sequence was to use the wMEM method with multiple starting values.

The simulation study and the real data analysis showed the proposed wMEM method had the equivalent performance to PHASE in haplotype inference. The simulation study was designed in a way that the SNPs were dependent under different mutation scenarios. The wMEM method had the equivalent performance to PHASE under different mutation scenarios and short sequence lengths. To accommodate the assumption of the proposed wMEM method, we divided long genotype sequences in the real data into subsequences so that the sites in each subsequence were presumably dependent on each other. Although the wMEM method underperformed PHASE for subsequences of length 32 and 16, its performance was improved to the level of PHASE's for sequence length of 8. Both the simulation study and the analysis on real data suggest the wMEM method with multiple starting values is a good alternative to the currently available methods such as PHASE for short sequences.

Chapter 4

Model for genotypes

The motivation of this chapter was to establish a diagnostic tool for analyzing genotype data directly based on *DOF* calculations. We examined the correlation between different DOF calculations and the haplotyping error. We then related the findings in the DOF calculation and haplotyping error to the genetic concept of linkage disequilibrium. We proposed a simple partition strategy to divide long sequences into smaller subsequence such that the dependence between the sites within each subsequences were strong. We also explored the possibility of clustering the genotypes in the same way as we presented in Chapter 2 but the results of clustering were overall negative.

4.1 Models and kernels for genotype problem

4.1.1 Models

We will use the independence starting density, π_0 described earlier in Chapter 2, for genotype, φ of conditional independence form.

$$\pi_0(\varphi) = \prod_s \gamma_s^{I\{\varphi(s)=2)}(2)\gamma_s^{I\{\varphi(s)=1)}(1)\gamma_s^{I\{\varphi(s)=0)}(0),$$

where $\gamma_s(.)$ is the probability of genotype x at site s.

For γ_s, one could use either the observed probability of genotypes or the expected ones. The observed probabilities of genotypes are just the relative frequency of genotype 0, 1 and 2, denoted as $\alpha_s(0)$, $\alpha_s(1)$ and $\alpha_s(2)$. The starting density would be

$$\pi_0(\varphi) = \prod_s \alpha_s^{I\{\varphi(s)=2)}(2)\alpha_s^{I\{\varphi(s)=1)}(1)\alpha_s^{I\{\varphi(s)=0)}(0). \tag{4.1}$$

The expected probabilities of genotype are defined under the random mating assumption, known as the Hardy Weinberg Equilibrium (HWE) model. First we get the estimated probability of haplotypes 0 and 1 by $\alpha_s(0) + \frac{1}{2}\alpha_s(1)$ and $\alpha_s(2) + \frac{1}{2}\alpha_s(1)$, respectively. Then the expected probabilities of the three genotypes, β_s, are

$$\begin{aligned}
\beta_s(0) &= \{\alpha_s(0) + \frac{1}{2}\alpha_s(1)\}^2 \\
\beta_s(1) &= 2\{\alpha_s(0) + \frac{1}{2}\alpha_s(1)\}\{\alpha_s(2) + \frac{1}{2}\alpha_s(1)\} \\
\beta_s(2) &= \{\alpha_s(2) + \frac{1}{2}\alpha_s(1)\}^2.
\end{aligned} \tag{4.2}$$

The starting density using the expected probabilities is

$$\pi_0(\varphi) = \prod_s \beta_s^{I\{\varphi(s)=2)}(2)\beta_s^{I\{\varphi(s)=1)}(1)\beta_s^{I\{\varphi(s)=0)}(0).$$

We call the second model the ″Hardy Weinberg model″ and the first one the ″independent site model″.

4.1.2 Kernels

We define two genotype kernels and call them the **A**symmetric kernel (g_{A0}) and the **N**aive Kernel (g_{N0}).

4.1.2.1 The Asymmetric kernel

We start with the idea of the **genotype mutation kernel** from last chapter,

$$\begin{aligned} g(0;a,b) &= m_p(0,a)m_p(0,b) \\ g(2;a,b) &= m_p(1,a)m_p(1,b) \\ g(1;a,b) &= m_p(1,a)m_p(0,b) + m_p(0,a)m_p(1,b). \end{aligned} \tag{4.3}$$

Note that when x is fixed at 0, 1, or 2, $g(x;a,b)$ only depends on the ancestral haplotypes (a,b) only through their total $a+b$. Thus we define the ancestral genotype, $\phi(s) = a+b$. Here "ancestral genotype" is not a precise definition because its definition, $\phi(s) = a+b$, does not have a counterpart in reality. Let us revisit the mutation kernel in the haplotype model (Chapter 2). The mutation kernel is defined between two haplotypes based on continuous time Markov Chain (CTMC). The two haplotypes in the mutation kernel are considered the states of a Markov Chain. A mutation is then deemed the change between the states in the Markov Chain. On the other hand, a genotype is not simply a mutated version of its "ancestor" but a random mixture of two haplotypes. Here we construct kernel functions between two genotypes not entirely in the context of genetics.

We define a kernel on genotype space by

$$g_{A0}(x(s), \phi(s)) = g(x(s)|\phi(s) = a+b)$$

Here $\phi(s)$ takes value of 0, 1, 2. We can write the kernel in the matrix form (Table 4.1a). Note the kernel matrix is not symmetric because $g_{A0}(0,1) \neq g_{A0}(1,0)$ for $i \neq j$. It is a density in x but not ϕ. So we call it the Asymmetric kernel.

g_{A0}	ϕ=0	1	2
x=0	$m_\tau(0,0)m_\tau(0,0)$	$2m_\tau(0,0)m_\tau(0,1)$	$m_\tau(0,1)m_\tau(0,1)$
1	$m_\tau(0,0)m_\tau(1,0)$	$m_\tau(1,0)m_\tau(0,1) + m_\tau(0,0)m_\tau(1,1)$	$m_\tau(0,1)m_\tau(1,1)$
2	$m_\tau(1,0)m_\tau(1,0)$	$2m_\tau(1,0)m_\tau(1,1)$	$m_\tau(1,1)m_\tau(1,1)$

(a) Asymmetric kernel for genotype

g_{N0}	y=0	1	2
x=0	$1-p-p^2$	p	p^2
1	p	$1-2p$	p
2	p^2	p	$1-p-p^2$

(b) Naive Kernel for genotype

Table 4.1: Genotype kernels

4.1.2.2 The Naive kernel

We define a simple kernel based on the number of changes between genotype x and genotype y (the second sub table in Table 4.1b). It is symmetric and a density in both variables.

$$g_{N0}(1,0) = g_{N0}(0,1) = g_{N0}(1,2) = g_{N0}(2,1) = p$$

and

$$g_{N0}(0,2) = g_{N0}(2,0) = p^2.$$

Then we have

$$g_{N0}(0,0) = g_{N0}(2,2) = 1 - p - p^2$$

and

$$g_{N0}(1,1) = 1 - 2p,$$

where p is defined as the probability of one change (for instance, $0 \to 1$ is one change and $0 \to 2$ are two changes). We consider 0, 1 and 2 to be discrete states and ignore the random mating principle. Here the naive kernel has the identifiability problem of p. This p has the same notation of mutation rate, p but has different meaning. For the genotype state to be identifiable, we must have $p < \frac{\sqrt{3}-1}{2}$.

4.1.3 LEM step 1 and the updated kernels

We can obtain one-step LEM updated densities for (4.1) and (4.2) using the procedures described in Section 2.2. We start with the initial density for genotype,

$$\pi_0(\phi) = \prod_s \gamma_s^{I\{\phi(s)=2)}(2)\gamma_s^{I\{\phi(s)=1)}(1)\gamma_s^{I\{\phi(s)=0)}(0),$$

where γ_s takes either α or β (4.1 and 4.2). The genotype kernel is then

$$G_0(x_i, \phi) = \prod_s g_0(x_i(s), \phi(s))$$

where g_0 can be either g_{A0} or g_{N0}. Then the gradient function is

$$\Delta(\phi) = n^{-1} \sum_i \frac{G_0(x_i, \phi)}{\sum_\phi G_0(x_i, \phi)\pi_0(\phi)}.$$

The numerator is written as

$$G_0(x_i, \phi) = \prod_s g_0(x_i, \phi(s))$$

and the denominator becomes

$$\sum_\phi G_0(x_i, \phi)\pi_0(\phi) = \prod_s \{g_0(x_i, 0)\gamma_s(0) + g_0(x_i, 1)\gamma_s(1) + g_0(x_i, 2)\gamma_s(2)\}$$

Thus,

$$\Delta(\phi) = n^{-1} \sum_i \prod_s \frac{g_0(x_i, \phi(s))}{g_0(x_i, 0)\gamma_s(0) + g_0(x_i, 1)\gamma_s(1) + g_0(x_i, 2)\gamma_s(2)}$$

and the one-step LEM updated density for φ,

$$\begin{aligned}
\pi_1(\phi) &= \pi_0(\phi)\Delta(\phi) \\
&= \prod_s \gamma_s^{I\{\phi(s)=2\}}(2)\gamma_s^{I\{\phi(s)=1\}}(1)\gamma_s^{I\{\phi(s)=0\}}(0)\Delta(\phi) \\
&= n^{-1} \sum_i \prod_s \frac{\gamma_s^{I\{\phi(s)=2\}}(2)\gamma_s^{I\{\phi(s)=1\}}(1)\gamma_s^{I\{\phi(s)=0\}}(0) g_0(x_i(s), \phi(s))}{g_0(x_i, 0)\gamma_s(0) + g_0(x_i, 1)\gamma_s(1) + g_0(x_i, 2)\gamma_s(2)} \\
&= n^{-1} \sum_i \prod_s g_1(x_i(s), \phi(s)). \qquad (4.4)
\end{aligned}$$

The g_1 (updated kernel) under different models are given in Table 4.2. Notice that the updated kernels are not symmetric. Recall that we have the Hardy Weinberg model and the independence site model for the initial genotype density and two kernel functions, the asym-

metric kernel and the naive kernel. Therefore we have four models of 6.12 each corresponding to one of the combinations of the kernel functions and the initial densities,

- the Hardy Weinberg density with the Asymmetric kernel
- the Hardy Weinberg density with the Naive kernel
- the Independent site density with the Asymmetric kernel
- the Independent site density with the Naive kernel

To carry out the DOF calculation, we first calculate the $L2$ distance kernel,

$$k(x(s), y(s)) = \sum_{\phi} g_1(x_i(s), \phi(s)) g_1(y_i(s), \phi(s))$$

and then calculate

$$K(x, y) = \prod_{s} k(x(s), y(s)),$$

which is the kernel function of the DOF calculation.

4.2 Clustering genotype

We applied the MEM method described in Section 2.3 to the updated genotype density (6.12) to find the modes.

We used a 200-site SNP genotype data set of 100 unrelated CEU (Utah residents with ancestry from northern and western Europe) and 100 unrelated YRI (Yoruba in Ibadan, Nigeria) samples from HapMap. We first used STRUCTURE (Pritchard, 2003 and 2006) to cluster the genotypes. STRUCTURE is a very popular software for clustering genetic data. STRUCTURE clusters genotypes using a Bayesian mixture models with Markov Chain

$g_{.H1}$	ϕ=0	1	2
x=0	$\frac{g_{.0}(0,0)\beta_s(0)}{g_{.0}(0,0)\beta_s(0)+g_{.0}(0,1)\beta_s(1)+g_{.0}(0,2)\beta_s(2)}$	$\frac{g_{.0}(0,1)\beta_s(1)}{g_{.0}(0,0)\beta_s(0)+g_{.0}(0,1)\beta_s(1)+g_{.0}(0,2)\beta_s(2)}$	$\frac{g_{.0}(0,2)\beta_s(2)}{g_{.0}(0,0)\beta_s(0)+g_{.0}(0,1)\beta_s(1)+g_{.0}(0,2)\beta_s(2)}$
1	$\frac{g_{.0}(1,0)\beta_s(0)}{g_{.0}(1,0)\beta_s(0)+g_{.0}(1,1)\beta_s(1)+g_{.0}(1,2)\beta_s(2)}$	$\frac{g_{.0}(1,1)\beta_s(1)}{g_{.0}(1,0)\beta_s(0)+g_{.0}(1,1)\beta_s(1)+g_{.0}(1,2)\beta_s(2)}$	$\frac{g_{.0}(1,2)\beta_s(2)}{g_{.0}(1,0)\beta_s(0)+g_{.0}(1,1)\beta_s(1)+g_{.0}(1,2)\beta_s(2)}$
2	$\frac{g_{.0}(2,0)\beta_s(0)}{g_{.0}(2,0)\beta_s(0)+g_{.0}(2,1)\beta_s(1)+g_{.0}(2,2)\beta_s(2)}$	$\frac{g_{.0}(2,1)\beta_s(1)}{g_{.0}(2,0)\beta_s(0)+g_{.0}(2,1)\beta_s(1)+g_{.0}(2,2)\beta_s(2)}$	$\frac{g_{.0}(2,2)\beta_s(2)}{g_{.0}(2,0)\beta_s(0)+g_{.0}(2,1)\beta_s(1)+g_{.0}(2,2)\beta_s(2)}$

(a) Genotype kernels for Hardy Weinberg model (expected probabilities of genotypes, β)

$g_{.I1}$	ϕ=0	1	2
x=0	$\frac{g_{.0}(0,0)\alpha_s(0)}{g_{.0}(0,0)\alpha_s(0)+g_{.0}(0,1)\alpha_s(1)+g_{.0}(0,2)\alpha_s(2)}$	$\frac{g_{.0}(0,0)\alpha_s(1)}{g_{.0}(0,0)\alpha_s(0)+g_{.0}(0,1)\alpha_s(1)+g_{.0}(0,2)\alpha_s(2)}$	$\frac{g_{.0}(0,2)\alpha_s(2)}{g_{.0}(0,0)\alpha_s(0)+g_{.0}(0,1)\alpha_s(1)+g_{.0}(0,2)\alpha_s(2)}$
1	$\frac{g_{.0}(1,0)\alpha_s(0)}{g_{.0}(1,0)\alpha_s(0)+g_{.0}(1,1)\alpha_s(1)+g_{.0}(1,2)\alpha_s(2)}$	$\frac{g_{.0}(1,1)\alpha_s(1)}{g_{.0}(1,0)\alpha_s(0)+g_{.0}(1,1)\alpha_s(1)+g_{.0}(1,2)\alpha_s(2)}$	$\frac{g_{.0}(1,2)\alpha_s(2)}{g_{.0}(1,0)\alpha_s(0)+g_{.0}(1,1)\alpha_s(1)+g_{.0}(1,2)\alpha_s(2)}$
2	$\frac{g_{.0}(2,0)\alpha_s(0)}{g_{.0}(2,0)\alpha_s(0)+g_{.0}(2,1)\alpha_s(1)+g_{.0}(2,2)\alpha_s(2)}$	$\frac{g_{.0}(2,1)\alpha_s(1)}{g_{.0}(2,0)\alpha_s(0)+g_{.0}(2,1)\alpha_s(1)+g_{.0}(2,2)\alpha_s(2)}$	$\frac{g_{.0}(2,2)\alpha_s(2)}{g_{.0}(2,0)\alpha_s(0)+g_{.0}(2,1)\alpha_s(1)+g_{.0}(2,2)\alpha_s(2)}$

(b) Genotype kernels for independent site model (observed probabilities of genotypes, β)

Table 4.2: The general form of the updated genotype kernel, $g_{.1}$under different models

Models for genotype density	the tuning parameter value when there are only two modes in the density *	Cluster 1 †	Cluster 2 †
The Hardy Weinberg model with the Asymmetric kernel	.34	38 CEU, 31 YRI	62 CEU, 69 YRI
The Hardy Weinberg model with the Naive kernel	.31	28 CEU, 35 YRI	72 CEU, 65 YRI
The independence site model with the Asymmetric kernel	.36	33 CEU, 42 YRI	67 CEU, 58 YRI
The independence site model with the Naive kernel	.305	33 CEU, 43 YRI	67 CEU, 57 YRI

Table 4.3: The clustering of 200 genotype sequences by MEM. There are 100 YRI samples and 100 CEU samples in the data set.
* The tuning parameter was increased sequentially till the MEM method found only two modes in the density. No numerical analysis was conducted to find the minimum tuning parameter for which the associated genotype density had two and only two modes. So the numbers in this column should be interpreted with caution..
† The small cluster was labeled cluster 1 and the larger cluster cluster 2. The numbers of CEU samples and YRI samples were counted for each cluster.

Monte Carlo (MCMC) implementation. We set $K = 2$ (clusters) and chose "no mixing between clusters" for the STRUCTURE run. The MCMC algorithm converged after 500 burn-in iterations and 2,000 iterations of sampling. The results of STRUCTURE showed all the YRI samples were in one cluster and all the CEU samples were in the other cluster.

We chose the tuning parameter for the MEM method in a way that there were two and only two modes found in the genotype density (6.12). Here "two and only two modes found in the genotype density" means one should be able separate the individual genotype sequences into two clusters based on the modes. Recall that we constructed four genotype densities (6.12) with different combinations of the kernel functions and the initial densities. None of these densities produced the same result as STRUCTURE, which was to cluster YRI in one group and CEU in the other group. We summarized the clustering results using the four genotype densities in Table 4.3. We did not investigate the cause of the misclustering. Instead, we will focus on the DOF calculation of the genotype densities.

4.3 DOF of the genotype densities

We calculated the DOF's of the densities using the genotype kernels and the kernel from Chapter 3 and examine the correlation between them and the $DOFs$ of the haplotype density from Chapter 2.

First we construct the DOF calculation of genotype-haplotype model from Chapter 3. We define $k_{r\tau}$ to be the kernel of the DOF_r calculation

$$k_{r\tau}(x(s), y(s)) = \sum_{\phi=0}^{1} r_{s\tau}(x(s), \phi) r_{s\tau}(y(s), \phi).$$

We also include the DOF of the haplotype distribution, DOF_h in the analysis. The kernel of DOF_h is

$$k_{h\tau}(x(s), y(s)) = \sum_{\phi=0}^{1} h_{s\tau}(x(s), \phi) h_{s\tau}(y(s), \phi).$$

We define the kernel of DOF_g, the DOF of genotype-haplotype model as

$$k_{g\tau}(x(s), y(s)) = \sum_{\varphi=0,1,2} g_{\bullet 1\tau}(x(s), \varphi) g_{1\tau}(y(s), \varphi).$$

Recall that the DOF calculation has the form

$$DOF_a = \frac{[\sum K_{a\tau}(\underline{x}_i, \underline{x}_i)]^2}{\sum\sum K^2_{a\tau}(\underline{x}_i, \underline{x}_j)} = \frac{[\sum \prod_s k_{a\tau}(x_i(s), x_j(s))]^2}{\sum\sum \prod_s k_{a\tau}(x_i(s), y_i(s))}.$$

4.3.1 Correlations between $DOF_\bullet$ within a dataset

We give notations to the DOF's as follows, the asymmetric genotype kernel model under HWE (DOF_{ah}), the naive genotype kernel model under HWE (DOF_{nh}), the asymmetric genotype kernel model under IS (DOF_{as}) and the naive genotype kernel model under IS (DOF_{ns}). The DOF calculations of haplotype densities and the genotype densities are denoted as DOF_h and DOF_r.

We selected a data set from the simulation study of chapter 3. Recall we generated paired haplotypes from two ancestor haplotype sequences with a certain mutation rate. The chosen set was of 50 genotypes (100 haplotypes) and 32 sites. All six DOF were calculated for this set with tuning parameter, $p = .1, .11, \ldots, .48, .49$. The scatter plots of DOFs versus tuning parameter are in Figure 4.1. These show that the DOFs of the naive genotype kernel has a minimum around p=.36-.37, which is the upper bound for its tuning parameter in terms of identifiability. We will not use the naive kernel in any further analysis.

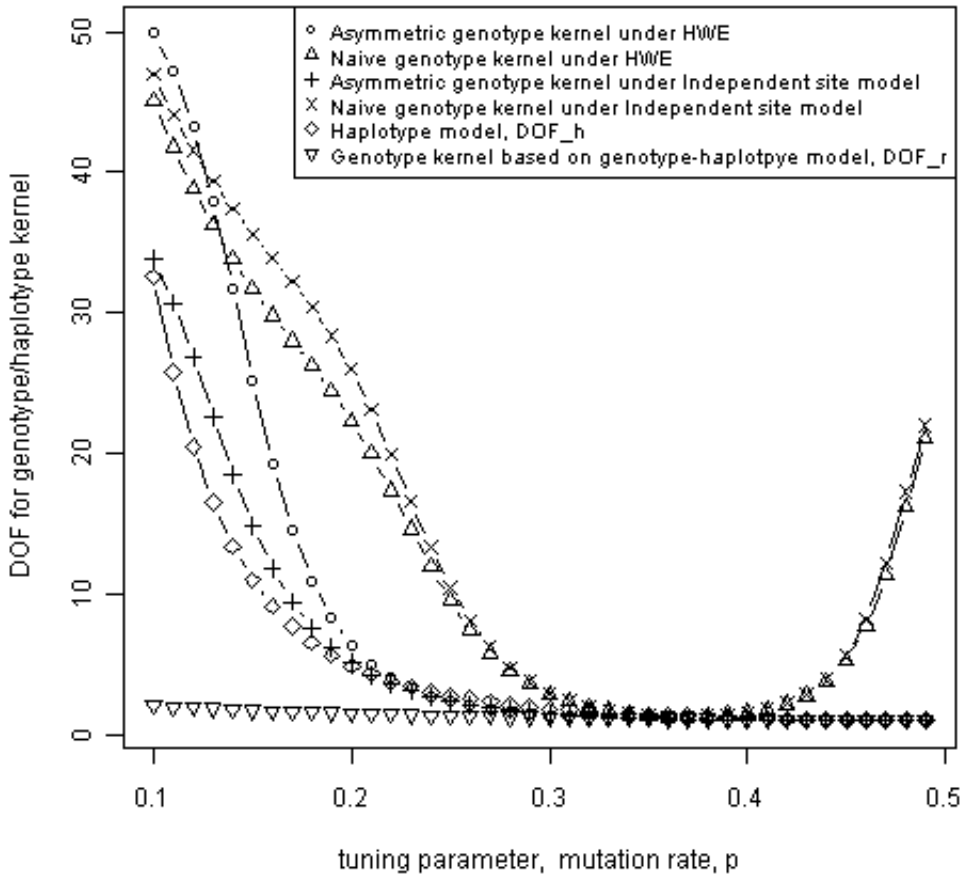


Figure 4.1: DOFs of different kernels for a simulated set of $L = 32$ and $n = 100$ genotypes (200 haplotypes)

ρ when p=.1	DOF_{ah}	DOF_{as}	DOF_h	DOF_r	err_{site} of wMEM with neighbor starting value
DOF_{ah}	-	0.92	*0.87*	0.76	0.48
DOF_{as}	0.92	-	*0.87*	0.76	0.57
DOF_h	0.87	0.87	-	0.66	0.6
DOF_r	0.76	0.76	*0.66*	-	0.28
err_{site} of wMEM with neighbor starting value	0.48	0.57	*0.6*	0.28	-

Table 4.4: Spearman's ρ between DOFs and haplotype errors for the tuning parameter $p = .1$. The correlations with DOF_h are highlighted.

4.3.2 Correlations between $DOF_\bullet$ across data sets

We used the first 4,000 SNPs on chromosome 1 for the Yoruba people (from Nigeria, Africa) in HapMap release 2. The data was the longer version of the data set we used in Chapter 3. The 4,000 sites were divided into 80 subsequences each with the length of 50 sites.

We used the 80 subsequences of real data to examine the correlation of DOF_{ah}, DOF_{as}, DOF_h and DOF_r. We chose $p = .1$ and .01 to calculate DOFs for the 80 sets and then calculate the pairwise correlation between the DOF_s and/or haplotyping error (err_{site}) using Spearman's ρ. The results are in Table 4.4 for p=.1 and in Table 4.5 for p=.01.

The high correlation among DOF's (except DOF_r) suggest we may use DOF_{ah} or DOF_{as} instead of DOF_h to diagnose the haplotyping error. DOF_{ah} or DOF_{as} can be directly calculated based on the genotype data while DOF_h requires the phased haplotype information. The correlations between DOF_{ah}, DOF_{as} and err_{site} for p=.01 are higher than for p=.1. So we recommend to choose the tuning parameter p=.01 when using DOF_{ah} or DOF_{as}.

ρ when p=.01	DOF_{ah}	DOF_{as}	DOF_h	DOF_r	err_{site} of wMEM with neighbor starting value
DOF_{ah}	-	0.99	*0.96*	0.47	0.62
DOF_{as}	0.99	-	*0.96*	0.48	0.63
DOF_h	0.96	0.96	-	0.48	0.67
DOF_r	0.47	0.48	*0.48*	-	0.17
err_{site} of wMEM with neighbor starting value	0.62	0.63	*0.67*	0.17	-

Table 4.5: Spearman's ρ between DOFs and haplotype errors for the tuning parameter $p = .01$. The correlations with DOF_h are highlighted.

4.4 Linkage disequilibrium, haplotyping error and degrees of freedom

The key to haplotyping lies in the dependence structure among the sites (Scheet and Stephens, 2006; Kimmel and Shamir, 2005). In genetic terms, this dependence is called linkage disequilibrium (LD) and/or haplotype diversity (HD). To investigate this question for our methodology, we choose a measure of LD, denoted by D', to run diagnostics on haplotyping errors. We define the frequencies of haplotype 0 and 1 on sites a and b to be p_0, p_1, q_0 and q_1, respectively. In genetics term, the p's and the q's are the allele frequency. The frequencies of haplotype 01, 10, 11, 00 are p_{01}, p_{10}, p_{11} and p_{00}. The D term is defined as,

$$D = p_{11} - p_1 \times q_1.$$

The D' term is then defined as

$$D' = \begin{cases} \frac{D}{min\{p_{01}, p_{10}\}} & \text{if } D \geq 0 \\ \frac{D}{max\{-p_{11}, -p_{00}\}} & \text{if } D < 0 \end{cases}$$

4.4.1 The relationship between LD and the haplotyping error

For the 80 subsequences of 50 sites we used in section 4.3, we calculated the pairwise D' for every subsequence. A contour map based on the D' values was then created for each dataset with the darker color corresponding to higher dependence between two sites. In the top row of Figure 4.2, we align the LD contour maps from subsequence 1 to 10. Note that *only* the dependence structure between the sites within the same subsequence are evaluated. Hence one should use the figure to examine the dependence between nearby sites not across subsequences. For each subsequence, we also show in Figure 4.2 the number of ambiguous sites, the number of errors and ratio of the two (err_{site} measured by site) made by wMEM method with the neighbor starting values.

There is a clear correlation between the dark colored blocks in the LD contour maps and increased error rates in the bottom subgraph. If a block shows strong dependence throughout the whole subsequence (eg, set 5 and 6), the error rate from wMEM runs is very low. If the dependence is very weak (eg, set 1 and set 9), the performance of wMEM is very poor. If there is a disruption of dependence (represented by the white/lighter colored blocks) in the subsequence, the error rates are relatively high and within the subsequence, the error rates tend to arise in the regions with weaker dependence (the first half of subsequence 8 and the second half of subsequence 10). In subsequence 4 and subsequence 7, the strength of the dependence structure on the two sides of the disruption may be balanced. Such a pattern poses a large problem for both the choice of starting values (where we suspect the neighbor starting value strategy may only produce random starting values instead of using what is already in the data) and the MEM procedure, for which the identified mode may be far away from the true haplotype.

Base on this result, We hypothesize that one might find a better way of partitioning the 4,000 sites other than the simple partition of 50 sites per set . It would seem that one could partition the original 4,000-site sequence based on the degree of dependence between sites.

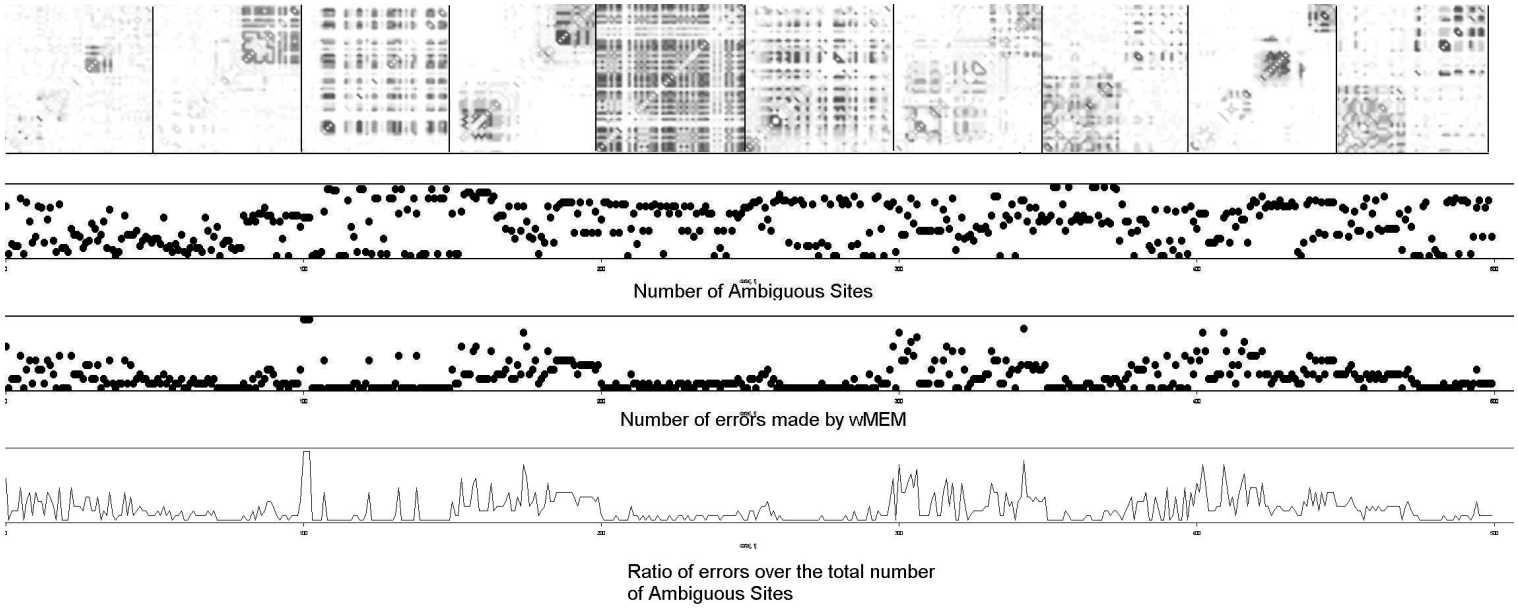


Figure 4.2: An illustration of linkage disequilibrium (D′) vs haplotyping errors using the subsequences 1-10 of the real data. The top row is the aligned contour maps of pairwise D′; the second row shows the number of ambiguous sites per site and the third row shows the number of haplotyping errors made by wMEM using the neighbor starting values. The bottom row is the ratio between errors and the number of ambiguous sites In the top subgraph the boundaries of the subsequences are defined by the vertical straight lines. The order of the subsequences are 1-10 from left to right.

4.4.2 A simple strategy for partitioning long sequences

Recall that we recommend using DOF_{as} or DOF_{ah} to diagnose haplotyping error. This is our algorithm for evaluating the possible breakpoints on a given sequence:

1. We calculate the DOF for a data set of length, L=50 using all the sites; call it DOF_{all}.

2. Then we partition the set into two pieces starting with site 1 and get two subsequences, site 1 and (sites 2-50). And we calculate the DOF for both pieces and denote the sum as DOF_{p1}. Notation wise, $p1$ means **partitioning** at site **1**.

3. Repeat step 2 for site s=1,...,49. Then we plot a horizontal line with the height of DOF_{all} and plot DOF_{ps} against s. The scatter plot is used to compare DOF_{ps} to DOF_{all} at every possible breakpoint on the sequence.

The logic behind these calculations is that the DOF_{all} might be seen as a measurement of overall dependence or diversity. The DOF_{ps} for the possible breakpoints might be seen as a measurement of local dependence in the two pieces. When $DOF_{ps} < DOF_{all}$ for some s a possible interpretation is that if one broke the whole set in two at site s, then the dependence structures within the two resulting subsequences are both stronger than the overall dependence structure. This in turn suggests some independence between the two pieces.

In order to assess this logic, we partitioned sets 1 to 10 at each site 1:49 and calculated DOF_{ps} and DOF_{all} for DOF_{as} for p=.1 and .01 (Figure 4.3). The DOF_{pi} value drops below DOF_{all} for set 3, 6, 8, and 10 when p=.01. Given the fact the haplotyping error rates and DOF_{as} are correlated, we hypothesize that one can find a specific partition of the 4,000 sites on which the performance of wMEM can be improved by using DOF_{as}.

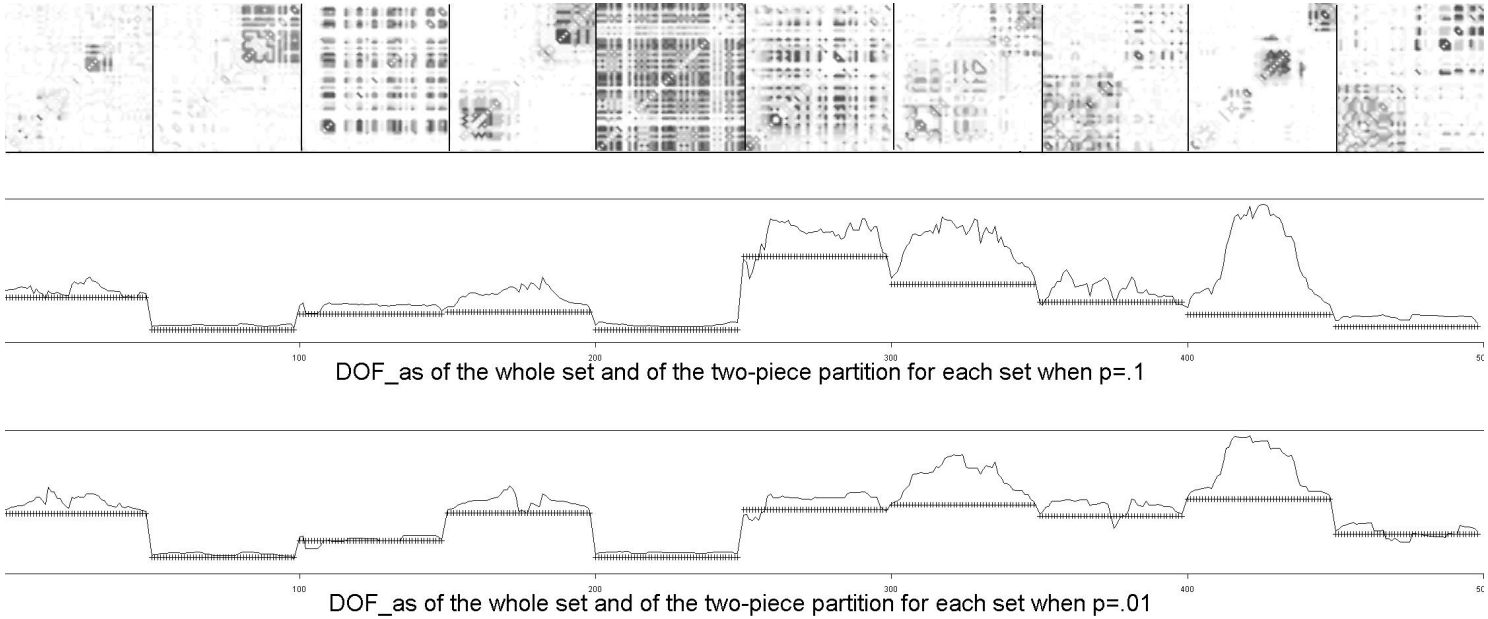


Figure 4.3: Comparison of DOFs and Linkage disequilibrium using subsequences 1-10. The top graph is the pairwise D′contour maps for set 1-10. The subsequences are separated by straight vertical lines. The next two rows are the DOF_{all} (denoted by †) for each of the subsequences and the DOF_{ps} (sum of DOF of two subsequences, the solid line) of DOF_{as} at site i for each subsequence. The graphs are aligned for direct comparison.

Chapter 5

Large scale haplotype inference

In this chapter, we will present a strategy for large scale haplotype inference. The strategy includes partitioning long sequences into smaller subsequences, solving the haplotype inference problem on the subsequences and ligate the solved subsequences to the full length sequences.

5.1 fastPHASE and philosophy of haplotyping

We have briefly surveyed methods for haplotyping in Chapter 1. Researchers have recently developed some algorithms that are suitable for large scale haplotyping. The fastPHASE program (Scheet and Stephens, 2006) is one of most popular haplotyping tools. The method is based on a mixture model for haplotypes (or the frequencies of 0 or 1 on each site of haplotypes). Individuals are the pairs (mixtures) of haplotypes. A strategy called **local clustering** is used to account for the dependence structure among sites. The actual haplotyping is done by first fitting the mixture model, obtaining the components and then resampling from the components to construct pseudosamples of genotypes (with known phased haplotype). Each genotype is then compared against the pseudosamples and the most probable haplotype pairs is chosen for the given genotype. Another popular haplotyping method, GERBIL (Kimmel and Shamir, 2005) uses the same idea of resampling with a different strategy of

generating **the components**, which are called **haplotype matrices** in its case. The haplotype matrices are generated by first identifying recombination-poor regions, **blocks** and by generating **core haplotypes** based on the dependence structure among the sites within each region.

There are two lessons we have learned from these available methods. First, modeling switch / recombination seems to be an integral part of any good haplotyping method. Some methods achieve this by investigating the dependence structure of the data, partitioning the data and solving each piece individually. Secondly, recent methods like fastPHASE and GERBIL seems to rely on a resampling scheme which creates as a pool of genotypes with known haplotype pair for haplotyping. This is a necessary step of re-attaching the solutions of individual pieces together.

Our model is nonparametric and so it should carry more flexibility than the most of the available parametric models. It has the basic structure of a density estimator which is roughly equivalent to the **local** clustering algorithm of fastPHASE. Our method may invoke great computational cost for very long sequences, which leads to developing a partitioning scheme. Later we will also propose a MEM type method for reattaching haplotype pair from adjacent partitioned pieces.

In this chapter, we will utilize the findings in Chapter 3 and Chapter 4 to further develop a strategy for large-scale haplotyping. In Chapter 3, we have shown the general form of the MEM algorithm for haplotype inference. In Chapter 4, we have shown the DOF calculation of a genotype density can be used to diagnose the haplotyping errors. Therefore our overall strategy is to start with genotype data, calculate DOF based on genotypes, partition the data, solve haplotyping problem in the partitioned subsequences and merge the solved subsequences.

5.2 A scheme for large scale haplotyping

To apply our model to the large scale haplotyping problem, we need to solve three problems, the multiple starting value problem, the partition of long sequences problem and the ligating the subsequences problem. In Chapter 3, we had shown different strategies of choosing starting values. In this section, we will address the partition problem and the ligation problem.

5.2.1 Partitioning a large set

Researchers have used the strategy of partitioning long length genotype data to facilitate haplotyping (Zhang et. al., 2002 and Kimmel and Shamir, 2005). The idea of partitioning is to break the full sequence up into subsequences in order to carry out analysis on the shorter subsequences. Different approaches have been developed to partitioning long sequences; for example, minimum description length (Rissanen, 1978) was used in GERBIL and a dynamic programming algorithm was proposed by Zhang et. al. (2002). Our method was inspired by the dynamic programming algorithm. It had some resemblance to sequence alignment. We will discuss the new algorithm and its simplified version.

We first create a pairwise matrix $M = \{m_{st}\}$ for the sites, that is, the columns and rows of the matrix corresponds to the site indices, s and t. Each cell m_{st} represents the DOF calculation for the subsequences from site i to site j, left inclusive. A naive strategy to partition the data would be to compute all the m_{st} for $s < t$. Then find the path that minimizes the sum of m_{st} where s and t are the connecting points. A path is then defined as a list of site indices corresponding to segment endpoints that starts with site 1 and ends with site L,

$$P = \{c_1, ..., c_l\} \tag{5.1}$$

where $c_1 = 1$ and c_l=L +1. The subsequences will then be (c_1;c_2), ..., (c_{l-1};c_l). The c's will be

called the connecting points, meaning the points connecting the path. The connecting points define the boundaries of the partitioned subsequences. For instance, the first subsequence starts at site $c_1 = 1$ and ends at site $c_2 - 1$. For a given path, one may calculate the sum of DOFs between adjacent connecting points,

$$DOF_{path} = \sum_{i=1}^{l-1} DOF_{c_i;c_{i+1}}. \tag{5.2}$$

A naive way to find the path with minimum DOF_{path} is to enumerate all the possible paths but it is too computationally costly. So one may use some heuristic such as limiting the size of the subsequences to be less than some small number, say, 30. Our proposed method of finding the path with the minimum value of (5.2) is as follows:

1. We first find the leftmost segment, with $c_1 = 1$. We start with site 1 and move towards site 2, in order to evaluate the segment 1;3. To determine whether we should partition between site 1 and 2, we calculate $DOF_{1;3}$, $DOF_{2;3}$ and $DOF_{1;2}$. If $DOF_{1;2} + DOF_{2;3}$ is smaller than $DOF_{1;3}$, $c_2 = 2$ and we start with site 2. If not, we move to the next step.

2. Suppose now we start with site 1 and move towards site 3, in order to evaluate segment 1;4. In this step, we will set $c_2 = 1$ if $DOF_{1;2} + DOF_{2;4} < DOF_{1;4}$; if that fails, we set $c_2 = 3$ if if $DOF_{1;3} + DOF_{3;4} < DOF_{1;4}$. If both fail, we move on to examine segment 1;5 for a breakpoint.

3. In general, if we have left endpoint, say $c_m = i$, and have examined $i; j$ for breakpoints and have failed, we move to $i; j + 1$and proceed as follows. we start at site i and move right towards site j. First we calculate $DOF_{i;k}$, $DOF_{k;j+1}$ and $DOF_{i;j+1}$, where $i < k < j + 1$ for every k. Then we compare $DOF_{i;k}$+$DOF_{k;j+1}$ with $DOF_{i:j+1}$ for k from i+1 to j. We partition between $k - 1$ and k (set $c_{m+1} = k$) for the smallest k with $DOF_{i:k}$+$DOF_{k:j}$ <$DOF_{i:j}$. If no partition can be made, we examine $i; j + 1$ for a breakpoint.

We have also developed a simplified version, which saves time on evaluating $DOF_{i:k}$, $DOF_{k+1:j}$ and $DOF_{i:j}$. We have made the following observations,

- If segment $i; j+1$ and the site $j+1$ are highly dependent, the DOF values, $DOF_{i;j+1}$ and $DOF_{i;j+2}$ are similar.
- If only some of the sites in segment sites $i; j+1$ and site $j+1$ are rather dependent, the DOF values, $DOF_{i;j+1}$ and $DOF_{i;j+2}$ are different. If the right end of segment $i; j+1$ and site $j+1$ are highly dependent, a partition somewhere in the middle of $i; j+1$ is warranted.
- DOF values are always larger or equal to 1. Since we decided to calculate DOF under relatively small tuning parameters, $DOF > 1$ always holds.

So in order to have $DOF_{i;k}$+$DOF_{k;j+1}$ <$DOF_{i;j+1}$ at some k, we need to make sure $DOF_{i;j+1}$ is very different from $DOF_{i;j}$. Thus we design the following simplified version.

- Given a left endpoint at site i, we start calculating $DOF_{i;j+1}$, $DOF_{i;j+2}$,... . If at some step j, we find

$$|DOF_{i:j} - DOF_{i:j-1}| > T \tag{5.3}$$

  where T is selected by user (we choose T to be 1.5), we do step 3. Otherwise, we continue to calculate $DOF_{i+1;j+1}$, $DOF_{i+1;j+2}$,... .

This is a very fast algorithm due to the heuristic in eq (5.3). One can think of eq (5.3) as a magnifying glass. With a large T, the algorithm skips a lot of sites so that there no endpoints will be made among them. The resolution can be poor but one certainly gains speed. To partition the 4000-site big set (L=4,000 and n (genotype) =60) by DOF_{as} (from Chapter 4), the original algorithm gave a 357-subsequence partition and the simplified version had 325 subsequences. On a single node server (Sun SunFire v20z) with 2 AMD Opteron 250 2.4 GHz Processors and 8 GB memory, the original algorithm has a run time of 40 minutes on the 4,000-site set and the simplified version takes 7 minutes. With a brief examination,

the partition by the original algorithm and the partition by the simplified version have 70% of their breakpoints in common.

5.2.2 Ligation

Suppose that one solves for the haplotype pairs in each of the subsequences of a partition. We define $(h^{<k>}, \bar{h}^{<k>})$ to be the estimated haplotype pair from the kth subsequence, where $\bar{h}^{<k>}$ is the complementary haplotype to $h^{<k>}$. If one wishes to create a full haplotype sequence from these pieces, one immediately encounters the labeling problem: which one of the haplotype pair should be used in the sequence?

To merge neighboring subsequences, one might use the following two strategies. The first one is called the 'vote with overlap' method. Firstly, one partitions full sequences into m small subsequences by (5.1),

$$P = \{c_1, ..., c_l\}.$$

One could then use the MEM method to analyze overlapping subsequences. If one obtains a haplotype for $(c_1; c_4)$ then one has haplotypes also for also for segments $(c_1; c_2)$, $(c_2; c_3)$, $(c_3; c_4)$. If we continue on to haplotype $(c_2; c_5)$, (c_3, c_6), and so on, one has three sets of inferred haplotype pairs for $h_{c_3;c_4}$, one each from $c_1; c_4$, $c_2; c_5$ and $c_3; c_6$. One could then use the three candidate sets to vote for the haplotype found in sites $c_3; c_4$. The **winning** set of inferred haplotype pairs with majority votes is selected and then merged with the winning sets for other subsequences.

We think of the 'vote with overlap' method to be less efficient compared to the second method (to be described below) because this algorithm requires resolving most of the subsequences three times. It also requires solving longer subsequences ($c_1; c_4$ at a time as opposed to $c_1; c_2$ at a time). Moreover, we have an extended algorithm from our density model for haplotyping in (3.5) of Chapter 3. The basic assumption is although we partition the large sequences into shorter subsequences in such a way that the dependence between subsequences

is small, there is some remnant signature of dependence between neighboring subsequences. Notice that the ligation of subsequences is performed individual by individual. We therefore propose the second algorithm as follows.

First we rewrite $(h^{<k>}, \bar{h}^{<k>})$ as $(h_{c_k;c_{k+1}}, \bar{h}_{c_k;c_{k+1}})$, the estimated haplotype pair from the kth subsequences. We define the genotype for individual k in the subsequences, $c_i; c'_{i+1}$s to be $x_{k,c_i;c_{i+1}}$. We proceed from left to right, and start by picking one of $h_{c_k;c_{k+1}}$ and $\bar{h}_{c_k;c_{k+1}}$. Now assume we have constructed a haplotype $h^*_{c_1;c_a}$ for subsequences $c_1; c_a$.

Let us discuss the labeling problem here. For simplicity, assume we want to merge two subsequences. The first subsequence has the resolved haplotype pair , a and b ($b = \bar{a} = x_1 - a$) for genotype x_1. And the second sequence has the resolved haplotype pair, c and d ($d = \bar{c} = x_2 - c$) for genotype x_2. If $a = b$ or $c = d$ or both, one should have no trouble merging these two sequences because the choice of the merged haplotype pair $\big((a, c), (b, d)\big)$ is same as $\big((a, d), (b, c)\big)$. The labeling problem arises when $a \neq b$ and $c \neq d$. In that case, $\big((a, c), (b, d)\big)$ is different from $\big((a, d), (b, c)\big)$. Recall that $a \neq b$ is equivalent to $'$at least one ambiguous sites on x'_1. So we can use the number of ambiguous sites on the subsequences to determine whether extra care should be taken to address the labeling problem. The ligation algorithm works as follow.

- If individual k has no ambiguous sites on subsequence $c_a; c_{a+1}$, then there is no labeling ambiguity in adding $h_{c_a;c_{a+1}} = \bar{h}_{c_a;c_{a+1}}$ to the current haplotype $h^*_{c_1;c_a}$.

- If there are ambiguous sites on subsequence $c_a; c_{a+1}$ for individual k, we have the labeling issue; one needs to determine which one of $h_{c_a;c_{a+1}}$ and $\bar{h}_{c_a;c_{a+1}}$ should be attached to $h^*_{c_1;c_a}$. First we count the number of the ambiguous sites in every $c_i; c_{i+1}$ subsequence for $i = 1, ..., a-1$ in the $c_1; c_a$ subsequence for individual k. We locate the subsequence, $c_{j*}; c_{j*+1}$, the nearest subsequence to the left of $c_i; c_{i+1}$ that contains ambiguous site(s) for individual k. Then we merge one of the solved haplotype pairs in $c_a; c_{a+1}$ with $h^*_{c_1;c_a}$ for individual k by directly ligating the one of the solved haplotype pairs in $c_a; c_{a+1}$ to $h^*_{c_{j*};c_{j*+1}}$. We call the combination of $c_a; c_{a+1}$ and $c_{j*}; c_{j*+1}$ to be the

target subsequence. Subsequently, we have the genotype of the target subsequence, $x_{k,t} = (x_{k,c_{j*};c_{j*+1}}, x_{k,c_a;c_{a+1}})$. There are two possible combinations of haplotype pairs from $c_{j*}; c_{j*+1}$ and $c_a; c_{a+1}$, which gives rise to the labeling problem. We denote the two possible ligations as

$$\begin{aligned}(h_{t,1} \quad, \bar{h}_{t,1}) = \quad & ((h^*_{c_{j*};c_{j*+1}}, h_{c_a;c_{a+1}}), (\bar{h}^*_{c_{j*};c_{j*+1}}, \bar{h}_{c_a;c_{a+1}})) \\ (h_{t,2} \quad, \bar{h}_{t,2}) = \quad & ((h^*_{c_{j*};c_{j*+1}}, \bar{h}_{c_a;c_{a+1}}), (\bar{h}^*_{c_{j*};c_{j*+1}}, h_{c_a;c_{a+1}}))\end{aligned}$$

We use the objective function of haplotype inference, (3.6) of Chapter 3 to compute value of the objective function for two alternatives:

$$n^{-2} \sum_m \sum_n \prod_{t \in (c_{j*};c_{j*+1},c_a;c_{a+1})} k_4(h_{t,1}(s), x_{m,t}(s)) k_4(\bar{h}_{t,1}(s), x_{n,t}(s))$$

and

$$n^{-2} \sum_m \sum_n \prod_{s \in (c_{j*};c_{j*+1},c_a;c_{a+1})} k_4(h_{t,2}(s), x_{m,t}(s)) k_4(\bar{h}_{t,2}(s), x_{n,t}(s))$$

We then pick the higher one as our solution. Since we have fixed the labeling of the haplotype pair in $c_1; c_a$ by the bookkeeping on $h^*_{c_1;c_a}$, we update $h^*_{c_1;c_a}$ to $h^*_{c_1;c_{a+1}}$ based on the ligation between $c_{j*}; c_{j*+1}$ and $c_a; c_{a+1}$. We than move to the next subsequence.

However, we realize that if there are no ambiguous sites present in the subsequence $c_{a-1}; c_a$, it may not be the best solution to directly ligate subsequence $c_a; c_{a+1}$ to the subsequence, $c_{j*}; c_{j*+1}$, which is the one closest to $c_a; c_{a+1}$ in $c_1; c_a$ among those that contains ambiguous sites for individual k while skipping all the in-between subsequences, $c_{j*+1}; c_{j*+2}$,... , $c_{a-1}; c_a$. Improvement may be done by first identifying $c_j; c_{j+1}$ and ligating subsequence $c_a; c_{a+1}$ to subsequence $c_{j*}; c_a$, where $c_{j*}; c_{a-1}$ is the ligated subsequence of $c_{j*}; c_{j*+1}$,... , $c_{a-1}; c_a$.

5.2.3 Results for large-scale haplotyping

5.2.3.1 Partitioning long sequences and analysis on the subsequences

We applied various wMEM algorithms with different starting values (with the feature of selecting the highest modes) to analyze the 325 partitioned subsequences from section 5.2.1. We use two error measurements to assess the performance of the method. The first one is the total error, err_{site}, introduced in Chapter 3. The second one is the switch error, err_{switch}, which equals the ratio of the minimum number of switches required to recover the true haplotypes to the total number of non-trivial heterozygous sites. In this chapter, we use the numbers of error instead of the ratio. The number of total errors and switch errors are summarized in Table 5.1. We also partition the 4,000 sites into 333 subsequences of 12 sites (excluding the last 4 sites). Here we call the 333 subsequences, the arbitrary partition and call the 325-subsequence partition, the informative partition. Note that these errors are pre-ligation errors.

The comparison of the number of pre-ligation errors suggests that algorithms with more choices of starting values perform better. It also suggests one should consider using a smaller tuning parameter when one uses the multiple starting value strategy. We also see the effect of partitioning by comparing the results of the informative partition and the arbitrary partition. For all the wMEM algorithms applied to both the informative partition and the arbitrary partition, the number of errors made in the informative partition are smaller than those made in the arbitrary partition. The improvement of wMEM algorithms on the informative partition partially justifies the idea of partitioning in large scale haplotyping.

5.2.3.2 Merging the subsequences

We merged the results of the subsequences in the informative partition using the method described in section 5.2.2. Here we used switch errors as the sole indicator of performance because the total error rate gets close to .5 for long sequences due to accumulated switches.

		tuning parameter, p=.1		.4	
		err_{site}	err_{switch}	err_{site}	err_{switch}
a. The 325 subsequences of the informative partition	1a. wMEM with the neighbor starting value	3245	3306	3705	3844
	2a. wMEM with missing data based starting values	1783	1809	1897	1936
	3a. wMEM of neighbor and missing data based starting value	1631	1648	1788	1846
	4a. wMEM with neighbor, missing data based, trivial individual based and naive starting values	*1212*	*1290*	*1435*	*1470*
b. The 333 subsequences of 12 sites	1b. wMEM with the neighbor starting value	3556	3598	4092	4176
	4b. wMEM with the neighbor starting value, missing data based , trivial individual based and naive starting values	1954	2033	2318	2347

Table 5.1: The number of total error err_s and switch error se of different wMEM algorithms with different starting values and with different tuning parameters.Best solutions in each column not using the true values as start are highlighted.

Computationally, a recombination event is equivalent to a switch. In all the ligation runs, we chose p=.1 as the tuning parameter. We merged the true data of the 325 subsequences. The merged results are now considered the haplotyping results for the entire set of 4,000 sites. The number of switch errors are summarized in Table 5.2. Recall that our overall strategy of large scale haplotyping is partition and ligation. The results about the superiority of the informative partition over the arbitrary partition in Table 5.1 only partially justify the use of the informative partition. To fully assess the improvement on the haplotyping accuracy by using the informative partition, we need to compare the switch errors of the merged informative and the merged arbitrary partition. We also included the number of switch errors made by fastPHASE when the program was applied to the entire 4,000-site data set.

The number of switch errors in the merged result of the informative partition was still smaller than that of the arbitrary partition, which indicates the informative partition is advantageous to the overall strategy of haplotype inference. However, the number of switch errors made by fastPHASE was smaller than our overall strategy; we believe the Markov Chain assumption of the fastPHASE model is the main cause.

Although our method underperforms fastPHASE, the partition-ligation strategy presented here can still be used as a stepping stone towards a faster and more accurate tool for haplotype inference. In addition, the partition algorithm can be used separately to facilitate association studies by grouping genetic markers into segments (subsequences) and analyzing the subsequences instead of the genetic markers.

		Number of switch errors
	fastPHASE	2,398
The informative partition	ligation of true subsequences	1,712
	ligation of the wMEM results with multiple starting values (p=.1 for wMEM)	3,172
The arbitrary partition	ligation of true subsequences	1,767
	ligation of the wMEM results with multiple starting values (p=.1 for wMEM)	3,668

Table 5.2: Number of switch errors by fastPHASE and by the ligation algorithm for different partitions.

Chapter 6

Recombination: a brute force model

In previous chapters, I have presented EM type methods for clustering binary sequences and inferring haplotypes from genotypic data. In some real applications, the computational time would be increased drastically in a high dimensional dataset compared to the ones in the examples. One of the solutions may be to divide the sequences of long length L into small pieces and work on each small piece individually. One must then reconstruct the long sequence from the short ones. Such an idea is very well related to the abundant existence of recombination in humans. Each recombination breaks a long sequence into two shorter strings independent of each other. We consider here how one can model recombinations in a manner similar to mutation. One may consider in its full complexity to be a "brute force" model as the likelihood for a single observation requires a calculation summed over all 2^{L-1} possible partitions of the sequence into subsequences. Our methodology will cut the summations to $\begin{pmatrix} L \\ 2 \end{pmatrix}$, the number of possible subsequences originating from the full length sequence. In other words, the 2^{L-1} possible partitions generates many subsequences but the number of the unique subsequences is $\begin{pmatrix} L \\ 2 \end{pmatrix}$. The $\begin{pmatrix} L \\ 2 \end{pmatrix}$ calculation means we focus on the subsequences rather than the partitions.

The density model we will use for haplotype sequences will include mutation as well as

recombination. It is based on the current population that arises from ancestral haplotypes having undergone a prescribed level of mutation and recombination.

Recombination can be estimated based on unrelated individuals. This type of recombination could be called *archaic*. *Archaic* recombinations are not often studied by researchers compared to the recombinations found in familial relationships such as the HapMap project.

6.1 Introduction

The recombination problem can be viewed as a missing data problem where the hidden data is the switching of states between the ancestral sequences. The hidden parameters of interest are the locations of the recombinations. We start by extending the ancestral mixture model to include a Markov mechanism for recombination. The following is a simple continuous time Markov Chain that will generate sequences with the desired structure.

We start with a distribution Q that describes the frequencies of ancestral sequences, with the possible ancestral sequences $\phi_1, ..., \phi_K$, and associated probabilities $\pi_1, ..., \pi_K$. Later we will associate Q with a density $\pi(\phi)$ on the space of binary sequences. We suppose that randomly chosen sequence X from the population at time η consists of strings of subsequences from the ancestors, where the changes between ancestors represent recombination points. (We ignore mutational change for the moment.) As time passes, the number of recombinations increase, resulting in more different ancestral segments co-existing in the same sequence. The distribution of the ancestors comes from a Markov Chain moving left to right along the sequence. First, let $A(1) = k$ mean that the ancestor at site 1 has label k, so the first term in the sequence x is $\mu_k(1)$. We suppose $A(1)$ is generated from distribution Q.

The ancestor $A(2)$ is then generated conditionally on using $A(1)$ the following continuous

time Markov Chain. If we use the $K \times K$ rate matrix

$$R = \begin{bmatrix} \pi_1 - 1 & \pi_2 & ... & \pi_K \\ \pi_1 & \pi_2 - 1 & ... & \pi_K \\ : & : & : & : \\ \pi_1 & \pi_2 & ... & \pi_K - 1 \end{bmatrix} \tag{6.1}$$

then the transition matrix at time η is $P_\eta = I + (1 - e^{-\eta})R$, where I is the matrix identity, and the stationary distribution for the Markov Chain is π. The ancestral state $A(2)$ at site s=2 will be generated from $A(1) = k$ using this transition probability matrix at a fixed time η. We next generate $A(3)$, the ancestral state at site 3 using the same transition matrix, conditional on state $A(2)$. We continue in this manner, generating $A(s)$, s=1 to L.

If we rewrite $P_\eta = q$I+(1-q)(I+R), for q=$e^{-\eta}$, we get an alternative representation of this process as a mixture of a "stay at home process", with probability q and a "make an iid move" process, with probability 1-q. The transitions from this process can be simulated as follows: Generate a Bernoulli trial H_2, a recombination indicator, such that

$$P(H_2 = 0) = e^{-\eta} = 1 - P(H_2 = 1). \tag{6.2}$$

If $H_2 = 0$, then no recombination occurs between sites 1 and 2, and the state $A(2)$ will be the same as $A(1)$. If $H_2 = 1$, a recombination occurs, and we then select ancestor m from the full set $\{1, 2, ..., K\}$ with probability π_m. Notice that $A(2) = A(1)$ could happen by random draw in this step, which we would interpret as meaning that a recombination did occur, but the ancestors involved were identical.

Next, continue to site 3, where one could generate $A(3)$ from $A(2)$ using an independent recombination indicator H_3 just as $A(2)$ was generated from $A(1)$ using H_2, again with density (6.2). Alternatively, if the recombination distance between sites 1 and 2 is known to be different than between 2 and 3, one could change the rate parameter η to $\alpha_2\eta$ as

appropriate. One possibility is to $\alpha_2\eta$ is to set α_2 proportional to the inter-site distances. Continue in this fashion to the end of the sequence. If we define $H_1 = 1$, then we can say that $H_k = 1$ if and only if a new ancestor is chosen at site k. When done, we have generated a sequence

$$H_1, A(1), H_2, A(2), ..., H_L, A(L)$$

that indicates where the recombinations occurred as well as the ancestral identity of each site.

In summary, the resulting data structure for the hidden process $H_1, ..., H_L$ is a binary sequence with $H_i = 1$ indicating that a recombination occurred between site $i-1$ and site i. For now we assume that the underlying observed data are haplotype sequences. In addition, the mutation kernel will be put to use.

From this point, one could compute the probability of a particular x sequence (x_1,...,x_L) as

$$P(X = x) = \sum_{\underline{h}} P(X = x|\underline{H} = \underline{h})P(\underline{H} = \underline{h})$$

where the summation over $\underline{h} = (h_2, ..., h_L)$ requires 2^{L-1} summands to be calculated. Clearly this problem will require calculation shortcuts for L of any magnitude. In addition, the problem of calculating $P(X = x|\underline{H} = \underline{h})$ is itself quite messy. In the following section we devise further notation to help us express the calculation.

6.2 Calculating the density

We start this section by changing notation somewhat. In the preceding section we referred to ancestors A(1),...,A(K) by their labels 1 to K. In this section we will identify the k-th ancestor with its corresponding binary sequence ϕ_k, and we will also associate Q with a density $\pi(\phi)$ on the space of binary sequences.

For bookkeeping purposes, we will let $u : v$ denote the integers from u to v inclusive and

we will let $u;v = u:v-1$. We specify that $H_{L+1} = H_1 = 1$ for notational reasons. Given a random H sequence of $(H_2, ..., H_L)$, we let the *number of runs*, $J = J(H)$, be the number of $1's$ in $H_{1:L} = (H_1, ..., H_L)$. The number of true recombinations is then $J-1$ because $H_1 = 1$ is not true recombination. Define the indices $R_1, ..., R_{J+1}$ to be the locations of the $1's$ in the sequence $(H_1, ..., H_{L+1})$. Note that $R_1 = 1$ and $R_{J+1} = L+1$ by the definition of H_1 and H_{L+1}.

Furthermore, given the $J-1$ true recombinations, we have J runs each with an ancestor ϕ_k, k=R_1,...,R_J. We can replace (J, $R_{1:J+1}$) by (J, $R_{1:J}$) without losing information.

We will say $u;v$ is a **run** in the sequence $(H_1, ..., H_{L+1})$ if $H_u = 1$ and $H_v = 1$ but $H_a = 0$ for $a \in (u, v)$. This means that there was a recombination at each end of the run and none in between. This in turn implies there was a single common ancestor drawn for **sites** $u;v = (u, u+1, ..., v-1)$. The probability of a run such as $u;v$ can be calculated as

$$\begin{aligned} & \Pr\{(u;v) \text{ is a run in H}\} \\ = \; & \Pr\{H_u = 1\} \times \Pr\{H_{u+1} = 0\} \times ... \times \Pr\{H_{v-1} = 0\} \times \Pr\{H_v = 1\}. \end{aligned}$$

Note that $P(H_t = 1)$ is calculated as in (6.2) except that $P(H_1 = 1) = P(H_{L+1}) = 1$. Setting $P(H_1 = 1)$ and $P(H_{L+1} = 1)$ to be 1 means that these are 'not true recombinations'. The length of a sequence $u;v$, which is $v-u$, will be written as $|u;v|$.

Since H_1 and H_{L+1} are set to equal 1, it is possible to define **full runs** and **end runs**. A **full run** is a run for which $u \neq 1$ and $v \neq L+1$. An **end run** has $u = 1$ and $v = L+1$ or both. The calculation of the run probability for an end run is different than a full run because $H_1 = 1$ and $H_{L+1} = 1$ are not random. The reason of $H_1 = 1$ and $H_{L+1} = 1$ being not random is that we do not have information on the sites to the left of H_1 and the sites on the right of H_{L+1}. As an example, suppose $L = 9$ and

$$H = (H_1, H_2, ..., H_{10}) = (1, 1, 0, 0, 1, 0, 0, 1, 0, 1).$$

where $h_1 = h_{10} = 1$ by default. The actual recombinations occurred at $h_2 = 1$, $h_5 = 1$, $h_8 = 1$ and the number of runs is $J = 4$. Furthermore, the **run** sequence R is

$$R_1 = 1, R_2 = 2, R_3 = 5, R_4 = 10 \text{ and } R_5 = 10.$$

The runs are then (1;2), (2;5), (5;8) and (8;10). These can be written succinctly as $r_k; r_{k+1}$ for $k = 1, 2, 3, 4$.

When a run occurs on $u; v$, then the sequence $A(u; v) = (A(u), ..., A(v-1))$ is constant, so the entire block $u; v$ is descended from the same ancestor, which we have now denoted as Φ_k if u=R_k. Notice that the random variables J and $R_1, ..., R_J$ are equivalent to the vectors H. If the recombination probabilities are a constant q between sites, then using the independence of the H's

$$\Pr\{J = j, r_{1:J} = r_{1:j}\} = q^{j-1}(1-q)^{L-j} \tag{6.3}$$

Further, we can now write the conditional probability of the sequence X, conditional on the run information $\{J = j, R_{1:J} = r_{1:j}\}$ and the realized ancestors. Let $\phi_1, ..., \phi_j$ be the ancestral binary sequences corresponding to ancestors $A(r_1)$, $A(r_2), ..., A(r_j)$. Define

$$M_{u;v}(x, \phi) = \prod_{s=u}^{v-1} m(x(s), \phi(s))$$

be the product mutation kernel for the block $u; v = u : (v-1)$. Then

$$\begin{aligned} \Pr\{X = x | J = j, R_{1:J} = r_{1:j}, \phi_1, ..., \phi_j\} &= M_{r_1;r_2}(x, \phi_1) \times M_{r_2;r_3}(x, \phi_2) \times ... \times M_{r_J;r_J+1}(x, \phi_J) \\ &= \prod_{k=1}^{j} M_{r_k;r_{k+1}}(x, \phi_k) \end{aligned} \tag{6.4}$$

If we let that the ancestral distribution be $\pi(\phi)$, we then can write

$$\Pr\{X = x | J = j, R_{1:J} = r_{1:j}\} = \sum_{\phi_1} \sum_{\cdots} \sum_{\phi_j} \prod_{k=1}^{j} M_{r_k;r_{k+1}}(x, \phi_k)\pi(\phi_1)...\pi(\phi_j). \tag{6.5}$$

We define

$$M_{u;v}(x, \pi) = \sum_{\phi} M_{u;v}(x, \phi_k)\pi(\phi)$$

Finally, we can put the last expression together with $\Pr\{J = j, R_{1:J} = r_{1:j}\}$ in (6.5) to calculate $\Pr\{X = x\}$

$$\Pr\{X = x\} = \sum_{j} \sum_{r_{1:j}} \Pr\{X = x | J = j, R_{1:J} = r_{1:j}\} \Pr\{J = j, R_{1:J} = r_{1:j}\} \tag{6.6}$$

NOTE: it is in (6.6) that one sees the potential for using a simplified approximation when $E(J)$, the expected number of runs J, is small. In these cases $J-1$ is approximately Poisson (for L large), and one could truncate the above sum to only those j that are most likely. This vastly reduces the number of partitions $j, r_{1:j}$ that need to be considered. The choice of an appropriate j may be approached by empirical study examining the relationship between j and L.

We also want to work out the calculation of $Pr_{\pi_0}\{X = x\}$ where π_0 is the starting density for ϕ. First let us introduce the two following lemmas.

Lemma 1. *Let Hardy-Weinberg start of the haplotype density be*

$$\pi_0(\phi) = \prod_{s} \gamma_s^{\phi(s)}(1 - \gamma_s)^{1-\phi(s)}$$

where γ_s is the probability of 1 at site s. We have

$$\sum_{\phi(s)} M(x_i(s), \phi(s))\pi_0(\phi(s)) = \gamma_s m(x, 1) + (1 - \gamma_s)m(x, 0).$$

We then have

$$M_{u:v}(x_i, \pi_0) = \prod_{s=u}^{v-1} \{\gamma_s m(x,1) + (1-\gamma_s) m(x,0)\} \tag{6.7}$$

Formula (6.7) can be shown to be true by exchanging sum and product because sites in $M(x_i, \phi)$ and $\pi_0(\phi)$ are conditionally independent.

Lemma 2. *Given that Lemma 1 holds,*

$$\sum_{\phi} \frac{M(x_i, \phi)\pi_0(\phi)}{M_{1;L+1}(x, \pi_0)} = 1, \tag{6.8}$$

and

$$\sum_{\phi_{u;v}} \frac{M_{u;v}(x_i, \phi)\pi_0(\phi)}{M_{u;v}(x, \pi_0)} = 1 \tag{6.9}$$

Then we have $\Pr\{X = x | J = j, R_{1:J} = r_{1:j}\}$ under π_0 as the calculations in $\phi_1, ..., \phi_j$ are missing the dependence on ϕ that must be summed over. It has been shown in (6.5)

$$\begin{aligned} & \Pr_{\pi_0}\{X = x | J = j, R_{1:J} = r_{1:j}\} \\ = & \sum_{\phi_1,...,\phi_j} \Pr_{\pi_0}\{X = x | J = j, R_{1:J} = r_{1:j}, \phi_1, ..., \phi_j\} \\ = & M_{r_1;r_2}(x, \pi_0) \times M_{r_2;r_3}(x, \pi_0) \times ... \times M_{r_j;r_{j+1}}(x, \pi_0) \\ = & M_{1;L+1}(x, \pi_0) \end{aligned} \tag{6.10}$$

Notice that the right side does not depend on j or $r_{1:j}$.Hence

$$f_0(x) = \Pr_{\pi_0}\{X = x\} \quad = \quad M_{1;L+1}(x, \pi_0) \tag{6.11}$$

6.3 The LEM1 estimator

The recombination problem is not a simple mixture problem, but rather more of a ″product of mixtures″ due to the repeated draws from π needed to generate the data. We here generalize likelihood EM method to this problem by defining the EM algorithm steps through a missing data formulation. While the EM algorithm for this problem is technically very complex, the use of an independence starting value helps immensely. We now start the LEM algorithm with $\pi_0(\phi) = \pi_0(\phi) = \prod_s \gamma_s^{\phi(s)}(1-\gamma_s)^{1-\phi(s)}$, the independence starting density.

6.3.1 The one step EM estimator

Proposition 3. *The one step EM estimator from the independence density is*

$$\begin{aligned}\pi_1(\phi) &= \frac{1}{n}\sum_i \sum_{u<v} \pi_0(\phi) M_{u;v}(x_i,\phi) M_{u;v}^{-1}(x_i,\pi_0) \Pr\{u;v \text{ is a run}\} \times C^{-1} \\ &= \frac{1}{n}\sum_i \sum_{u<v} \frac{M_{u;v}(x_i,\phi)\pi_0(\phi)}{M_{u;v}(x_i,\pi_0)} \Pr\{u;v \text{ is a run}\} \times C^{-1} \end{aligned} \tag{6.12}$$

where

$$C = (L-1)q+1 \tag{6.13}$$

$$M_{u;v}(x_k,\pi_0) = \sum_\phi M_{u;v}(x_i,\phi)\pi_0(\phi) = \prod_{s=u}^{v-1} D_s(x_k(s)), \tag{6.14}$$

$$D_s(x) = \gamma_s m(x,1) + (1-\gamma_s)m(x,0)$$

and

$$\pi_0(\phi_{u;v}) = \prod_{s=u}^{v-1} \gamma_s^{\phi_i(s)}(1-\gamma_s)^{1-\phi_i(s)}.$$

The proof the proposition is in next subsection. Here we show the proof of (6.13).

$$
\begin{aligned}
C &= \sum_{\phi} \frac{1}{n} \sum_{k} \sum_{u<v} \frac{M_{u;v}(x_k, \phi)\pi_0(\phi_{u;v})}{M_{u;v}(x_k, \pi_0)} \Pr\{u; v \text{ is a run}\} \\
&= \frac{1}{n} \sum_{k} \sum_{u<v} \sum_{\phi} \frac{M_{u;v}(x_k, \phi)\pi_0(\phi_{u;v})}{M_{u;v}(x_k, \pi_0)} \Pr\{u; v \text{ is a run}\} \\
&= \frac{1}{n} \sum_{k} \sum_{u<v} \Pr\{u; v \text{ is a run}\} \\
&= \frac{1}{n} \sum_{k} E[\sum_{u<v} \mathbb{I}(u; v \text{ is a run })] \\
&= \frac{1}{n} \sum_{k} E(J) \\
&= (L-1)q + 1
\end{aligned}
$$

We also make the following observations:

- $M_{u;v}(x, \pi_0)$ can be calculated for all $u < v$ and k.

- In assessing the computational challenges via the number of summation terms in the calculation, π_1 is a sum over n data points, but now also one sums over $\binom{L+1}{2}$ pairs $u < v$. Some savings might be obtained by truncating terms where $\Pr\{u; v \text{ is a run}\}$ is small. (i.e., long runs when the recombination rate is high; or short runs when the recombination rate is low)

- Now we can find the modes ϕ^* of $\pi_1(\phi)$ by the MEM algorithm, although the computational cost is possibly rather steep due to the summations. The modal solutions ϕ^* would correspond to the most likely ancestral sequences. In other words, the MEM of π_1 provides the estimate of the ancestral sequences under the assumption of both mutation (τ) and recombination (q or $a_i q$).

- It would of course be more desirable to use the improved estimator π_1 and calculate

$$Pr_{\pi_1}\{u; v \text{ is a run with ancestor } \phi^* | X = x_i\}$$

or other items like $\Pr_{\pi_1}\{H_s = 1|X = x_i\}$, the probability that a recombination occurred at a particular location s in a particular sequence x_i. However when one uses π_1 instead of π_0, the calculations start to involve much larger sums over partitions. Here one will certainly find the calculations feasible only for sequence lengths such that the expected number of recombinations is small. I will save this calculation for later.

6.3.2 Proof of proposition

First we construct a likelihood EM algorithm where the missing data are the $J, R_{1:J}$, and $\phi_{1:j}$. The ϕ_i term represents the labels of the ancestors $A(r_1)$, $A(r_2)$,..., $A(r_j)$. Let

$$\phi_1, \phi_2, ..., \phi_j = \phi_{1:j} \tag{6.15}$$

be the associated block of sites of the $r_k; r_{k+1}$ runs. We start by defining the EM log likelihood

$$E_{J,R_{1:J},\phi_{1:j}|x_1,x_2,...,x_n,\pi_{old}}[\log(f(\phi;\pi_{new}))] \tag{6.16}$$

$$= \sum_{J,R_{1:J},\phi_{1:j}} \frac{1}{n} \sum_{i=1}^{n} \log(\pi_{new}(\phi_1) \times ... \times \pi_{new}(\phi_j)) \times \Pr_{\pi_{old}}(J = j, R_{1:J} = r_{1:j}, \phi_{1:j}|X = x_i)$$

Here $\log(f(\phi;\pi_{new}))$ is the complete-data log likelihood and π_{old} is the initial density, the independence density in (6.11). The EM log likelihood function becomes

$$\sum_{J,R_{1:J},\phi_{1:j}} \frac{1}{n} \sum_{i=1}^{n} \{\log \pi_1(\phi_1) + ... + \log \pi_1(\phi_j)\} \Pr_{\pi_0}(J = j, R_{1:J} = r_{1:j}, \phi_{1:j}|X = x_i)$$

Given a fixed j , $r_{1:j}$ and x_i, there is an associated summand in the EM log likelihood

$$\frac{1}{n} \Pr_{\pi_0}\{J = j, R_{1:J} = r_{1:j}, \phi_{1:j}|X = x_i) \left[\log(\pi_0(\phi_1)) + ... \log(\pi_0(\phi_j)\right]$$

where

$$\begin{aligned}
&\Pr_{\pi_0}\{J=j, R_{1:J}=r_{1:j}, \phi_{1:j}|X=x_i) \\
= \;& \frac{\Pr_{\pi_0}\{J=j, R_{1:J}=r_{1:j}, \phi_{1:j}, X=x_i)}{\Pr_{\pi_0}(X=x_i)} \\
= \;& f_0^{-1}(x_i)\left[\prod_{k=1}^{j} M_{r_k;r_{k+1}}(x_i,\phi_k)\pi_0(\phi_k)\right]\Pr\{J=j, R_{1:J}=r_{1:j}\}
\end{aligned}$$

and $f_0(x_i) = \Pr_{\pi_0}(X=x_i) = \prod_s \gamma_s^{x_i(s)}(1-\gamma_s)^{1-xI(s)}$ because of (6.11). If we consider a particular ϕ, say ϕ^o, then the weights associated with $\log(\pi(\phi_1))$ when the first segment, $\phi_1 = \phi^o$ can be summed over the remaining segments, $\phi_2, ..., \phi_j$ to give the term

$$\begin{aligned}
\Pr_{\pi_0}\{J=j, R_{1:J}=r_{1:j}, \phi_1=\phi^o|X=x_i) \;=\; & f_0^{-1}(x_i)M_{r_1;r_2}(x_i,\phi^o)\pi_0(\phi^o) \\
& \times \prod_{k=2}^{j} M_{r_k;r_{k+1}}(x_i,\pi_0)\Pr\{J=j, R_{1:J}=r_{1:j}\}
\end{aligned}$$

In general this EM calculation would be very tedious due to all the terms in the product. What makes it feasible is using the independence starting density for π_0. Then within a given run,

$$M_{r_k;r_{k+1}}(x_i,\pi_0) = \sum_{\phi} M_{r_k;r_{k+1}}(x_i,\phi)\pi_0(\phi).$$

Hence the calculation becomes

$$\begin{aligned}
&\Pr\{J=j, R_{1:J}=r_{1:j}, \Phi_1=\phi|X=x_i\} \\
= \;& f_0^{-1}(x_i)\pi_0(\phi^o)M_{r_1;r_2}(x_i,\phi)M_{r_2;r_3}(x_i,\pi_0)\times ... \times M_{r_j;r_{Jj}+1}(x_i,\pi_0) \\
& \times \Pr\{J=j, R_{1:J}=r_{1:j}\} \qquad (6.17) \\
= \;& f_0^{-1}(x_i)\pi_0(\phi^o)M_{r_1;r_2}(x_i,\phi)M_{r_2;r_{j+1}}(x_i,\pi_0)\Pr\{J=j, R_{1:J}=r_{1:j}\} \\
= \;& \pi_0(\phi^o)M_{r_1;r_2}(x_i,\phi)M^{-1}_{r_1;r_2}(x_i,\pi_0)\Pr\{J=j, R_{1:J}=r_{1:j}\} \qquad (6.18)
\end{aligned}$$

where

$$f_0(x_i) = M_{1;L+1}(x_i, \pi_0)$$

without any further calculation. In a similar way, one can sum up the weights for $\log(\pi(\phi_2))$ when ϕ_2 is ϕ, and get

$$\Pr\{J, r_{1:J}, \Phi_2 = \phi | X = x_i\} = \pi_0(\phi^o) M_{r_2;r_3}(x, \phi) M^{-1}_{r_2;r_3}(x_i, \pi_0) \Pr\{J = j, R_{1:J} = r_{1:j}\}.$$

Now we can see that the weight associated with a particular $\log \pi(\phi^o)$ has to be summed up over the various possible runs where ϕ^o is the ancestor. We do so by focusing on each possible run $u; v$, and summing up the weights belonging to that particular run. It will be

$$\pi_0(\phi) M_{u;v}(x_i, \phi) M^{-1}_{u;v}(x_i, \pi_0) \sum_j \sum_{r_{1:j}} \Pr\{J = j, R_{1:J} = r_{1:j}\} I\{u; v \text{ is a run in } r_{1:j}\}$$

But the latter double summation is just $\Pr\{u; v \text{ is a run}\}$. Now we have the weights

$$w_i(\phi) = \frac{1}{n} \sum_{u<v} \pi_0(\phi) M_{u;v}(x_i, \phi) M^{-1}_{u;v}(x_i, \pi_0) \Pr\{u; v \text{ is a run}\}$$

associated with each $\log \pi(\phi^o)$ and x_i. To maximize, we note that the form of the likelihood is multinomial, so the solution $\pi_1(\phi)$ has the form $C^{-1} \sum_i w_i(\phi)$, where C acts as a normalizing **c**onstant to force the sum over ϕ to be one (Lindsay, 1995).

6.4 1-at-a-time update of π_1

The basic form of π_1 can be represented as a sum of products but the summation is over $u < v$. We need to adapt our EM algorithm to account for $\sum_{u<v}$.

If we wish to maximize $\pi_1(\phi)$ over ϕ, we can pick one site b to optimize over, holding the other sites fixed. The expression $\phi_{<b>}$ means the sites on ϕ other than site b. The relative density (ignoring the constant, C) we wish to maximize over $\phi(b)$ can be written as

$$\pi_1(\phi \text{ given } \phi_{<b>} \text{ fixed }) = n^{-1} \sum_i \sum_{u<v} (\{ \frac{\pi_0(\phi) M_{u;v}(x_i, \phi)}{M_{u;v}(x_i, \pi_0)} \mathrm{I}\{b \in (u;v)\} + \frac{\pi_0(\phi) M_{u;v}(x_i, \phi)}{M_{u;v}(x_i, \pi_0)} \mathrm{I}\{b \notin (u;v)\}\} \Pr\{u; v \text{ is a run}\}) \quad (6.19)$$

Since ϕ_b appears in certain runs $u; v$, one may do the maximization of equation (6.19) by maximizing the following,

$$\pi_1(\phi \text{ given } \phi_{<b>} \text{ fixed }) = n^{-1} \sum_i \sum_{u<v} \frac{\pi_0(\phi) M_{u;v}(x_i, \phi)}{M_{u;v}(x, \pi_0)} I\{b \in (u;v)\} \Pr\{u; v \text{ is a run}\}$$

Therefore

$$\pi_1(\phi|\phi_{<b>}) = n^{-1} \sum_i \frac{\pi_0(\phi_{b;b+1}) M_{b;b+1}(x_i, \phi)}{M_{b;b+1}(x, \pi_0)} \times \{ \sum_{u<v \& b \in (u;v)} \frac{\pi_0(\phi_{1;b}) M_{u;b}(x_i, \phi)}{M_{u;b}(x, \pi_0)} \frac{\pi_0(\phi_{b+1;L+1}) M_{b+1;v}(x_i, \phi)}{M_{b+1;v}(x, \pi_0)} \Pr\{u; v \text{ is a run}\}\} \quad (6.20)$$

The maximizer of $\pi_1(\phi|\phi_{<b>})$ can be found by holding $\phi_{<b>}$ fixed and evaluating (6.20) at $\phi(b) = 1$ and 0.

6.4.1 The modes of π_1 (in a toy example)

We will illustrate how modes of π_1 can be found by the 1-site-a-time strategy. The number of modes found in the recombination model density will then be compared to the number of modes found in genotype-haplotype density. Note that our interest lies in the inference of recombination which is in the next section.

I use the same training dataset as used in chapter 2 ($L = 50$, $n = 52$). I set the recombination rate to be .0001, .001, .002 and .01 in different runs. The τ is set to be .05 with .05 increments for all four runs. The p, q and the corresponding number of modes

The tuning parameter for mutation, p=	Haplotype model from Chapter 2 (no recombination)	Recombination model with the tuning parameter for recombination, q=				
		.00001	.0001	.001	.01	.02
.1	11	11	10	9	7	7
.12	10	10	8	7	5	5
.16	8	8	7	6	4	2
.18	7	7	6	4	3	1
.255	4	4	1	1	1	-
.26	3	3	-	-	-	-
.27	1	1	-	-	-	-

Table 6.1: **The results of the MEM of π_1 of the recombination model under different recombination rates** . Four recombination rates are used, .00001, .0001, .001, .01 and .02. The number of modes inferred by the MEM method of π_1 (the haplotype model) from Chapter 2 is also shown. The τ values for all the runs are .1, .12, .14, .16, .18, .2, .22, .24

computed by the MEM method are summarized in the Table 6.1.

Based on Table 6.1, it seems the recombination model starts to change the clustering patterns from the no-recombination model when the tuning parameter for recombination, q, is larger than .0001. The number of modes decreased when we increased the tuning parameter for recombination.

6.5 Inferring a recombination event using π_1

We now consider how we might infer the location of recombination events.

6.5.1 $Pr\{H_i = 1|X = x_k, J = 1$ or 2$\}$ and Bayes factor K

Let revisit some notation here. The H term is the indicator for recombination events. So if $H_i = 1$, a recombination has occurred between i and $i + 1$ sites with respect to a certain sequence. The J term is the number of runs in the sequence. $J = 1$ means there is no recombination and $J = 2$ means there is one recombination in the sequence. Also notice

that we are interested in H_i with $1 < i \leq L$ because we define $H_1 = 1$.

One could consider calculating

$$\begin{aligned} \Pr_{\pi_1}\{H_i = 1 | X = x_k\} &= \frac{\Pr_{\pi_1}\{H_i = 1, X = x_k\}}{\Pr_{\pi_1}(X = x_k)} \\ &= \frac{\Pr_{\pi_1}(X_{1;i} = x_{k_{1;i}}) Pr\{H_i = 1\} \Pr_{\pi_1}(X_{i;L+1} = x_{k_{i;L+1}})}{\Pr_{\pi_1}(X = x_k)} \end{aligned} \tag{6.21}$$

But the calculation of $\Pr_{\pi_1}(X = x_k)$ and $\Pr_{\pi_1}(X_{u;v} = x_{k_{u;v}})$ is very expensive. So we consider the simpler calculation of $\Pr_{\pi_1}\{H_i = 1 | X = x_k, J = 1 \text{ or } 2\}$. We will be able to answer the question: Given that there was at most one recombination, where was it most likely to have been?

The conditional probability of recombination at site i is

$$\Pr_{\pi_1}\{H_i = 1 | X = x_k, J = 1 \text{ or } 2\} = \frac{\Pr_{\pi_1}\{H_i = 1, X = x_k, J = 1 \text{ or } 2\}}{\Pr_{\pi_1}(X = x_k, J = 1 \text{ or } 2)}. \tag{6.22}$$

Let us focus on the numerator of (6.22) first,

$$\begin{aligned} \Pr_{\pi_1}\{H_i = 1, X = x_k, J = 1 \text{ or } 2\} &= \Pr_{\pi_1}\{H_i = 1, X = x_k, J = 1\} + \Pr_{\pi_1}\{H_i = 1, X = x_k, J = 2\} \\ &= M_{1;i}(x_i; \pi_1) M_{i;L+1}(x_i; \pi_1) q(1-q)^{L-2}. \end{aligned} \tag{6.23}$$

The denominator becomes

$$\begin{aligned} \Pr_{\pi_1}(X = x_k, J = 1 \text{ or } 2) &= \Pr_{\pi_1}(X = x_k, J = 1) + \Pr_{\pi_1}(X = x_k, J = 2) \\ &= M_{1;L+1}(x_i; \pi_1)(1-q)^{L-1} + \sum_i M_{1;i}(x_i; \pi_1) M_{i;L+1}(x_i; \pi_1) q(1-q)^{L-2} \end{aligned}$$

Then we put the numerator and the denominator back together and (6.22) becomes,

$$\begin{aligned}
\Pr_{\pi_1}\{H_i = 1 | X = x_k, J = 1 \text{ or } 2\} &= \frac{\Pr_{\pi_1}\{H_i = 1, X = x_k, J = 1 \text{ or } 2\}}{\Pr_{\pi_1}(X = x_k, J = 1 \text{ or } 2)} \\
&= \frac{M_{1;i}(x_i;\pi_1) M_{i;L+1}(x_i;\pi_1) q (1-q)^{L-2}}{M_{1;L+1}(x_i;\pi_1)(1-q)^{L-1} + \sum_i M_{1;i}(x_i;\pi_1) M_{i;L+1}(x_i;\pi_1) q (1-q)^{L-2}} \\
&= \frac{M_{1;i}(x_i;\pi_1) M_{i;L+1}(x_i;\pi_1) q}{M_{1;L+1}(x_i;\pi_1)(1-q) + \sum_i M_{1;i}(x_i;\pi_1) M_{i;L+1}(x_i;\pi_1) q}
\end{aligned}$$

Note that for comparing two sites, we can use the ratio

$$\frac{\Pr_{\pi_1}\{H_i = 1 | X = x_k, J = 1 \text{ or } 2\}}{\Pr_{\pi_1}\{H_j = 1 | X = x_k, J = 1 \text{ or } 2\}} = \frac{M_{1;i}(x_i;\pi_1) M_{i;L+1}(x_i;\pi_1)}{M_{1;j}(x_i;\pi_1) M_{j;L+1}(x_i;\pi_1)}.$$

One can also estimate

$$Pr\{H_i = 0 | x_k, J = 1 \text{ or } 2\} = 1 - Pr\{H_i = 1 | x_k, J = 1 \text{ or } 2\}$$

Then one might use the odds ratio of

$$\frac{Pr\{H_i = 1 | x_k, J = 1 \text{ or } 2\}}{Pr\{H_i = 0 | x_k, J = 1 \text{ or } 2\}}$$

to infer the most likely location of recombination for the sequence x_k.

Another interesting application will be to calculate the ratio of the probability of exactly one recombination in the sequence x_i versus the probability of no recombination.

$$\frac{Pr\{J = 2 | X = x_k\}}{Pr\{J = 1 | X = x_k\}} = \frac{Pr\{J = 2, X = x_k\}}{Pr\{J = 1, X = x_k\}} = \frac{\sum_i M_{1;i}(x_i;\pi_1) M_{i;L+1}(x_i;\pi_1) q}{M_{1;L+1}(x_i;\pi_1)(1-q)}$$

Bayes factor

The calculation of $Pr\{H_i = 1|X = x_k, J = 1 \text{ or } 2\}$ may be modified to a Bayes factor of recombination at location i for individual k. First, we have

$$\begin{aligned} & \Pr_{\pi_1}\{X = x_k|H_i = 1, J = 1 \text{ or } 2\} \\ = & \frac{\Pr_{\pi_1}\{X = x_k, H_i = 1, J = 1 \text{ or } 2\}}{Pr_{\pi_1}(H_i = 1, J = 1 \text{ or } 2)} \\ = & \frac{M_{1;i}(x_i;\pi_1)M_{i;L+1}(x_i;\pi_1)q(1-q)^{L-2}}{q(1-q)^{L-2}} \\ = & M_{1;i}(x_i;\pi_1)M_{i;L+1}(x_i;\pi_1) \end{aligned}$$

Then we calculate

$$\begin{aligned} & Pr_{\pi_1}\{X = x_k|H_i = 0, J = 1 \text{ or } 2\} \\ = & \frac{\Pr_{\pi_1}\{X = x_k, H_i = 0, J = 1 \text{ or } 2\}}{Pr(H_i = 0, J = 1 \text{ or } 2)} \end{aligned}$$

where

$$\begin{aligned} & \Pr_{\pi_1}\{X = x_k, H_i = 0, J = 1 \text{ or } 2\} \\ = & \Pr_{\pi_1}(X = x_k, J = 1 \text{ or } 2)\Pr_{\pi_1}\{H_i = 0|X = x_k, J = 1 \text{ or } 2\} \\ = & \Pr_{\pi_1}(X = x_k, J = 1 \text{ or } 2)(1 - \Pr_{\pi_1}\{H_i = 1|X = x_k, J = 1 \text{ or } 2\}) \\ = & \Pr_{\pi_1}(X = x_k, J = 1 \text{ or } 2) - \Pr_{\pi_1}\{H_i = 1, X = x_k, J = 1 \text{ or } 2\} \\ = & M_{1;L+1}(x_i;\pi_1)(1-q)^{L-1} + \sum_i M_{1;i}(x_i;\pi_1)M_{i;L+1}(x_i;\pi_1)q(1-q)^{L-2} \\ & -M_{1;i}(x_i;\pi_1)M_{i;L+1}(x_i;\pi_1)q(1-q)^{L-2} \end{aligned} \tag{6.24}$$

Consequentially, the Bayes factor is defined by

$$
\begin{aligned}
K_i &= \frac{Pr_{\pi_1}\{X = x_k | H_i = 1, J = 1 \text{ or } 2\}}{Pr_{\pi_1}\{X = x_k | H_i = 0, J = 1 \text{ or } 2\}} && (6.25)\\
&= \frac{Pr_{\pi_1}\{X = x_k, H_i = 1, J = 1 \text{ or } 2\}}{Pr_{\pi_1}\{X = x_k, H_i = 0, J = 1 \text{ or } 2\}} \times \frac{Pr_{\pi_1}\{H_i = 0, J = 1 \text{ or } 2\}}{Pr_{\pi_1}\{H_i = 1, J = 1 \text{ or } 2\}} \\
&= \frac{Pr_{\pi_1}\{X = x_k, H_i = 1, J = 1 \text{ or } 2\}}{Pr_{\pi_1}\{X = x_k, H_i = 0, J = 1 \text{ or } 2\}} \times \frac{(1-q)^{L-1} + (L-2)q(1-q)^{L-2}}{q(1-q)^{L-2}} && (6.26)\\
&= \frac{Pr_{\pi_1}\{X = x_k, H_i = 1, J = 1 \text{ or } 2\}}{Pr_{\pi_1}\{X = x_k, H_i = 0, J = 1 \text{ or } 2\}} \times \frac{(1-q) + (L-2)q}{q}
\end{aligned}
$$

where $Pr_{\pi_1}\{X = x_k, H_i = 1, J = 1 \text{ or } 2\}$ can computed by (6.24) and $Pr_{\pi_1}\{X = x_k, H_i = 1, J = 1 \text{ or } 2\}$ can be computed by (6.25).

One interpretation of Bayes factor by Jeffreys (Jeffreys, 1961) is

K	Strength of evidence
<1	Negative (No recombination at location i for individual k)
1 to 3	Barely worth mentioning
3 to 10	Substantial evidence for recombination
10 to 30	Strong
30 to 100	Very strong
>100	Decisive

6.6 Infer recombinations: simulation

Recombination, in a retrospective view, is a relative term. Suppose we start with two distinct haplotypes, h_1 and h_2 and the mutation rate is very low. We can call h_1 and h_2 the parental haplotypes because when a recombination occurs, a new haplotype, h_3, which is different from the first two, is created from the parental haplotypes. The sequence h_3 may be called the recombinant and all three of them can be called the ancestral haplotypes at the new point in time. Generation after generation, the ancestral haplotypes are still present in the population with little changes in their sequences but their relative frequencies change. Now

h_3 and h_2 are more frequent than h_1. So if one observes in the current haplotype population a lot of h_3 and h_2 but only a few h_1, one might think that h_2 and h_3 are the parental haplotypes and h_1 is the recombined version of them, which would be wrong. This is a simple case and one might find solutions such as using an out-group as reference to determine which of h_1, h_2 and h_3 are more ancient. But if the mutation rate is high, the haplotypes in the current haplotype population are very diverse though they are all mutated version of the ancestral haplotypes, h_1, h_2 and h_3. Here it becomes much harder to tell which ones are the parental haplotypes and which are the recombined ones.

The identification of recombination events also depends on where the events occurred in terms of the observed sequence. Recombinations close to either ends of the observed sequence are generally harder to detect than those in the middle. Intuitively, one needs highly dependent flanking sequences of a recombination to detect it. The recombinant, which is a hybrid version of two parental haplotypes, should be identical to one of the parental haplotypes on one side of the recombination event and identical to the other parental haplotypes on the other side. Here we assume the recombinant was observed right after the recombination. So it can be consider as one of the ancestral haplotypes and it later could have its own mutated versions in the current haplotype population.

Now let us recall that the calculations related to π_1 in (6.12) are far more expensive than the genotype-haplotype density in previous chapters. To partially mitigate the immense challenge of computation, we adopted an sliding window technique. The sliding window technique is to partition a string of data points into overlapping windows (bins). The partition step simplifies computation because smaller windows are easy to compute than the whole string. Recall also that we conditioned on $J = 1$ or 2, a feature most likely to be true in shorter sequences. The overlapping property of the windows ensure that every data point except those at both ends of the string is analyzed at least twice and the results are not greatly affected by how the start points and the end points of the windows are selected.

For analyzing simulated data 2 and the real data of HIV sequences, we chose the window

size to be 15 and the increment to be 7. We use the same notation for subsequence as in Section 6.2 and we define a subsequence from site j and $k-1$ as $x_{j;k}$. Then the whole sequence can be partitioned from the left end into the following subsequences with the window size of 15 and the increment of 7.

$$x_{1;16},x_{8;23},x_{15;30,....}$$

We then calculate the Bayes factor, $k_{t,s}$ of recombination at site s for subsequence t. One might transform $k_{t,s}$ values back to the Bayes factors for the original sequence, k_j by matching the subscripts of t and s to j. Then one should have the Bayes factors for all $j = 2, ..., L$ in the original sequence as

$$\begin{array}{llll} k_2, k_3, ... & k_9, k_{10}, ... & .... \\ & k'_9, k'_{10}, ... & .... \end{array} \tag{6.27}$$

Notice that k_1 cannot be estimated because location indicator, H_1 is at the left end of the sequence. Also note that we denote the second row of Bayes factor labeled as k' because with the overlapping windows, we have estimated the Bayes factor of recombination twice for k'_9 and rightwards. Similarly, at the right end of the original sequence, some of the sites are only analyzed once.

We then merge the two rows of the Bayes factor values by choosing the larger one wherever a site is analyzed twice. The merged Bayes factor values are denoted as K and the list of the merged Bayes factors is

$$\begin{array}{c} K_{12}, K_{13}, ...K_{1L}. \\ .... \\ K_{n2}, K_{n3}, ...K_{nL} \end{array} \tag{6.28}$$

where the first subscript is the individual index and the second one is the location index

6.6.1 Simulated data 1: Effects of tuning parameters

We used an artificial data set to illustrate the effects of tuning parameters, recombination rate, q and mutation rate p. The set consists of 96 copies of 1111100000, 96 copies of 0000011111 and one copy of 0000000000. We think of 0000000000 as the recombined version of the first two types of haplotypes and the most probable recombination event is between site 5 and site 6 ($H_5 = 1$). We use the recombination rates, $q = .01 - .49$ with .03 increment and the mutation rates $p = .01 - .49$ with .03 increment to compute the Bayes factor, $K(H_5 = 1)$ by equation (6.25). Contour maps of the calculated Bayes factor for different tuning parameter values is generated in Figure 6.1.

Figure 6.1 shows the Bayes factor for $H_5 = 1$ is a decreasing function of both the tuning parameters. Intuitively speaking, if both the recombination rate and the mutation rate in the model are considered to be high, the 0000000000 haplotype could be thought of as the result of multiple mutations and recombinations. Hence the chance of recombining in the middle is not much higher than recombining at other locations with additional mutations.

Maybe the Bayes factor value at the true recombination location alone was not informative. So we calculated the Bayes factor for $H_4 = 1$ under different tuning parameters as the background. We hoped to see how the $K(H_5 = 1)$ changed relatively to $K(H_4 = 1)$. So we took the ratio of the Bayes factors, $\frac{K(H_5=1)}{K(H_4=1)}$, which could be view as a measurement of relative strength of the recombination signal and generated the contour maps for the relative strength in Figure 6.2. The contour maps showed the relative strength measurement was also a decreasing function of the tuning parameter. However, we could theorize what happened to the Bayes factor, $K(H_5 = 1)$ and the ratio, $\frac{K(H_5=1)}{K(H_4=1)}$ for $p = .01$ and $q = .22 - .49$ (at the bottom right corner of the first graph in Figure 6.1 and the first graph in Figure 6.6). We interpret that to be when recombination rate is assumed to be high, Bayes factor values of recombination at other locations were elevated.

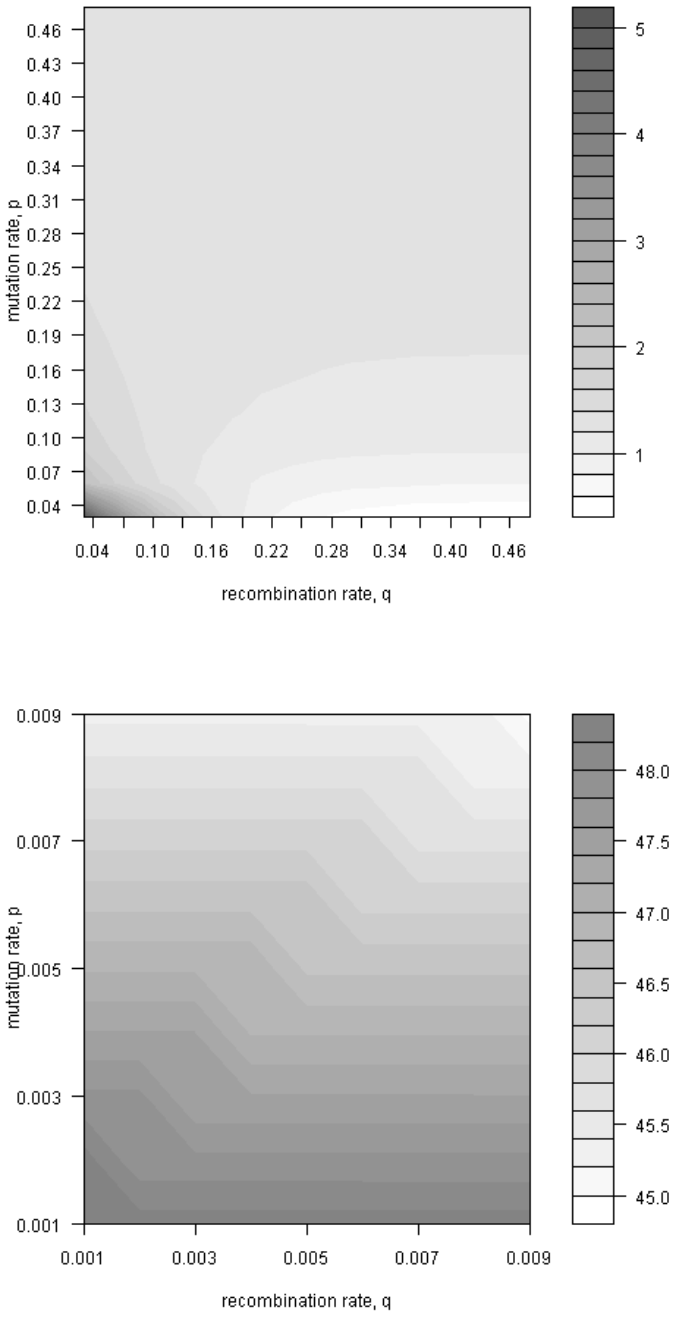
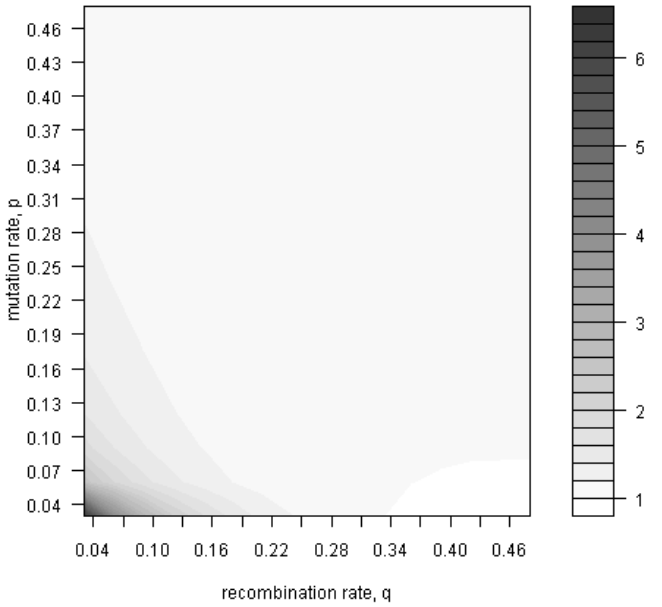


Figure 6.1: On the top is the contour map for $K(H_5 = 1)$ with respect to tuning parameters, recombination rate q and mutation rate, p (q and p are in the range of .01-.49 with .03 increment). At the bottom is the contour map for $K(H_5 = 1)$ over a different range of tuning parameters (q and p are in the range of .001-.01 with .001 increment).



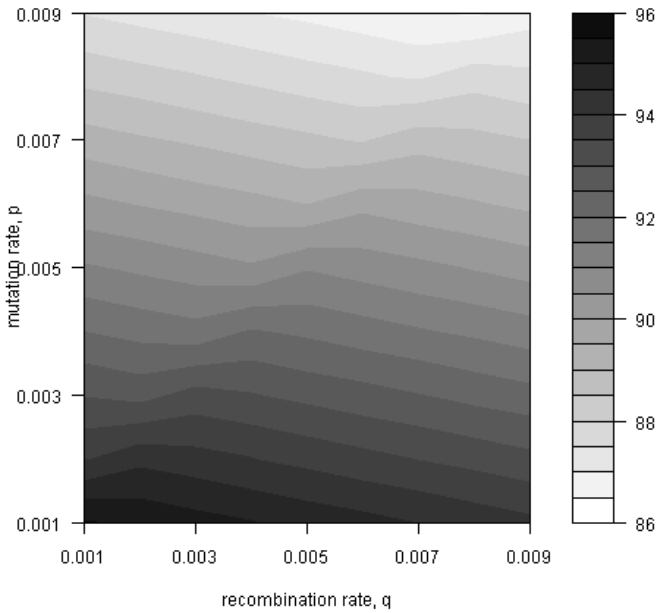


Figure 6.2: On the left is the contour map for $\frac{K(H_5=1)}{K(H_4=1)}$ with respect to tuning parameters, recombination rate q and mutation rate, p (q and p are in the range of .01-.49 with .03 increment). On the right is the contour map for $\frac{K(H_5=1)}{K(H_4=1)}$ with respect to tuning parameters, recombination rate q and mutation rate, p (q and p are in the range of .001-.01 with .001 increment)

If the mutation rate is considered to be higher, one might again think of the recombinant as a result of multiple mutations. If the recombination rate is higher, the recombinant might be considered as the result of multiple recombination. Both situations diminish the signal of the true recombination. The true signal becomes strong only when both rates are set to be low. We learned from this example that we should choose a small recombination rate and a small mutation rate in order to strengthen the recombination signal. So we use simulation 2 to investigate the performance of the Bayes factor (6.25) with different underlying (simulated) recombination rates and mutation rates.

6.6.2 Simulation 2: Infer recombination events

We follow the same procedures of generating haplotypes as described in section 6.1. We start by simulating a distribution Q that describes the frequencies of two ancestral sequences, with two ancestral sequences ϕ_1 and ϕ_2, and associated probabilities π_1 and $\pi_2 = 1 - \pi_1$. The length L is 64. We generate a sequence X by subsequences of ϕ_1 and ϕ_2, where the choices of subsequences results from recombinations.

1. First, we randomly choose between ϕ_1 and ϕ_2 with regards to π_1 and π_2 to start x. We denote the ancestor starting at site 1 as ϕ^1. The first site of x, $x(1)$ is now $\phi^1(1)$.

2. Before we move to site 2, we need to decide whether there is recombination between site 1 and site 2, which is equivalent to generating H_1. Here H_1 is generated by a Bernoulli draw with the probability of 1 being the simulated recombination rate, q_s. If $H_1 = 1$, we need to repeat step 1 for site 2 to find ϕ^2 (which may or may not be the same as ϕ^1) and then $x(2) = \phi^2(2)$. If $H_1 = 0$, we have $x(2) = \phi^2(2)$ and $\phi^2 = \phi^1$.

3. Repeat step 1 and step 2 to generate the complete sequence of x

4. Each site of the generated x is mutated with probability of the simulated mutation rate, p_s

each with 10 replications	p_s =.001	.005	.01	.05
q_s = .001	3(3)	3(3)	3(3)	3(3)
.005	46(40)	46(40)	46(40)	46(40)
.01	97(69)	97(69)	97(69)	97(69)
.05	430(134)	430(134)	430(134)	430(134)

Table 6.2: Numbers of true recombination events with respect to p_s and q_s. The numbers in () are the numbers of individuals that have at least one true recombination events.

We created 50 sequences with length, L=64 with different mutation rate p_s and recombination rate q_s. We chose p_s and q_s = .001, .005, .01 and .05.

We recorded the locations of the recombination events for $\{H_i = 1\}$, distinguish the recombinations between the same ancestor from the recombinations between different ancestors. We labeled the first kind 0 and the second kind 1. The recombination between different ancestors will be called true recombinations. This labeling actually tells us whether they can be identified or not. Table 6.2 summarizes the numbers of true recombination events with respect to p_s and q_s. The numbers in the parenthesis are the numbers of individuals having at least one true recombination.

In this simulation scheme, we encountered a problem of defining the recombination signal so as to match the recombination events on record. The calculated Bayes factor values for recombination for sites near the true recombination events are sometimes elevated all together. Thus instead of generating one large Bayes factor value for each recombination, the model may generate a few such signals that are close to each other. This often happens when the parental sequences are quite similar in the neighboring sites of the recombination. It becomes worse if the peak (the highest Bayes factor in the neighborhood) is not at the location of the recombination. We decided to call a run of consecutive high Bayes factors a "signal".

Here we propose a two-stage method for detecting recombinations using the Bayes factor. The first stage is to identify a list of candidate signals from the calculated Bayes factors by using a cutoff value, called the pilot cutoff, to identify the signals. The second stage is to

pick final signals from the list of candidate signals based on another cutoff value, called the final cutoff. The final cutoff might be determined by some prior information such as the empirical recombination rates or might be based on experience, such as inferring on average one recombination for each string of 50 sites.

Before we describe the two-stage method, let us discuss the choice of the cutoff values for the Bayes factor. In simulation 2, we have different numbers of true recombinations in different scenarios for the underlying recombination and mutation (Table 6.2). It would be wise to set the cutoff value according to the number of true recombinations. For instance, when there were three recombinations for $q_s = .005$ and $p_s = .005$, one might just choose the largest two signals.

Now let us introduce the concept of the candidate signal. We start by choosing the pilot cutoff, denoted as C_p, for which we used 1.2, to recode the Bayes factor values in (6.28) to a binary sequence of S_{ij} (S is short for signal),

$$S_{12}, S_{13}, ..., S_{1L}$$

$$....$$

$$S_{n2}, S_{n3}, ..., S_{nL}$$

We use the following rule to code the (S_{ij}) sequence.

$$S_{ij} \begin{cases} 1 & K_{ij} \geq C_p \\ 0 & K_{ij} < C_p \end{cases}.$$

We define candidate signals, denoted as SC_a each with a start point s_a and a end point e_a. We start with individual 1 and scan through the S_{1j} list.

- When we find the first $S_{1j} = 1$, we set s_1=j. If $S_{1j+1} = 0$, we set e_1 =j and set $SC_1 = K_{ij}$.

- If in the above case, $S_{1j+1} = 1$, we will find the next $S_{1k} = 0$ and set e_1 =k-1. We then let SC_1 be the highest K among (K_{1j}, ...,$K_{1(k-1)}$).

In similar fashion, we move along the S_{ij} list to find SC_2, its end point and its start point and so on.

For the second stage, we then choose the final cutoff value based on the SC_a list as well as based on the number of true recombinations in each scenario. For example, if we have two true recombinations, we just choose the top two off the SC_a list as the positive signals and match the corresponding s_a and e_a to the recorded locations of the true recombinations. The number of times that a true recombination is between the s_a and e_a of a positive signal is the number of true positives, denoted as n_t. Here the number of false positives, n_f is just $2 - n_t$. If the number of SC_a is smaller than the number of true recombinations, we just use all the SC_a as the positive signals. Here we do not want to lower the pilot cutoff (default at 1.2) because a Bayes factor close to 1 can be interpreted as $'$the chance of a recombination is no greater than the chance of no recombination$'$.

The recombination rate and the mutation rate of the tuning parameter were both set to .001 and the sliding window technique was adopted to estimate the Bayes factor in (6.25) of recombinations. We first generated the SC_a list based on the Bayes factor calculation with the pilot cutoff =1.2. The histograms of SC_a for different scenarios of p_s and q_s are in Figure 6.3. For illustrative purpose, we grouped the rates of .001 and .005 together and rates of .01 and .05 together in Figure 6.3. The SC_a lists clearly indicated that the signals of recombination were weakened in sets with large p_s and q_s.

Then we chose the final cutoff value for each scenario based on the number of true recombinations (Table 6.2). The final cutoff are values are summarized in Table 6.3. The final cutoff values for $q_s = .05$ were very close to the pilot cutoff (=1.2), which meant some of the candidate signals were very weak. We expected to capture a lot of false positives in these cases. The number of positive signals were summarized in Table 6.5

cutoff values	p_s =.001	.005	.01	.05
q_s = .001	8.99	9.02	8.53	7.67
.005	1.55	5.25	4.79	4.59
.01	1.24	2.86	3.86	3.09
.05	1.2	1.2	1.2	1.34

Table 6.3: The final cutoff values of each scenario

cutoff values	p_s =.001	.005	.01	.05
q_s = .001	3/3	3/3	3/3	3/3
.005	41/46	46/46	46/46	46/46
.01	98/97	97/97	98/97	97/97
.05	219/430	251/430	246/430	442/430

Table 6.4: Number of positive signals in each scenario based on the final cutoff values in Table 6.3. The first number is the number of positive signals based on the final cutoff values and the second number is the number of true recombinations in each scenario (from Table 6.2).

We used the recorded locations of the true recombinations to count the number of true positives (Table 6.5). The proposed strategy of the sliding window technique combined with constructing the list of composite signals worked fairly well when simulated mutation rates were small. It still identified some signals of recombination with large p_s but its performance deteriorated rapidly for large simulated mutation rates.

True positives	p_s =.001	.005	.01	.05
q_s = .001	1/3	0/3	0/3	0/3
.005	38/46	34/46	32/46	12/46
.01	85/97	80/97	65/97	27/97
.05	219/430	251/430	246/430	185/430

Table 6.5: Number of true positives in each scenario. The first number is the number of true positives and the second number is the number of the true recombinations. The pilot cutoff is 1.2 and the final cutoff values are in Table 6.3.

We also summarized the false positive rate (the ratio of the number of false positives and the number of positive signals) in Table 6.6. Combining the findings in Table 6.5 and Table 6.6, we were convinced that the signal of recombinations was very sensitive to the underlying mutation and recombination sites.

False positive rate	p_s =.001	.005	.01	.05
q_s = .001	.67	1	1	1
.005	.07	.26	.3	.74
.01	.13	.18	.34	.72
.05	.1	.17	.33	.58

Table 6.6: False positive rate for each scenario, the ratio of the number of false positives and the total number of positive signals with the pilot cutoff =1.2 and the final cutoff values (in Table 6.3).

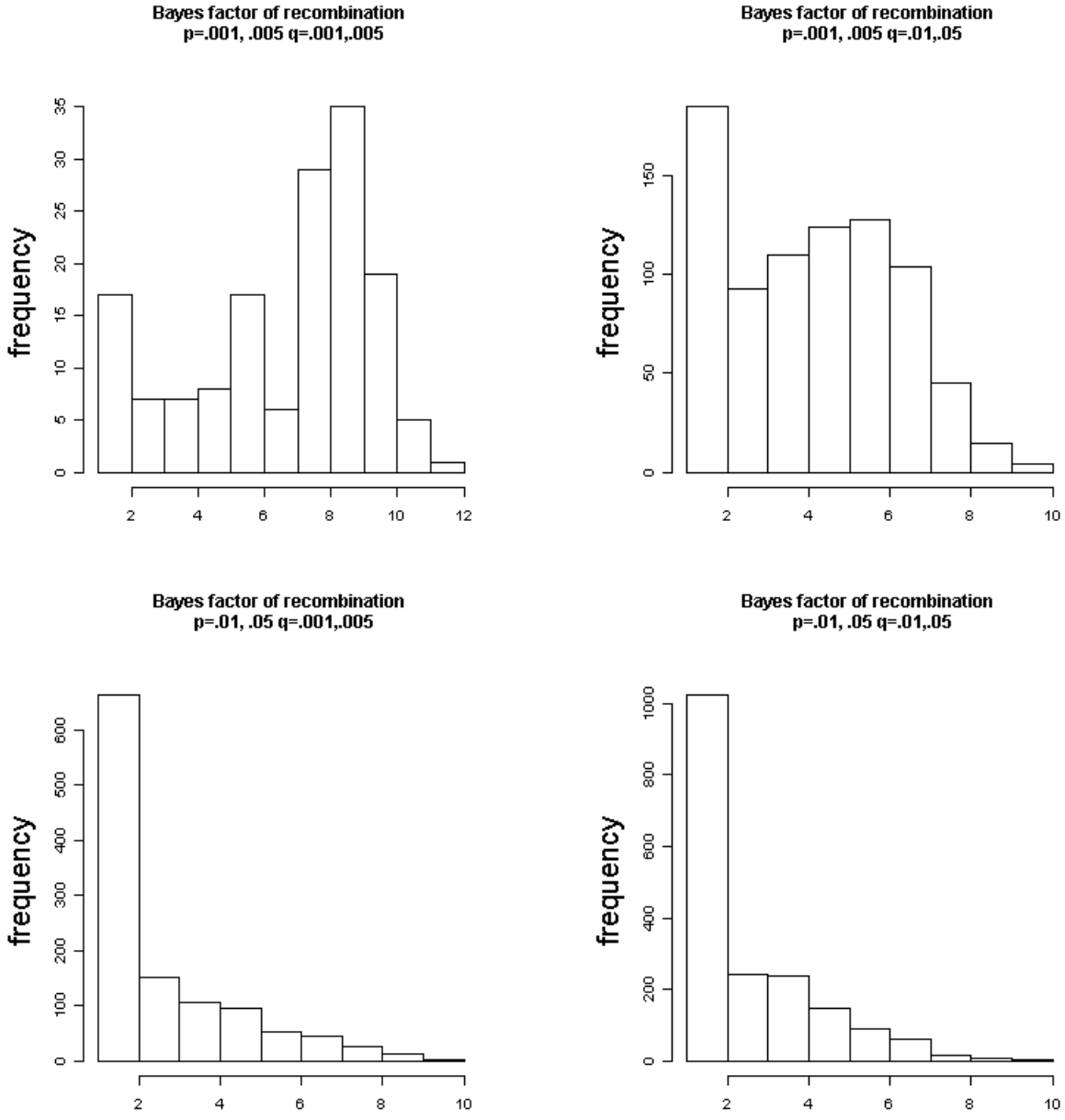


Figure 6.3: Histograms of the candidate SC'_as for the simulated sets. The lists of $SC'_a s$ are grouped together for $p_s = .0002, .002$ and $q_s = .0002, .002$; for $p_s = .0002, .002$ and $q_s = .01, .05$; for $p_s = .01, .05$ and $q_s = .0002, .002$; and for $p_s = .01, .05$ and $q_s = .01, .005$.

Chapter 7

Summary and Future work

We will summarize the research contributions of this thesis and then propose some future work. The summary of contributions is in Section 7.1. The proposal of the future work is in Section 7.2.

7.1 Summary of research contribution.

In this thesis, we used kernel density estimation and the modal expectation-maximization (MEM) method to tackle the haplotype-genotype problem and recombination estimation. We presented the haplotype model, the haplotype-genotype model and the recombination model based on haplotype data in this thesis. They were all based on continuous time Markov Chain kernels with a mixture model structure. The kernels were designed to mimic important biological processes.

In Chapter 2, we derived a density estimator for the haplotype mutation model and used the MEM method to cluster the individuals via modes. The topic of clustering binary sequences using this kernel had been studied extensively by Chen and Lindsay (Chen and Lindsay, 2006). Their clustering via mode method was based on full maximum likelihood and so is considerably slower than the modal density methodology we developed here. We did not focus on this methodology here, but instead used the haplotype model as a precursor

to more complex models so that we could introduce the concepts of Likelihood expectation-maximization (LEM), density estimation, the degrees of freedom (DOF) calculation and the MEM method.

We assumed there was a haplotype ancestral distribution, $\pi(\phi)$, in each of our three models. We updated the haplotype density from an initial density $\pi_0(\phi)$, to a new density with higher likelihood by LEM. For a starting density we used an independent-sites MLE in all three models, as it makes for the easiest calculations. For individual tasks of interest, such as clustering, haplotyping, and inferring recombinations, we then constructed appropriate inference functions based on the new density.

We can summarize the research contributions of this dissertation as follows:

- We constructed a density estimator $\pi_1(\phi)$ for ancestral haplotypes, ϕ, for the genotype-haplotype model. We then constructed an objective function for haplotype pairs, h and $\bar{h}$ based on the ancestral distribution $\pi_1(\phi)$. We proposed two MEM algorithms, the whole length MEM (wMEM) and the reweighted MEM (rMEM) to (partially) maximize the objective function with specified starting value.

- We proposed several strategies of choosing starting values for the haplotyping problem. The first one was the neighbor starting value method. We constructed the so-called partial haplotypes and selected the neighbor starting value for a particular individual based on the unambiguous sites on the other individuals. The second strategy of choosing starting values was based on the ancestral haplotype density and was called the missing data based strategy. We constructed the ancestral distribution, $\pi_1^*(\phi)$ based on the partial haplotypes, which was indeed a density estimator with missing data. We then started with each partial haplotype to find the complete haplotype that was fixed at the unambiguous sites and maximized $\pi_1^*(\phi)$. The third strategy was to use a list of solved haplotypes (from individuals with less than two ambiguous sites) to assign starting values for the rest of the samples. The last one was to set the starting values to 1 for genotypes = 1 or 2 and to 0 for genotypes = 0.

- We used a simulation study to show that the wMEM algorithm with multiple starting values had slightly better performance than PHASE (considered among the most accurate methods of haplotype inference). Although the rMEM algorithm worked well even with naive starting value, we deemed it was too slow.

- We chose a 1,600-site genotype data set of Yoruba population from HapMap Project as an example. We partitioned the 1,600 sites into subsequences of different length. The partitions were 100 subsequences with 16 sites each, 133 with 12 sites each and 266 with 6 sites each. The performance of wMEM with multiple starting values became closer to PHASE's when the length of subsequences was smaller.

- We chose a 4,000-site genotype data set of Yoruba population from HapMap Project. The 4,000 sites were partitioned into 80 subsequences of 50 sites. The haplotyping error on the subsequences was correlated with the degrees of freedom (DOF) values of the ancestral density of the haplotype model, DOF_h, as well as with the DOF values of several genotype models. We then showed D$'$, a measurement of linkage disequilibrium, was a indicator of haplotyping error. We proposed that long subsequences could be partitioned into smaller subsequences to achieve stronger dependence structure within the subsequences. We also illustrate a way of detecting possible breakpoints in long sequences based on DOF values.

- We then developed a strategy for partitioning the long sequences into smaller subsequences in a way that the sites from the same subsequence were more dependent than the sites from different subsequences. We partitioned the 4,000-site data set into 325 subsequences using the partition stratgey and we called this partition the informative partition. We created a partition containing 333 subsequences each with 12 sites and called this partition the arbitrary partition.

- We applied different MEM methods with multiple starting values to solve the haplotype configuration of the subsequences from the informative partition as well as from the

arbitrary partition. We found that compared to the arbitrary partition, the informative partition improved the accuracy of the MEM methods in terms of reducing the total number of haplotyping errors.

- We developed a ligation method for sewing the partitioned subsequences back to the full length sequence. The ligation was done on an individual basis and we ligated two neighboring sequences at a time. To solve the labeling problem, we ligated the subsequences sequentially; after each ligation step, we match the label (order) of haplotype pairs to the previous ligation step.

- We applied the ligation method to the solved subsequences in the arbitrary partition and in the informative partition to recover the full length sequences. The assessment of the errors on the full length sequences showed the partition-ligation strategy yielded fewer switch errors in the informative partition than in the arbitrary partition.

- Our overall strategy for large scale haplotyping was therefore to first create a partition using DOF, then apply the wMEM method with multiple starting values, and finally ligate using the subsequence density function.

- We proposed a new recombination model which allowed both mutation and recombination. In the recombination model, we assumed sites were independent conditional on the recombination sites and the ancestors of the runs. The calculation included a summation over all the $u; v$ runs for $u < v$, which essentially meant all the possible runs. Recall that in the haplotype model and the genotype-haplotype model, we only assumed that the sites are independent conditional on the ancestor of the full sequence. In the recombination model, we assumed recombinations might have happened between any two adjacent sites and only the sites in the subsequences from the same ancestor then were conditionally independent given the ancestor. In other words, one should think of each of the subsequences as a vessel carrying some (potentially different) information of the ancestral distribution.

- We used the MEM method to find the modes on the ancestral distribution for the recombination model. With the same tuning parameter of mutation, individual binary sequences formed fewer clusters (fewer modes in the density) for larger tuning parameter of recombination.

- We derived a strategy from the recombination model for inferring recombination between sites based on the Bayes factor. It was a two-stage strategy. First we simplified the problem by assuming that the segments of interest only had a few recombinations. We then computed a Bayes factor for inferring recombinations between every two adjacent sites. We then identified a list of candidate signals from the Bayes factor values based on a pilot cutoff. At the second stage, we chose a cutoff based on the number of true recombinations and identified the positive signals from the candidate signals. A simulation study showed the strategy worked well in the scenarios when both the underlying mutation rate and the underlying recombination rate were low. As expected, its performance deteriorated rapidly when the mutation rate was increased.

7.2 Future work

Our overall strategy for haplotyping underperformed fastPHASE, one of the most popular haplotyping softwares on long sequences. On the other hand, our performance on short sequences was quite competitive. We can think of several ways to improve our proposed strategy. First of all, the partition method we developed was meant to break a long sequence into subsequences in a way that the dependence structure was strong within each subsequence and was weak across different subsequences. An alternative would be to let dependence structure between the neighboring sites of the adjacent subsequences be stronger so that the ligation of adjacent sequences would be easier. A second way to improve our strategy would be to use all the modes inferred by the wMEM method in the subsequences to do the ligation other than to use only the highest mode. To merge sequences based multiple modes, one of

course needs to reconsider the ligation method. In addition to improving the haplotyping method, we should implement the methods for missing data and for tuning the predictive power of the MEM method via cross-validation.

We also plan to extend our haplotype-genotype model to a Markov chain model for the ancestral haplotypes. Our original setting was to define the ancestral distribution $\pi(\phi)$ on the entire sequence ϕ. We can assume $\pi(\phi)$ is a Markov Chain, which can be expressed as

$$\pi(\Phi = \phi) = \pi(\Phi(1) = \phi(1)) \times \pi(\Phi(2) = \phi(2) | \Phi(1) = \phi(1)) \times ... \times \pi(\Phi(L) = \phi(L) | \Phi(L-1) = \phi(L-1))$$

where $\phi(s)$ is the ancestral haplotype on the sth site, $\pi(\centerdot)$ is the marginal density and $\pi(\centerdot|\centerdot)$ is the conditional density. With the Markovian $\pi(\phi)$, one might infer the haplotypes of each site directly based on the Markovian structure rather than partition the long sequence in order to solve for the smaller subsequences. FastPHASE uses such a hidden Markovian model.

For the recombination problem, it would be useful to construct further statistics that provide hypothesis testing and/or statistical inference. One could also think of the recombinations in one sequence as a mixture model. For a mixture model, we assume there are two mixture components, θ_1 and θ_2 . The mixture density is

$$f(x; \theta) = \pi f(x; \theta_1) + (1 - \pi) f(x; \theta_2)$$

where π is the mixture proportion for θ_1. The hypothesis testing for the number of components is

$$H_0 : \quad \text{one component}$$
$$H_1 : \quad \text{2 components} \quad .$$

In a realization of the above in the recombination model, one can think of two ancestral

haplotypes as the two components. For a given observed haplotype, we denote N to be the number of sites from ancestor 1 (component 1) and L-N to be the number of sites from ancestor 2 (component 2) where N is a random variable. The hypothesis testing for the number of components becomes the hypothesis testing for recombination, which could be expressed as follows,

H_0 : N=0 or L (Sample has one component; sample is recombination free)

H_1 : N $\neq$ 0 or L .

One could also incorporate the concepts of sensitivity and specificity into the two-stage strategy of detecting recombinations.

We plan to apply the two-stage strategy of detecting recombination to some real data. As an example, we have begun investigating the HIV virus. HIV is a type of retrovirus (viruses made of RNA) which undergoes reverse transcription in its life cycle. HIV has a very high mutation rate of approximately 3 x 10^{-5} mutations per nucleotide base per cycle of replication (Mansky and Temin, 1995). There are three groups of HIV-1 viruses, called M, N, and 0, as classified primarily by the *env* gene (the gene of the protein envelope of the virus) (Thomas, et al. 2002). Group M is the most prevalent group; it has subtypes such as A, B, C, D and so forth. What is less well known is that HIV can recombine during reverse transcription (Robertson, et al. 1995). During the life cycle of a retrovirus, it is usually reverse transcribed to a DNA sequence, the retroviral DNA, and the retroviral DNA is then incorporated into host's genome. Therefore, if a patient is infected with more than one (sub)types of HIV, the reverse transcription process may cause recombinations between the retroviral DNA sequences of different types of HIV. Recombinations between different parental subtypes have been classified and the resulting recombinant strains are documented as the so-called circulating recombinant forms (CRF) (Carr et al. 1998).

At least 48 CRFs have been identified (Los Alamos National Laboratory HIV database,

http://www.hiv.lanl.gov/.). Some of the CRFs were formerly classified as subtypes. To infer recombinations on HIV sequences is a good way to test our model and compare it against other methods of detecting recombination (Robertson, et al. 1995; Martin and Rybicki, 1999; Zhuang, et. al. 2002; Levy, et al. 2005).

Moving beyond the scope of the thesis, we can consider extending the recipe of density estimation and MEM to solve many other problems. Some that come to mind are solving haplotypes with family data, the gene conversion problem, the admixture problem and the clustering of individuals by genotype problem.

Appendix A

Construction of a haplotype density in the genotype-haplotype model

We can construct a product density for the genotype X using the genotype mutation kernel conditional on the two ancestral haplotypes μ_1 and μ_2 as

$$G_p(X = x; \mu_1, \mu_2) = \prod_{s=1}^{L} g_p(X(s) = x(s); \mu_1(s), \mu_2(s)).$$

Assuming random mating from a common population of haplotypes $\pi(\mu)$, the density for x, unconditionally, is then

$$f(X = x, \pi) = \sum_{\mu_1} \sum_{\mu_2} G_p(X = x; \mu_1, \mu_2)\pi(\mu_1)\pi(\mu_2). \tag{A.1}$$

A.1 Likelihood EM steps: LEM1

In the **continuous case** that the likelihood EM (LEM) works as follows. Given a density for latent component ϕ, $\pi(\phi)$, and the component density $k(x; \phi)$, the mixture model density

is

$$f(x;\pi) = \sum_{\phi} k(x;\phi)\pi(\phi). \tag{A.2}$$

Given a random sample x_1,...,x_n from (A.2), with unknown model parameter π, the likelihood function becomes

$$L(\pi; X = x_1, x_2, ..., x_n) = \prod \sum_{\phi} k(x;\phi)\pi(\phi). \tag{A.3}$$

The idea of LEM is that one can take a current estimator, $\pi_{old}(\phi)$ and update it to a new estimator, $\pi_{new}(\phi)$ that has higher likelihood in (A.3) using the EM algorithm. One LEM step from a current estimator π_{old} is obtained from the update step as:

$$\pi_{new}(\phi) = \pi_{old}(\phi)\Delta(\phi),$$

where

$$\Delta(\phi) = n^{-1} \sum \frac{k(x_i;\phi)}{f(x_i;\pi_{old})}.$$

A.2 LEM step 1 towards the haplotype density π_1 of the genotype-haplotype model

We use an initial value for the density

$$\pi_0(\mu) = \prod_s \gamma_s^{\mu(s)}(1-\gamma_s)^{1-\mu(s)}$$

where γ_s is the relative frequency of 1 at site s. We use the update function for $\pi(\mu)$ given in Appendix A.1. We get

$$\Delta(\mu) = n^{-1} \sum_i \frac{\sum_\phi G_p(x_i;\mu,\phi)\pi_0(\phi)}{\sum_{\mu_1}\sum_{\mu_2} G_p(x_i;\mu_1,\mu_2)\pi_0(\mu_1)\pi_0(\mu_2)}.$$

Now the independence structure of both G_p and π_0 implies we can break the sum in the numerator up by site, and get

$$\sum_{\phi} G_p(x_i;\mu,\phi)\pi_0(\phi) = \prod_s \{\gamma_s g_p(x_i(s),\mu(s),1) + (1-\gamma_s) g_p(x_i(s),\mu(s),0)\}.$$

If we set $r_1(x_i(s),\mu) = [\gamma_s g_p(x_i(s),\mu(s),1) + (1-\gamma_s) g_p(x_i(s),\mu(s),0)]$, then from the above kernel description, we get

$$\begin{aligned}
r_1(0,\mu) &= \gamma_s g_p(0,\mu,1) + (1-\gamma_s) g_p(0,\mu,0) \\
&= \gamma_s m_p(0,\mu) m_p(0,1) + (1-\gamma_s) m_p(0,\mu) m_p(0,0) \\
&= m_p(0,\mu)\{\gamma_s m_p(0,1) + (1-\gamma_s) m_p(0,0)\} \\
&= m_p(0,\mu) A_s \\
&\text{where} \quad A_s = \gamma_s m_p(0,1) + (1-\gamma_s) m_p(0,0) \\
r_1(2,\mu) &= \gamma_s g_p(2,\mu,1) + (1-\gamma_s) g_p(2,\mu,0) \\
&= \gamma_s m_p(1,\mu) m_p(1,1) + (1-\gamma_s) m_p(1,\mu) m_p(1,0) \\
&= m(1,\mu)\{\gamma_s m_p(1,1) + (1-\gamma_s) m_p(1,0)\} \\
&= m(1,\mu) B_s \\
&\text{where} \quad B_s = \gamma_s m_p(1,1) + (1-\gamma_s) m_p(1,0) \\
r_1(1,\mu) &= 1 - m_p(0,\mu) A_s - m_p(1,\mu) B_s
\end{aligned}$$

and

$$\sum_{\phi} G_p(x_i;\mu,\phi)\gamma(\phi) = \prod_s r_1(x(s),\mu(s))$$

The same device works in the denominator to give

$$\begin{aligned}\sum_{\mu_1}\sum_{\mu_2} G(x_i;\mu_1,\mu_2)\gamma(\mu_1)\gamma(\mu_2) &= \sum_{\mu_2}\prod_s r_1(x(s),\mu_2(s))\gamma(\mu_2(s)) \\ &= \prod_s[\gamma_s r_1(x(s),1)+(1-\gamma_s)r_1(x(s),0)] \\ &= \prod_s r_2(x(s))\end{aligned}$$

where

$$r_2(x(s)) = \begin{cases} A_s^2 & x(s)=0 \\ B_s^2 & x(s)=2 \\ 2A_sB_s & x(s)=1 \end{cases}$$

Both the numerator and the denominator have product structure. Therefore we can modify the kernel $r_1(x,\mu)$ to

$$\begin{aligned} r_{3s}(0,\mu) &= r_1(0,\mu)/r_2(0,\mu) \\ &= m_p(0,\mu)A_s/A_s^2 \\ &= m_p(0,\mu)/A_s \\ r_{3s}(2,\mu) &= m_p(1,\mu)/B_s \\ r_{3s}(1,\mu) &= \frac{1-A_s m_p(0,\mu)-B_s m_p(0,\mu)}{2A_sB_s}\end{aligned}$$

This gives the LEM1 estimator of the density of the haplotype density as

$$\begin{aligned}\pi_1(\mu) &= \gamma(\mu)\Delta(\mu) \\ &= n^{-1}\sum_i\prod_s r_{3s}(x_i(s);\mu(s))\gamma_s^{\mu(s)}(1-\gamma_s)^{1-\mu(s)}\end{aligned}$$

Now it clearly makes sense to redefine r_s again to equal be $r_{3s}(x_i(s);\mu(s))\gamma_s^{\mu(s)}(1-\gamma_s)^{1-\mu(s)}$,

so that we have the simpler expression

$$\pi_1(\mu) = n^{-1} \sum_i \prod_s r_s(x_i(s), \mu(s)). \tag{A.4}$$

where

$$\begin{aligned} r_s(0,\mu) &= \frac{m_p(0,\mu)}{A_s}\gamma_s^{\mu}(1-\gamma_s)^{1-\mu} \\ r_s(2,\mu) &= \frac{m_p(1,\mu)}{B_s}\gamma_s^{\mu}(1-\gamma_s)^{1-\mu} \\ r_s(1,\mu) &= \frac{1 - A_s m_p(0,\mu) - B_s m_p(1,\mu)}{2A_s B_s}\gamma_s^{\mu}(1-\gamma_s)^{1-\mu} \end{aligned}$$

This $\pi_1(\phi)$ is just the updated density estimator of the ancestral haplotypes.

Appendix B

The conditional probability of the haplotype pair that generate the given genotype sequence

The objective function of inferring haplotypes,

$$h_{i1}^{*} = \arg\max_{h_{i1}} \sum_{\mu_1} \sum_{\mu_2} M_p(h_{i1}, \mu_1) M_p(\overline{h}_{i1}, \mu_2) \pi_1(\mu_1) \pi_1(\mu_2) \tag{B.1}$$

is the posterior probability of observing h_{i1} (and $\overline{h}_{i1} = x - h_{i1}$) given x_i. We focus on the right hand side of (B.1),

$$f^{*}(h_{i1}) = \sum_{\mu_1} \sum_{\mu_2} M_p(h_{i1}, \mu_1) M_p(x_i - h_{i1}, \mu_2) \pi_1(\mu_1) \pi_1(\mu_2)$$

where M_τ is the mutational kernel and π_1 is defined by (A.4). The π_1 density has the product structure, so we can interchange the product terms in the π_1's and the summations over μ_1 and μ_2,

$$
\begin{aligned}
& f^*(h_{i1}) \\
= \; & n^{-2} \sum_{\mu_1,\mu_2,j,k} (\prod_s m_p(h_{i1}(s),\mu_1(s)) m_p(x_i(s)-h_{i1}(s),\mu_2(s)) r_s(x_j(s),\mu_1(s)) r_s(x_k(s),\mu_2(s))) \\
= \; & n^{-2} \sum_{j,k} (\prod_s \sum_{\mu_1,\mu_2} m_p(h_{i1}(s),\mu_1(s)) m_p(x_i(s)-h_{i1}(s),\mu_2(s)) r_s(x_j(s),\mu_1(s)) r_s(x_k(s),\mu_2(s)))
\end{aligned}
$$

The inner sum over μ_1, μ_2 equals

$$
\begin{aligned}
& m_p(h_{i1}(s),0) m_p(x_i(s)-h_{i1}(s),0) r_s(x_j(s),0) r_s(x_k(s),0) \\
& + m_p(h_{i1}(s),1) m_p(x_i(s)-h_{i1}(s),0) r_s(x_j(s),1) r_s(x_k(s),0) \\
& + m_p(h_{i1}(s),0) m_p(x_i(s)-h_{i1}(s),1) r_s(x_j(s),0) r_s(x_k(s),1) \\
& + m_p(h_{i1}(s),1) m_p(x_i(s)-h_{i1}(s),1) r_s(x_j(s),1) r_s(x_k(s),1)
\end{aligned}
$$

for each site s.

The last displayed formula equals the $k_4(h_{i1}(s), x_j(s)) k_4(x_i(s) - h_{i1}(s), x_k(s))$, where k_4 can be expressed as follows,

$$
k_4(h,x) = m(h,0) r_s(x,0) + m(h,1) r_s(x,1)
$$

Appendix C

Haplotype model

Given haplotypes, h_1, ..., h_n, the kernel density estimator for the ancestral haplotype, ϕ, using the mutation kernel is

$$\hat{f}_\tau(\phi) = n^{-1} \sum_i \prod_s h_p(h_i(s), \phi(s))$$

where

$$h_p(h_i(s), \phi(s)) = \frac{m_p(h_i(s), \phi(s))}{D_s(h_i(s))} \gamma_s^{\phi(s)} (1 - \gamma_s)^{1-\phi(s)};$$

i and s are the sequence and site indices, respectively; $D_s(h_i(s)) = m(x_i(s), 0)(1 - \gamma_s) + m(x_i(s), 1)\gamma_s$; γ_s is the relative frequency of 1 at site s.

Appendix D

Simulation results

The following tables summarize the haplotyping errors, err_{site} of the MEM algorithms with tuning parameter, p =.05, .2, .3 and .4 in the simulation study of Chapter 3.

	Simulated mutation rate	1	2	3	4	5	6	7
L=8	.005	0.145 (0.089)	0 (0)	0.002 (0.002)	0 (0)	0 (0)	0 (0)	0.002 (0.002)
	.01	0.075 (0.075)	0.001 (0.001)	0.001 (0.001)	0.001 (0.001)	0.001 (0.001)	0.001 (0.001)	0.001 (0.001)
	.05	0.140 (0.046)	0.073 (0.018)	0.119 (0.016)	0.070 (0.017)	0.073 (0.018)	0.073 (0.018)	0.088 (0.013)
	.1	0.279 (0.029)	0.100 (0.013)	0.227 (0.019)	0.100 (0.017)	0.097 (0.013)	0.099 (0.012)	0.123 (0.009)
	.15	0.295 (0.014)	0.218 (0.013)	0.271 (0.006)	0.198 (0.013)	0.210 (0.012)	0.209 (0.010)	0.204 (0.021)
	.2	0.296 (0.019)	0.233 (0.007)	0.283 (0.016)	0.231 (0.008)	0.229 (0.009)	0.233 (0.008)	0.246 (0.015)
L=16	.005	0.278 (0.069)	0.004 (0.003)	0.007 (0.003)	0.004 (0.003)	0.004 (0.003)	0.004 (0.003)	0.003 (0.002)
	.01	0.137 (0.082)	0.015 (0.004)	0.013 (0.007)	0.015 (0.004)	0.015 (0.004)	0.015 (0.004)	0.021 (0.004)
	.05	0.221 (0.056)	0.110 (0.021)	0.124 (0.027)	0.095 (0.019)	0.095 (0.019)	0.096 (0.019)	0.107 (0.022)
	.1	0.306 (0.024)	0.185 (0.013)	0.221 (0.026)	0.133 (0.017)	0.135 (0.017)	0.139 (0.018)	0.168 (0.016)
	.15	0.320 (0.008)	0.284 (0.007)	0.306 (0.013)	0.231 (0.014)	0.231 (0.014)	0.246 (0.012)	0.246 (0.007)
	.2	0.344 (0.011)	0.328 (0.014)	0.334 (0.010)	0.274 (0.013)	0.278 (0.014)	0.291 (0.016)	0.276 (0.010)
L=32	.005	0.407 (0.024)	0.017 (0.008)	0.017 (0.007)	0.017 (0.008)	0.017 (0.008)	0.017 (0.008)	0.016 (0.006)
	.01	0.387 (0.035)	0.020 (0.003)	0.017 (0.004)	0.019 (0.003)	0.020 (0.003)	0.020 (0.003)	0.015 (0.002)
	.05	0.345 (0.016)	0.131 (0.015)	0.206 (0.034)	0.096 (0.020)	0.095 (0.020)	0.114 (0.021)	0.094 (0.018)
	.1	0.349 (0.006)	0.258 (0.023)	0.257 (0.015)	0.193 (0.010)	0.192 (0.010)	0.194 (0.010)	0.220 (0.013)
	.15	0.339 (0.015)	0.315 (0.007)	0.316 (0.010)	0.284 (0.019)	0.284 (0.019)	0.288 (0.019)	0.288 (0.012)
	.2	0.371 (0.010)	0.346 (0.006)	0.342 (0.009)	0.332 (0.010)	0.332 (0.010)	0.337 (0.007)	0.324 (0.012)

Table D.1: Comparison of wMEM with different starting values, rMEM and PHASE under the simulation scheme with the tuning parameter, $p = .05$. For each cell, the first number is the average err_{site} rate and the second number is its standard error. 1 means wMEM with the naive starting value; 2 means wMEM with the naive starting value and the trivial individual starting value; 3 means wMEM with the neighbor starting value; 4 means wMEM with the missing data based starting value; 5 means the highest modes among 1, 2, 3 and 4; and 6 means the rMEM algorithm with the naive starting value; 7 means the result from PHASE.

	Simulated mutation rate	1	2	3	4	5	6	7
L=8	.005	0.003 (0.003)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0.002 (0.002)
	.01	0.006 (0.004)	0.003 (0.002)	0.003 (0.002)	0.003 (0.002)	0.003 (0.002)	0.003 (0.002)	0.001 (0.001)
	.05	0.105 (0.018)	0.076 (0.020)	0.119 (0.014)	0.070 (0.018)	0.076 (0.020)	0.076 (0.020)	0.088 (0.013)
	.1	0.225 (0.039)	0.100 (0.011)	0.183 (0.023)	0.100 (0.013)	0.100 (0.012)	0.099 (0.009)	0.123 (0.009)
	.15	0.285 (0.021)	0.219 (0.013)	0.255 (0.011)	0.203 (0.013)	0.212 (0.012)	0.211 (0.011)	0.204 (0.021)
	.2	0.296 (0.019)	0.224 (0.010)	0.282 (0.009)	0.228 (0.009)	0.222 (0.012)	0.231 (0.010)	0.246 (0.015)
L=16	.005	0.004 (0.003)	0.004 (0.003)	0.004 (0.003)	0.004 (0.003)	0.004 (0.003)	0.004 (0.003)	0.003 (0.002)
	.01	0.019 (0.007)	0.013 (0.004)	0.012 (0.004)	0.013 (0.004)	0.013 (0.004)	0.013 (0.004)	0.021 (0.004)
	.05	0.155 (0.051)	0.089 (0.015)	0.097 (0.017)	0.088 (0.017)	0.089 (0.016)	0.089 (0.016)	0.107 (0.022)
	.1	0.250 (0.021)	0.151 (0.019)	0.152 (0.024)	0.142 (0.017)	0.143 (0.017)	0.141 (0.016)	0.168 (0.016)
	.15	0.293 (0.013)	0.261 (0.014)	0.275 (0.012)	0.234 (0.016)	0.230 (0.011)	0.233 (0.011)	0.246 (0.007)
	.2	0.340 (0.011)	0.321 (0.014)	0.321 (0.011)	0.276 (0.012)	0.273 (0.015)	0.283 (0.013)	0.276 (0.010)
L=32	.005	0.259 (0.099)	0.017 (0.008)	0.016 (0.008)	0.017 (0.008)	0.017 (0.008)	0.017 (0.008)	0.016 (0.006)
	.01	0.327 (0.081)	0.021 (0.003)	0.021 (0.003)	0.020 (0.003)	0.020 (0.003)	0.020 (0.003)	0.015 (0.002)
	.05	0.291 (0.044)	0.093 (0.019)	0.099 (0.023)	0.093 (0.021)	0.092 (0.020)	0.092 (0.020)	0.094 (0.018)
	.1	0.334 (0.020)	0.206 (0.011)	0.202 (0.010)	0.189 (0.010)	0.188 (0.010)	0.189 (0.011)	0.220 (0.013)
	.15	0.340 (0.009)	0.294 (0.018)	0.290 (0.014)	0.274 (0.021)	0.275 (0.020)	0.275 (0.022)	0.288 (0.012)
	.2	0.369 (0.010)	0.328 (0.009)	0.330 (0.009)	0.320 (0.014)	0.320 (0.014)	0.319 (0.013)	0.324 (0.012)

Table D.2: Comparison of wMEM with different starting values, rMEM and PHASE under the simulation scheme with the tuning parameter, $p = .2$. For each cell, the first number is the average err_{site} rate and the second number is its standard error. 1 means wMEM with the naive starting value; 2 means wMEM with the naive starting value and the trivial individual starting value; 3 means wMEM with the neighbor starting value; 4 means wMEM with the missing data based starting value; 5 means the highest modes among 1, 2, 3 and 4; and 6 means the rMEM algorithm with the naive starting value; 7 means the result from PHASE.

	Simulated mutation rate	1	2	3	4	5	6	7
L=8	.005	0.003 (0.003)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0.002 (0.002)
	.01	0.042 (0.033)	0.003 (0.002)	0.003 (0.002)	0.003 (0.002)	0.003 (0.002)	0.003 (0.002)	0.001 (0.001)
	.05	0.116 (0.016)	0.081 (0.023)	0.101 (0.019)	0.072 (0.019)	0.081 (0.023)	0.081 (0.023)	0.088 (0.013)
	.1	0.193 (0.028)	0.092 (0.008)	0.171 (0.022)	0.093 (0.011)	0.095 (0.009)	0.095 (0.010)	0.123 (0.009)
	.15	0.292 (0.016)	0.223 (0.014)	0.277 (0.016)	0.204 (0.015)	0.219 (0.013)	0.218 (0.011)	0.204 (0.021)
	.2	0.295 (0.016)	0.231 (0.011)	0.287 (0.013)	0.231 (0.010)	0.230 (0.010)	0.240 (0.011)	0.246 (0.015)
L=16	.005	0.004 (0.003)	0.004 (0.003)	0.004 (0.003)	0.004 (0.003)	0.004 (0.003)	0.004 (0.003)	0.003 (0.002)
	.01	0.023 (0.008)	0.013 (0.004)	0.016 (0.003)	0.015 (0.003)	0.013 (0.004)	0.013 (0.004)	0.021 (0.004)
	.05	0.126 (0.025)	0.087 (0.016)	0.099 (0.020)	0.092 (0.019)	0.090 (0.018)	0.088 (0.016)	0.107 (0.022)
	.1	0.210 (0.019)	0.146 (0.020)	0.153 (0.027)	0.148 (0.020)	0.151 (0.021)	0.148 (0.018)	0.168 (0.016)
	.15	0.289 (0.019)	0.255 (0.016)	0.273 (0.014)	0.237 (0.014)	0.231 (0.012)	0.227 (0.009)	0.246 (0.007)
	.2	0.334 (0.010)	0.321 (0.015)	0.319 (0.011)	0.274 (0.011)	0.278 (0.013)	0.280 (0.010)	0.276 (0.010)
L=32	.005	0.104 (0.084)	0.017 (0.008)	0.017 (0.008)	0.017 (0.008)	0.017 (0.008)	0.017 (0.008)	0.016 (0.006)
	.01	0.186 (0.084)	0.020 (0.003)	0.021 (0.003)	0.020 (0.003)	0.020 (0.003)	0.020 (0.003)	0.015 (0.002)
	.05	0.133 (0.047)	0.092 (0.019)	0.094 (0.023)	0.094 (0.022)	0.093 (0.020)	0.092 (0.019)	0.094 (0.018)
	.1	0.282 (0.032)	0.206 (0.011)	0.208 (0.012)	0.191 (0.012)	0.191 (0.013)	0.190 (0.015)	0.220 (0.013)
	.15	0.325 (0.013)	0.304 (0.021)	0.299 (0.019)	0.279 (0.020)	0.280 (0.019)	0.283 (0.020)	0.288 (0.012)
	.2	0.373 (0.006)	0.345 (0.011)	0.343 (0.011)	0.320 (0.013)	0.318 (0.014)	0.316 (0.013)	0.324 (0.012)

Table D.3: Comparison of wMEM with different starting values, rMEM and PHASE under the simulation scheme with the tuning parameter, $p = .3$. For each cell, the first number is the average err_{site} rate and the second number is its standard error. 1 means wMEM with the naive starting value; 2 means wMEM with the naive starting value and the trivial individual starting value; 3 means wMEM with the neighbor starting value; 4 means wMEM with the missing data based starting value; 5 means the highest modes among 1, 2, 3 and 4; and 6 means the rMEM algorithm with the naive starting value; 7 means the result from PHASE.

	Simulated mutation rate	1	2	3	4	5	6	7
L=8	.005	0.003 (0.003)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0.002 (0.002)
	.01	0.042 (0.033)	0.003 (0.002)	0.003 (0.002)	0.003 (0.002)	0.003 (0.002)	0.003 (0.002)	0.001 (0.001)
	.05	0.120 (0.017)	0.081 (0.023)	0.099 (0.022)	0.075 (0.018)	0.081 (0.023)	0.081 (0.023)	0.088 (0.013)
	.1	0.190 (0.028)	0.090 (0.009)	0.171 (0.023)	0.093 (0.011)	0.093 (0.010)	0.093 (0.010)	0.123 (0.009)
	.15	0.295 (0.014)	0.224 (0.016)	0.274 (0.012)	0.207 (0.019)	0.222 (0.017)	0.217 (0.014)	0.204 (0.021)
	.2	0.298 (0.016)	0.229 (0.010)	0.282 (0.013)	0.231 (0.011)	0.228 (0.010)	0.238 (0.017)	0.246 (0.015)
L=16	.005	0.004 (0.003)	0.004 (0.003)	0.004 (0.003)	0.004 (0.003)	0.004 (0.003)	0.004 (0.003)	0.003 (0.002)
	.01	0.073 (0.058)	0.015 (0.003)	0.016 (0.003)	0.015 (0.003)	0.015 (0.003)	0.015 (0.003)	0.021 (0.004)
	.05	0.118 (0.020)	0.093 (0.015)	0.110 (0.025)	0.095 (0.018)	0.094 (0.017)	0.093 (0.017)	0.107 (0.022)
	.1	0.208 (0.021)	0.148 (0.021)	0.156 (0.030)	0.147 (0.022)	0.147 (0.020)	0.149 (0.018)	0.168 (0.016)
	.15	0.283 (0.020)	0.261 (0.015)	0.267 (0.013)	0.230 (0.017)	0.225 (0.014)	0.222 (0.011)	0.246 (0.007)
	.2	0.338 (0.009)	0.319 (0.016)	0.324 (0.010)	0.277 (0.013)	0.282 (0.015)	0.277 (0.010)	0.276 (0.010)
L=32	.005	0.108 (0.088)	0.017 (0.008)	0.017 (0.008)	0.017 (0.008)	0.017 (0.008)	0.017 (0.008)	0.016 (0.006)
	.01	0.111 (0.073)	0.019 (0.003)	0.020 (0.003)	0.019 (0.003)	0.019 (0.003)	0.019 (0.003)	0.015 (0.002)
	.05	0.124 (0.052)	0.094 (0.021)	0.094 (0.021)	0.095 (0.021)	0.092 (0.020)	0.094 (0.021)	0.094 (0.018)
	.1	0.274 (0.034)	0.205 (0.010)	0.199 (0.013)	0.187 (0.014)	0.188 (0.015)	0.190 (0.014)	0.220 (0.013)
	.15	0.324 (0.019)	0.309 (0.021)	0.306 (0.019)	0.280 (0.019)	0.283 (0.018)	0.288 (0.019)	0.288 (0.012)
	.2	0.373 (0.006)	0.342 (0.012)	0.343 (0.011)	0.323 (0.015)	0.321 (0.016)	0.317 (0.011)	0.324 (0.012)

Table D.4: Comparison of wMEM with different starting values, rMEM and PHASE under the simulation scheme with the tuning parameter, $p = .4$. For each cell, the first number is the average err_{site} rate and the second number is its standard error. 1 means wMEM with the naive starting value; 2 means wMEM with the naive starting value and the trivial individual starting value; 3 means wMEM with the neighbor starting value; 4 means wMEM with the missing data based starting value; 5 means the highest modes among 1, 2, 3 and 4; and 6 means the rMEM algorithm with the naive starting value; 7 means the result from PHASE.

Bibliography

[1] Andersen, E. W.(2004). Composite likelihood and two-stage estimation in family studies *Biostatistics*, 5: 15-30

[2] Archer, J. (2008) Identifying the important HIV-1 recombination breakpoints. *PLoS Comput Biol.* 2008 Sep 12;4(9):e1000178.

[3] Bailey, T. and Elkan, C. (1994) Fitting a mixture model by expectation maximization to discover motifs in biopolymers. *Proceedings of International Conference on Intelligent Systems for Molecular Biology.* 2:28-36

[4] Beaumont, M., Zhang, W. and Balding, D. (2002) Approximate Bayesian Computation in Population Genetics. *Genetics.* 162: 2025–2035

[5] Besag, J. (1975). Statistical analysis of non-lattice data. *The Statistician*, 24(3): 179-195

[6] Bellman, R. (1957). *Dynamic Programming*, Princeton University Press

[7] Browning, S.R. (2006). Multilocus association mapping using variable-length markov chains. *Am. J. Hum. Genet*, 78: 903-913

[8] Carr, J. K., et al. (1998). Reference sequences representing the principle genetic diversity of HIV-1 in the Pandemic. Human retroviruses and AIDS: a compilation and analysis of nucleic acid and amino acid sequences, Los Alamos National Laboratory, Los Alamos, New Mexico.

[9] Chen, S. C. (2003). Clustering binary sequences using mixture trees.(PhD dissertation, Pennsylvannia State University)

[10] Chen, S. C. and Lindsay, B. G. (2006). Building mixture trees from binary sequence data *Biometrika*, 93, 4:843-860

[11] Chung, Y. and Lindsay, B. G. (2009). A Likelihood-tuned Density Estimator via a Nonparametric Mixture Model *IMS Lecture Notes Monograph Series* accepted

[12] Churchill, G. and Doerge, R. (1994) Empirical Threshold Values for Quantitative Trait Mapping. *Genetics*, Vol 138, 963-971.

[13] Clack, A. (1990). Inference of haplotypes from PCR-amplified samples of diploid populations. *Molecular Biology and Evolution* , 7: 111-122

[14] Cleveland, W. and Devlin, S. (1988) Locally weighted regression: an approach to regression analysis by local fitting. *Journal of the American Statistical Association,* Vol. 83, No. 403: 596-610

[15] Dempster, A. P., Laird, N. M. and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society Series B*, 39: 1-38

[16] Devlin, B. et. al. (1996). Disequilibrium Mapping: Composite Likelihood for Pairwise Disequilibrium *Genomics*, 36, 1(15): 1-16

[17] Diaconis, P. and Holmes, S. P. (1998). Matchings and phylogenetic trees *Proc. Natl. Acad. Sci.*, 95(25): 14600-14602

[18] Dib, C. et al. (1996). A comprehensive genetic map of the human genome based on 5264 microsatellties. *Nature*, 380: 152-154

[19] Efron, B., Hastie, T., Johnstone, I. and Tibshirani, R. (2004). Least angle regression (with discussion). *Ann. Statist.*, 32, 407–499.

[20] Everitt, B. and Hand D. (1981). *Finite mixture distributions*, Chapman & Hall

[21] Excoffier, L. and Slatkin, M. (1995). Maximum-likelihood estimation of molecular haplotype frequencies in a diploid population *Molecular Biology and Evolution*, 12: 921-927

[22] Fan, J., Negroni, M.and Robertson, D. (2007) The distribution of HIV-1 recombination breakpoints. *Infection, Genetics and Evolution* 7:717-723

[23] Felsenstein, J. (1985) Confidence limits on phylogenies: an approach using the bootstrap. *Evolution*, 39(4), 783-791

[24] Fredslund, J. (2006) PHY·FI: fast and easy online creation and manipulation of phylogeny color figures. *BMC Bioinformatics* 2006, 7:315

[25] Gelfand, A.E. and Smith, A.F.M. (1990) Sampling based approaches to calculating marginal densities *Journal of the American Statistical Association*, 85: 398-409

[26] Geman, S. and Geman, D. (1984). Stochastic relaxation, Gibbs distribution and Bayesian restoration of images. *IEE Transactions on Pattern Analysis and Machine Intelligence*, 6: 721-741

[27] Geyer, C. J. and Thompson, E. A. (1992). Constrained Monte Carlo maximum likelihood for dependent data, (with discussion). *J. Roy. Statist. Soc. Ser. B*, 54: 657-699

[28] Gower, J.C. (1966). Some distance properties of latent root and vector methods used in multivariate analysis. *Biometrika* 53:325-328

[29] Gusfield, D and Orzack, S. (2005). Haplotype Inference *Handbook of Computational Molecular Biology* Pages 18-1 to 18-28 Chapman & Hall/CRC Computer and Information Science Series

[30] Gunter, L. and Zhu, J. (2007). Efficient computation and model selection for the support vector regression. *Neural Computation*, 19, 1633–1655.

[31] Hastie, T. and Tibshirani, R. (1990). *Generalized Additive Models* Chapman & Hall/CRC Monographs on Statistics & Applied Probability

[32] Hellenthal, G., Stephens M. (2006). msHOT: modifying Hudson's ms simulator to incorporate crossover and gene conversion hotspots *Bioinformatics*,23(4): 520-521

[33] Hey, J. and Kliman, R. (2002). Interactions between natural selection, recombination and gene density in the genes of Drosophila. *Genetics*, 160: 595-608.

[34] Holmes, E., et. al. (2001) Metabonomic characterization of genetic variations in toxicological and metabolic responses using probabilistic neural networks. *Chem. Res. Toxicol.* 14 (2): 182–191

[35] Husmeier D. (2005) Discriminating between rate heterogeneity and interspecific recombination in DNA sequence alignments with phylogenetic hidden Markov models. *Bioinformatics*, 21 Suppl.2: ii166-ii172

[36] Kaplan, N.L., Darden, T., Hudson, R.R. (1988). The coalescent process in models with selection. *Genetics*, 120: 819-829

[37] Kimmel, G and Shamir, R (2005). Genotype resolution and block identification using likelihood. *PNAS* 2005 102 (1) 158-162

[38] Kindahl EC (1994). *Recombination and DNA polymorphism on the third chromosome of Drosophila melanogaster*. Cornell University, Ithaca, NY.

[39] Kingman, J.F.C. (1982). On the Genealogy of Large Populations. *Journal of Applied Probability* 19A: 27-43

[40] Kirsch et al. (2000). A systematic, high-resolution linkage of the cytogenetic and physical maps of the human genome. *Nat Genet.*, 24(4): 339-40

[41] Kuhner, M. K. and Felsenstein, J. (1994). A simulation comparison of phylogeny algorithms under equal and unequal evolutionary rates. *Mol. Biol. Evol.*, 11: 459-468

[42] Levy, D. , Aldrovandi G., Kutsch O. and Shaw G. (2004) Dynamics of HIV-1 recombination in its natural target cells. *PNAS* 2004 vol. 101 no. 12 4204-4209

[43] Li, J., Ray S. and Lindsay, B. G. (2007). A nonparametric statistical approach to clustering via mode identification, *Journal of Machine Learning Research*, 8(8): 1687-1723

[44] Liang, G. and Yu,B. (2003). Maximum pseudo likelihood estimation in network tomography. *IEEE T Signal Proces*, 51(8): 2043-2053.

[45] Lindsay, B. G. (1988). "Composite Likelihood Methods". *Contemporary Mathematics* , 80: 221-239

[46] Lindsay B. G. (1995). Mixture models Theory, Geometry and Applications. *NSF-CBMS Regional Conference Series in Probability and Statistics, Institute of Statistical mathematics*, vol.5, Hayward.

[47] Lindsay, B. G. et al. (2008). Quadratic distances on probabilities: A unified foundation *Ann. Stat.*, 36(2): 983-1006

[48] Los Alamos National Laboratory HIV database List of CRF `http://www.hiv.lanl.gov/content/sequence/HIV/CRFs/CRFs.html`

[49] Marchini, J. and et. al. (2006) A Comparison of Phasing Algorithms for Trios and Unrelated Individuals. *Amercan Journal of Human Genetics*, 78:437-450

[50] Mansky, L. and Temin, H.(1995) Lower In Vivo Mutation Rate of Human Immunodeficiency Virus Type 1 than That Predicted from the Fidelity of Purified Reverse Transcriptase. *Journal of Virology,* Aug. 1995, p. 5087–5094

[51] Martin, D. and Rybicki, E. (2000) RDP: detection of recombination amongst aligned sequences. *Bioinformatics* 16 (6): 562-563.

[52] McLachlan, G. and Basford, K. (1988) *Mixture Models: Inference and Applications to Clustering*. Marcel Dekker, New York.

[53] Minin VN, et al. (2005). Dual multiple change-point model leads to more accurate recombination detection. *Bioinformatics* 21: 3034-3042

[54] Mohle, M., Sagitov, S. (2001). A classification of coalescent processes for haploid exchangeable population models *The Annals of Probability*, 29: 1547-1562

[55] Moore, R. and Stevens, M. (2008). Local Patterns of Nucleotide Polymorphism Are Highly Variable in the Selfing Species Arabidopsis thaliana *Journal of Molecular Evolution*, 66(2): 116-129

[56] Morris, A.P., et al. (2002). Fine-scale mapping of disease loci via shattered coalescent modeling of genealogies. *Am. J. Hum. Genet*, 70: 686-707

[57] Nei, M. and Kumar, S. (2000). *Molecular Evolution and Phylogenetics*. Oxford University Press, New York.

[58] Neuhauser, C., Krone, S.M. (1997). The genealogy of samples in models with selection *Genetics* , 145: 519-534

[59] Niu, T. et al. (2002). Bayesian Haplotype Inference for Multiple Linked Single-Nucleotide Polymorphisms *Am. J. Hum. Genet*, 70(1): 157-169.

[60] Niu, T. et al. (2005). An expectation-maximization-likelihood-ratio test for handling missing data: application in experimental crosses. *Genetics*, 169(2): 1021-31.

[61] Nye, T. M. W. , Lio, P. and Gilks, W. R. (2006). A novel algorithm and web-based tool for comparing two alternative phylogenetic trees *Bioinformatics*, 22(1): 117-119

[62] Pearson, K. (1901). "On Lines and Planes of Closest Fit to Systems of Points in Space" . *Philosophical Magazine* 2 (6): 559–572.

[63] Pritchard, J. K. et al. (2000). Inference of population structure using multilocus genotype data *Genetics*, 155: 945-959

[64] Chowdhury1 R. et al. (2009) Genetic Analysis of Variation in Human Meiotic Recombination. *PLoS Genet* 5(9): e1000648.

[65] Rissanen, J. Modeling by shortest data description. *Automatica*, vol. 14 , pp. 465-471.

[66] Roach, J. C., et. al., (2010) Analysis of genetic inheritance in a family quartet by whole-genome sequencing. *Science*, Vol. 328. no. 5978, pp. 636 - 639

[67] Robertson D.L., Hahn B.H., Sharp P.M. (1995). "Recombination in AIDS viruses". J Mol Evol. 40 (3): 249–59.

[68] Robinson, D. R. and Foulds, L. R. (1981). Comparison of phylogenetic trees. *Mathematical Biosciences*, 53: 131-147.

[69] Rubin, D. (1976) Inference and missing data. *Biometrika* 63 (3): 581-592.

[70] Scheet P. and Stephens M. (2006). A fast and flexible statistical model for large-scale population genotype data: applications to inferring missing genotypes and haplotypic phase. *Am. J. Hum. Genet*, 78(4): 629-44.

[71] Smith, T. and Waterman, M. (1981). Identification of Common Molecular Subsequences. *Journal of Molecular Biology* 147: 195–197

[72] Stein, M., Chi, Z. and Welty, L. (2004). Approximating the Likelihood for Irregularly Observed Gaussian Random Fields. *JRSS B*, 66: 275-296.

[73] Stephens, M., N. J. Smith, and P. Donnelly (2001). A new statistical method for haplotype reconstruction from population data. *Am. J. Hum. Genet*, 68: 978-989.

[74] Stephens, M. and Donnelly, P. (2003)A comparison of Bayesian methods for haplotype reconstruction from population genotype data. *Am. J. Hum. Genet*, 73: 1162-1169

[75] Suchard MA, et al. (2003). Inferring spatial phylogenetic variation along nucleotide sequences: a multiple changepoint model. *J. Am. Stat. Assoc* 98: 427-437

[76] Tanner, M. A. and Wong, W. H. (1987). The calculation of posterior distributions by data augmentation (with discussion). *Journal of the American Statistical Association*, 82: 528-550

[77] The International HapMap Consortium. The International HapMap Project. Nature 426, 789-796 (2003).

[78] Thomson, M. M., Perez-Alvarez, L. and Najera, R. (2002). "Molecular epidemiology of HIV-1 genetic forms and its significance for vaccine development and therapy". *Lancet Infect. Dis.* 2 (8): 461–471

[79] Thompson, E. A. (1994). Monte Carlo Likelihood in Genetic Mapping *STATISTICAL SCIENCE* , 9(3): 355-366

[80] Torgerson. W. S. (1952). Multidimensional scaling: I. Theory and method. *Psychometrika.* 17. 401-419.

[81] Wright, S. I. et. al., (2008). Effective population size and tests of neutrality at cytoplasmic genes in Arabidopsis *Genetics Research*, 90: 119-128

[82] Xing, E. et al. (2004). Bayesian Haplotype Inference via the Dirichlet Process *Proceedings of the 21 st International Conference on Machine Learning*, Banff, Canada

[83] Ye, J. (1998) On measuring and correcting the effects of data mining and model selection. *J. Amer. Statist. Assoc.*, 93, 120–131.

[84] Zaykin, D., et al. (2002) Testing Association of Statistically Inferred Haplotypes with Discrete and Continuous Traits in Samples of Unrelated Individuals. *Human Heredity.* 53:79–91

[85] Zeng, Z. (1994) Precision mapping of quantitative trait loci. *Genetics.* Vol 136, 1457-1468,.

[86] Zhang, K. et al. (2002) A dynamic programming algorithm for haplotype partitioning. *Proc Natl Acad Sci* 99: 7335-7339.

[87] Zhao, P., Rocha, G. and Yu, B. (2006). Grouped and hierarchical model selection through composite absolute penalties. Technical report, Dept. Statistics, Univ. California, Berkeley

[88] Zhu, L. and Bustamante, C. D. (2005).A Composite-Likelihood Approach for Detecting Directional Selection From DNA Sequence Data *Genetics*, 170: 1411-1421

[89] Zhuang, J. et al. (2002) Human Immunodeficiency Virus Type 1 Recombination: Rate, Fidelity, and Putative Hot Spots. *Journal of Virology*, Vol. 76, No. 22, p. 11273-11282

[90] Zou, H., Hastie, T. and Tibshirani, R. (2007) On the "degrees of freedom" of the lasso. *Annuals of Statistics*, Volume 35, Number 5, 2173-2192.

Vita

Xianyun Mao

Education

Ph.D. , Statistics The Pennsylvania State University, Decemeber 2010

M.S.. , Genetics The Pennsylvania State University, Aug 2006

B.S., Biological Sciences Fudan University China, June 2003

Research Experience

2007–2010: Density Estimation and Expectation Maximization in biological problems under the direction of Dr. Bruce Lindsay

2003–2007: Admixture, population structure and related topics under the direction of Dr. Mark Shriver

Admixture and ancestral informative markers for US populations

Admixture mapping study for breast cancer in African American women by Dr. Ellsworth of Henry M. Jackson Foundation for the Advancement of Military Medicine

Statistical support for ancestry estimation and admixture for studies by Dr. Fernandez of University of Alabama at Birmingham

Statistical support for "African American Live" and "African American Live 2" (PBS series)

2002–2003: Population genetics and migration of Ethnic groups in China under the direction of Dr. Wen and Dr. Jin (See Publications 6-7)