Analysis and Visualization of Local Phylogenetic Structure within Species

Jeremy R. Wang

A dissertation submitted to the faculty of the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science.

Chapel Hill
2013

Approved by:

Leonard McMillan

Wei Wang

Fernando Pardo-Manuel de Villena

Gary Churchill

Vladimir Jojic

Patrick Sullivan

Abstract

JEREMY R. WANG. Analysis and Visualization of Local Phylogenetic Structure within Species.
(Under the direction of Leonard McMillan.)

While it is interesting to examine the evolutionary history and phylogenetic relationship between species, for example, in a sort of "tree of life", there is also a great deal to be learned from examining population structure and relationships within species. A careful description of phylogenetic relationships within species provides insights into causes of phenotypic variation, including disease susceptibility. The better we are able to understand the patterns of genotypic variation within species, the better these populations may be used as models to identify causative variants and possible therapies, for example through targeted genome-wide association studies (GWAS). My thesis describes a model of local phylogenetic structure, how it can be effectively derived under various circumstances, and useful applications and visualizations of this model to aid genetic studies.

I introduce a method for discovering phylogenetic structure among individuals of a population by partitioning the genome into a minimal set of intervals within which there is no evidence of recombination. I describe two extensions of this basic method. The first allows it to be applied to heterozygous, in addition to homozygous, genotypes and the second makes it more robust to errors in the source genotypes.

I demonstrate the predictive power of my local phylogeny model using a novel method for genome-wide genotype imputation. This imputation method achieves very high accuracy - on the order of the accuracy rate in the sequencing technology - by imputing genotypes in regions of shared inheritance based on my local phylogenies.

Comparative genomic analysis within species can be greatly aided by appropriate visualization and analysis tools. I developed a framework for web-based visualization and analysis of multiple individuals within a species, with my model of local phylogeny providing the underlying structure. I will describe the utility of these tools and the applications for which they have found widespread use.

Acknowledgments

I would like to thank my advisor, Leonard McMillan, for his constant support and advice, my parents for encouraging me in my love and pursuit of computers and education, my brilliant wife for her love and support, and my twin brother for talking about any and every geeky thing I came up with, and for reading and helping me revise this document. He hates that comma-full sentence.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Phylogenies describe the genetic relationships between species or individuals. Inferring these relationships helps us better understand genetic similarities and differences which may drive phenotypic variation and describe the organism's history. Inter-species phylogenies - evolutionary trees - describe the derivation of genomes between species. Intra-species phylogenetic structure describes the more complex multiple-inheritance relationships between individuals in an interbreeding population.

Genomic structure can be represented, at a high level, as a mosaic of ancestral genomes. This representation simplifies genome-wide analyses and is more robust than typical analyses based on point markers. Relationships between individuals can be used to assess genome-wide associations. I will first discuss phylogenetics and how my approach is related to higher-level genomic structure. I will then define the genomic features and events forming the basis for my model of local phylogenetic structure.

1.1 Phylogenetics

The study of phylogenetics aims to describe the relationships between phyla, species, or individuals by their shared genetic material. Phylogenetics has been a major area of study in biology for a very long time. At a high level, the phylogeny of all organisms depicts the

evolutionary tree of life, with branches implying division between entire groups of organisms. In a phylogeny among species, these branches indicate "speciation" events where a species diverged to evolve into two or more different species. Species do not interbreed, so there is no recombination between their genomes. Figure 1.1 shows an evolutionary tree describing the phylogenetic structure of a large portion of life on earth.

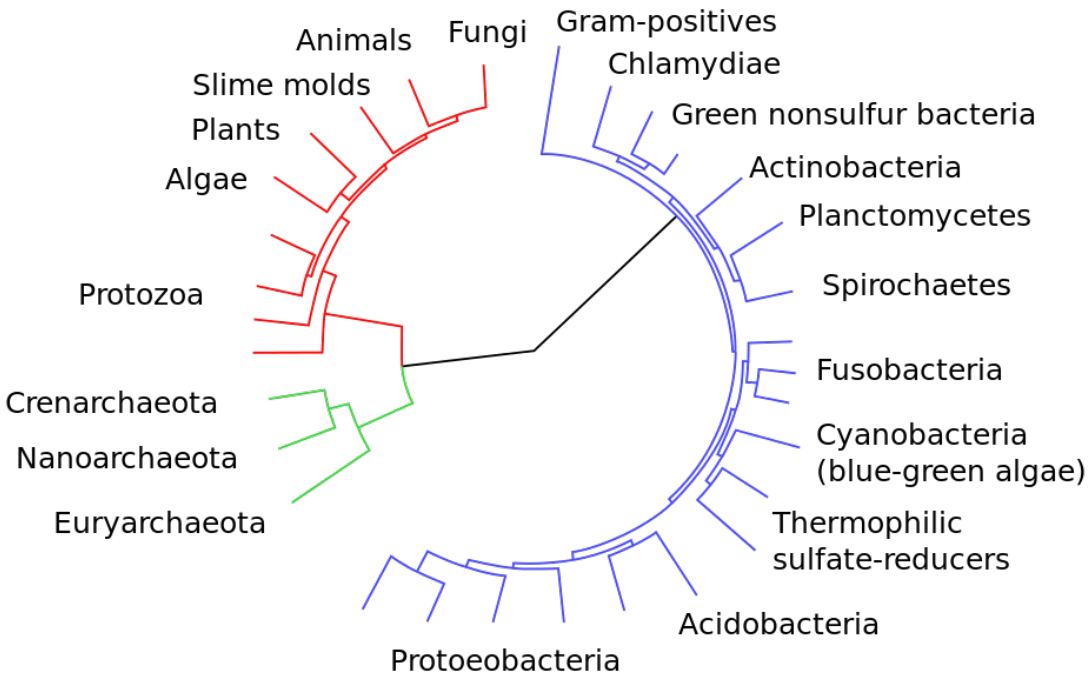


Figure 1.1: A phylogenetic tree indicating the evolutionary descent of organisms, including the top two taxonomic levels: Domain, including Eukaryotes (red), Bacteria (green), and Archaea (blue), and Kingdom, the most familiar of which will be Animals, Plants, and Fungi. This type of phylogenetic tree is rooted in the center with time progressing outward. Branches indicate the point at which groups diverged, those closer to the outside of the circle indicating more recent developments (on an evolutionary time scale). Viridae not shown. Credit Wikimedia Commons (`http://commons.wikimedia.org`).

Although there is considerable disagreement among biologists about the appropriate taxonomic levels and classifications, the schema for representing phylogenetic structure as an evolutionary tree down to the level of species or subspecies is widely agreed-upon [93]. The input data used initially were macroscopic or microscopic characteristics (e.g., abdominal bristles in *Drosophila*, coat color in mouse, and Gram stain and shape for bacteria).

DNA-based phylogenies became feasible beginning with one or a few genetic markers, and currently based on hundreds of thousands of genetic markers or full genome sequence. Although the methods I will describe are general, the applications and experimental results have focused on the laboratory mouse. Figure 1.2 shows a phylogenetic tree of a subset of the subfamily Murinae (family Muridae) including the species *Mus musculus* (the house mouse) and five known subspecies. It is at this point that we start to see a breakdown of the standard form of evolutionary trees. *Mus musculus molossinus* is generally considered a hybrid of *M. m. castaneus* and *M. m. musculus*, and thus not the result of a simple speciation event. As we begin to consider the phylogenetic structure within species, subspecies, and populations, this global depiction of phylogeny breaks down.

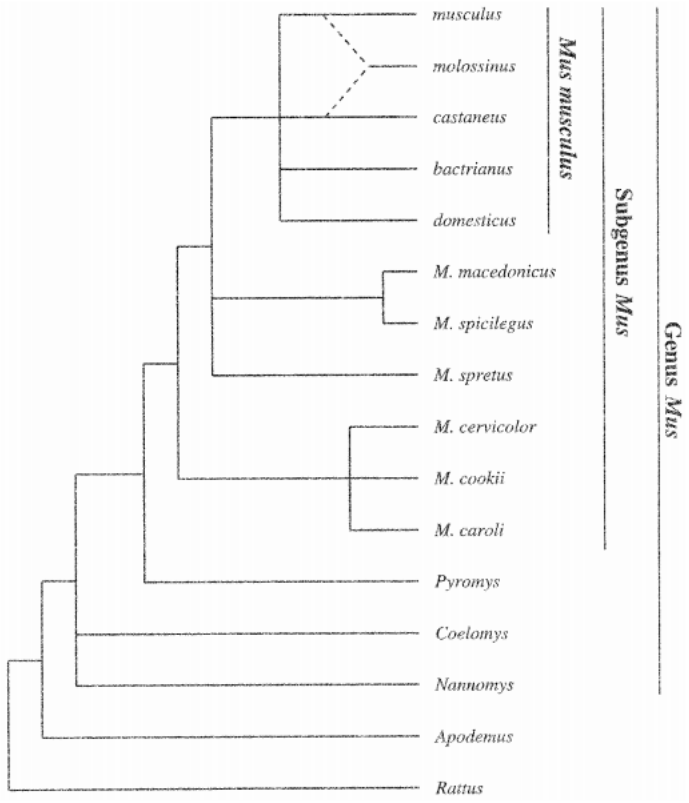


Figure 1.2: Phylogenetic tree of a subset of the subfamily Murinae (family Muridae) including the species *Mus musculus* (the house mouse), and five known subspecies, *Mus musculus domesticus*, *M. m. musculus*, *M. m. castaneus*, *M. m. molossinus*, and *M. m. bactrianus*. The format of this tree is similar to Figure 1.1 except it is rooted on the left and time progresses toward the right. (reproduced from [56])

Phylogenetic analyses among closely related species or individuals are often determined by some measure of global genetic similarity. For example, we can line up the genetic sequence of two individuals and compare them base-by-base, computing percent identity. Using a sample (individual) from each of several strains (for our purposes right now, consider these isolated subpopulations) of common laboratory mice, we can do this kind of comparison then construct a phylogenetic tree by recursively merging the "nearest neighbor" - the two strains with the highest percent identity (this is called neighbor-joining). Figure 1.3 depicts this type of tree. While this kind and construction of tree is not strictly an evolutionary tree, it somehow represents a phylogenetic relationship [66]. This is especially true of the placement of SPRET, CAST, and PWK. These strains are known to predominantly represent *Mus spretus* (a different but closely related species), *Mus musculus castaneus*, and *M. m. musculus*, respectively, while the remainder of the strains are predominantly *M. m. domesticus*. The tree in Figure 1.3 accurately reflects these species and subspecific divisions.

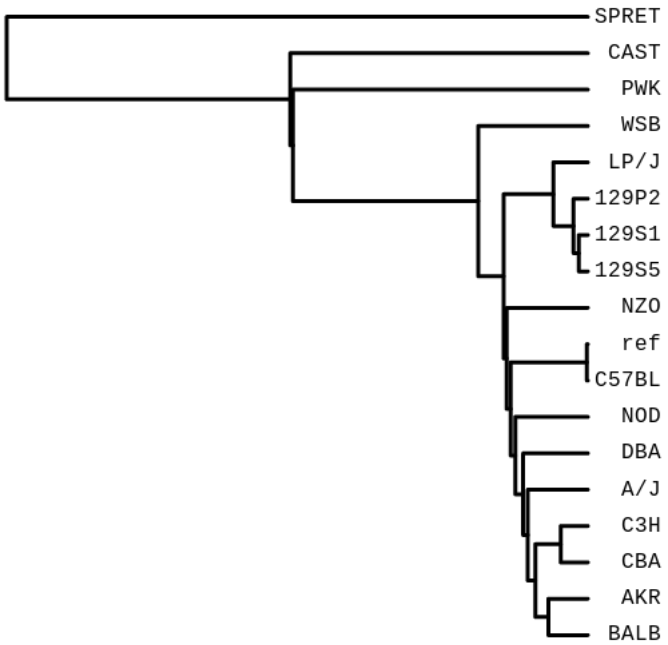


Figure 1.3: A phylogenetic tree constructed using neighbor-joining over a set of common strains of laboratory mice. This kind of tree, while not strictly an evolutionary tree, somehow represents the phylogenetic relationship among a set of samples.

In the case of species, this general approach to phylogenetics is appropriate and effective, mainly based on the constraint that individuals of different species cannot typically interbreed to produce fertile offspring. The first violation can be seen at the subspecies level. *Mus musculus molossinus* introduces an atypical feature into our tree because it is thought to be a hybrid of *M. m. musculus* and *M. m. castaneus*. In order to consider the phylogenetic relationships within a species, subspecies, or population, one must allow for the possibility of interbreeding. Within an interbreeding population, an individual inherits from both parents. When we attempt to model this, we cannot construct a simple bifurcating tree as we have done so far. What we end up with is a web of inheritance relationships. I will discuss the factors affecting genomic inheritance within species and how I derive the local phylogeny structure under these conditions.

1.1.1 Thesis Statement

I introduce methods for efficiently and accurately identifying compatible genomic intervals describing local phylogenetic structure within species. These intervals represent unique local phylogenetic trees. I show how these intervals can be used to perform accurate genome-wide imputation and to inform genome-wide association studies. I describe the design and utility of visualization tools using local phylogenetic structure to allow effective and novel comparative analyses between individuals. Lastly, I introduce an extension of my model of compatible intervals which is resistant to erroneous data.

Allele	One of two or more alternative forms of a gene that arise by mutation and are found at the same place on a chromosome
Base-pair (bp)	A pair of complementary bases in a double-stranded nucleic acid molecule
Chromosome	The distinct molecular units containing all or most of the genetic material of cellular organisms
Diploid	Having two sets of homologous chromosomes
Genome	The genetic material of an organism
Genotype	The genetic constitution of an individual organism
Haploid	Having one set of distinct chromosomes, often one of each pair of homologous chromosomes
Haplotype	A group of alleles of different genes on a single chromosome
Heterozygous	Having dissimilar alleles at corresponding chromosomal loci
Homozygous	Having identical alleles at corresponding chromosomal loci
Phylogenetics	The study of evolutionary relatedness among groups of organisms
Recombination	The rearrangement of genetic material, esp. by crossing over in chromosomes
Single-Nucleotide Polymorphism (SNP)	Genetic variation in DNA where a single nucleotide is altered

Table 1.1: Definition of terms

1.2 Genetic Structure and Inheritance

1.2.1 Genomic Nomenclature

I will introduce a general notion of genetic structure, inheritance, and phenomena which affect them as it pertains to intraspecific phylogenetic structure. The mouse genome is approximately 2.5 billion base-pairs long. Each base on the primary strand consists of a single nucleotide, of which there are four possible choices, adenine (A), cytosine (C), guanine (G), or thymine (T). Within a species, only a very small fraction of base-pairs vary between individuals. We call these single-nucleotide polymorphisms, or SNPs. These and other commonly used genomic terms are defined in Table 1.1.

Genetic information is broken into separate functional units called chromosomes. Every mammalian cell (except reproductive cells) contains two copies of each chromosome, one

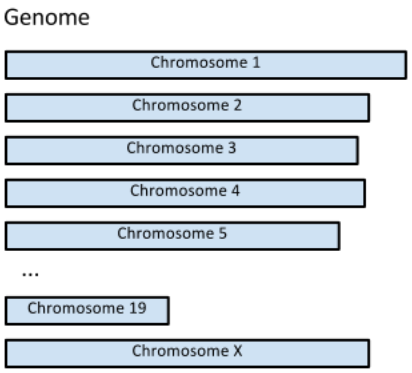


Figure 1.4: The genome is broken into blocks called chromosomes, each of which has two copies.

inherited from each parent. In mice, there are 20 pairs of chromosomes; 19 of them, numbered 1-19, are called autosomes and the last pair are the sex chromosomes, labeled X and Y (see Figure 1.4). As with humans, females have an XX pair and males have an XY pair. The collection of chromosomes makes up the genome of an individual. In naturally occurring sexually-reproducing populations, the two copies of each chromosome are different. One is transmitted from the mother and the other from the father.

1.2.2 Genetic Variation

One of the primary sources of genetic variation in sexually-reproducing species is mutation (see Table 1.1). Because mutation is very rare [53], I will make a common simplifying assumption known as the infinite-sites model [48]. This model states that mutation is sufficiently rare and the genome sufficiently long that no single position is likely to ever mutate more than once. I will evaluate the impact of this assumption and propose a relaxation in Chapter 5. Under this assumption, no SNP may include more than two alleles. As a result, I will frequently simplify the representation of a haplotype to include binary alleles, '0' and '1', '0' indicating the majority allele and '1' representing the minority allele or nucleic acid (see Figure 1.5). The infinite-sites model does not preclude the existence of heterozygosity, so we represent heterozygous loci with '2', meaning one copy contained '0' and the other

'1'. Violations of the infinite-sites model, where there have been multiple mutations at a single base position, are called homoplasy events. In Chapter 2, I will take further advantage of the infinite-sites model to make inferences about phylogenetic structure. We also often ignore everything but SNPs - that is, ignore all non-variable base-pairs - when comparing individuals.

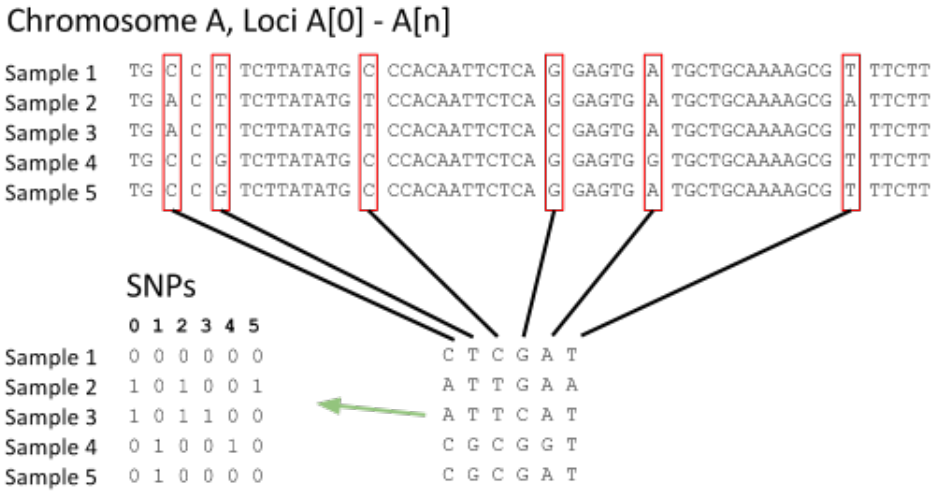


Figure 1.5: Reduction of genome sequences to binary single-nucleotide polymorphisms (SNPs). Most loci in the genome do not vary among individuals of the same species. Those which vary (are polymorphic), are called SNPs. Under the infinite-sites model, each SNP in a haplotype may have only two states, so we represent the majority allele with a '0' and the minority with a '1'.

A much more common event introducing genetic variation is recombination. Recombination is often the result of crossing-over during the formation of haploid reproductive cells. If we have two chromosomes A and B, with nucleotides $[A_0 \dots A_n]$ and $[B_0 \dots B_n]$, a recombination at locus r would result in $[A_0 \dots A_r B_{r+1} \dots B_n]$ and $[B_0 \dots B_r A_{r+1} \dots A_n]$. Recombinations are the major driving force for mixing heterozygous genotypes during inheritance. Recombination events cannot be represented in a simple bifurcating evolutionary tree since the resulting individual consists, in some part, of both parents' genotypes.

The model of local phylogeny that I describe enhances our ability to determine which

regions of the genome show no evidence of historical recombination among a set of individuals in a species. A set of genotypes admits a "perfect" phylogeny if a binary tree can be unambiguously constructed such that each branch represents a single SDP without homoplays. Since we can construct a perfect phylogenetic tree within these intervals, we can accurately describe the inheritance relationships within small segments of the genome, if not genome-wide. I demonstrate the utility and biological significance of these intervals along with useful visualization and analysis tools.

1.3 Compatible Intervals

In Chapter 2, I describe a method by which a set of genomes can be partitioned such that there is no evidence of historical recombination within each block, that is, they admit a perfect phylogeny. Although the discovery of perfect phylogenies is not new [71], I introduce an efficient method which defines an interval set with several desirable properties. This algorithm constructs an unambiguous set of intervals which is the smallest set necessary to cover the genome while each interval is maximal in size, called a Maximal-k-cover, or Max-k intervals. This set provides us with greater power to describe the phylogenetic structure within these intervals and identify likely inheritance relationships.

To identify regions with no evidence of recombination, I use a method call the four-gamete test (FGT) [40]. The four-gamete test states that, under the infinite-sites model, a pair of SNPs which are not separated by a recombination breakpoint should exhibit no more than three of the four possible allele pairs (gametes) among all samples. For example, if we have two SNP loci A and B, some number of samples n, where A_i and B_i represent the binary allele present in sample i at loci A and B, respectively, $FGT(A, B) = \{00, 01, 10, 11\} \not\subset \{A_0B_0, A_1B_1, A_2B_2, \ldots A_nB_n\}$. If a pair of SNPs passes the four-gamete test, we consider

them "compatible" with the same phylogeny. We can compute a pairwise matrix of four-gamete compatibility among all SNPs. Violations of the four-gamete rule imply a recombination or homoplasy event.

I describe the basic method - and computational complexity - for computing compatible intervals that can be applied to haplotype data. Haplotypes may be derived either from homozygous genotypes or heterozygous genotypes in which the component haplotypes are separated into distinct sequences - a procedure known as phasing. Many genotyping technologies, like the probe-hybridization microarrays used to generate the SNP data on which I demonstrate my method, cannot of themselves distinguish between the two copies of a chromosome. This results in heterozygous loci, 'H', where the two copies do not have the same allele, and it is unknown which haplotype (mother's copy or father's copy) contains which allele. Phasing methods attempt to infer the correct assignment of alleles to the appropriate haplotype, often by evaluating the co-occurrence statistics of nearby alleles within a population. Phasing methods may result in the two chromosome copies which are likely contributing to the genotype. In inbred mouse strains, this is often not a problem because the inbreeding has led to chromosomes which are nearly identical and there is no ambiguity. In other populations and species of interest, most notably humans, inbred genotypes are not available. To sidestep the often inaccurate process of phasing, I developed an extension of the four-gamete test which takes into account heterozygous 'H' calls. Due to the inherent ambiguity, I construct two sets of intervals depending on whether we take the optimistic view that all possibly compatible SNP pairs are probably compatible, or the pessimistic view where we consider all potentially compatible pairs incompatible.

Chapter 5 further extends this model to include a relaxation of the four-gamete rule which helps reduce the effect of the real-world complication of erroneous input genotypes. An application of my compatible intervals model used a probe-hybridization microarray, the Mouse Diversity Array (MDA), to collect genotype data over more than 500,000 SNPs known to

represent much of the diversity among classical laboratory mice [97]. This data allows us to distinguish differences in this mouse population and predict the local phylogenetic structure. However, with only half a million SNPs, there is an average genotype resolution of only one SNP every 5,000 bases (5 kb). I can harness a large collection of additional genotype data representing different technologies and analyses to improve the precision and accuracy of my phylogeny model. Specifically, I discuss the inclusion of SNPs derived from paired-end high-throughput short-read sequencing, or next-generation sequencing. These data provide a far greater density of genotype data than do microarrays. However, misalignments of short reads, homologous sequences, contamination, and sequencing error can all contribute to inaccuracies in these data. To include high-throughput sequence and other genotype information into my model, I introduce a relaxation of the four-gamete rule and the compatible intervals model that allows for a small fraction of violations (incompatibilities) to be overlooked as probable errors. Recombination produces an incompatibility signature distinctly different from other types of error, which my relaxed model is tuned to ignore.

1.4 Imputation using Local Phylogeny

DNA microarrays are an established technology for querying a genome for specific subsequences. They are especially effective for detecting the alleles of known single-nucleotide polymorphisms (SNPs). I describe a method to impute missing SNPs or other features using the inferred local phylogeny structure derived from microarray-based SNP data over a large set of individuals of a species. One can treat compatible intervals as regions of shared ancestral haplotypes, and their observed variants as alleles. I identified haplotype regions shared among classical laboratory mouse strains. The collection of haplotypes within an identified region implies a local phylogenetic structure that is useful for partitioning samples to a finer

degree than is possible with individual SNPs. Others have previously used blocks of contiguous SNPs as haplotypes to group samples locally [70]. The strength of our partitioning into haplotypes is that it is always consistent with a phylogenetic tree.

Haplotype structure can be used to accurately impute missing genotypes among closely related laboratory mouse strains [86]. I can confidently identify shared haplotype blocks among related individuals using a relatively low-density set of loci. This allows us to reliably predict that all intervening features are shared. I imputed 88 classical laboratory strains using this method and have shown that these imputed genotypes are more accurate than alternative imputation methods and exhibit an error rate approaching that of the sequencing technology itself.

1.5 Visualization of genomic structure

Visualization is essential to understanding of the structure and function of genomes. Genome-wide association relies on comparative analysis of closely related strains or individuals to determine the relationship between genotype and phenotype. Such analyses are informed by local haplotypes and phylogenetic structure. I have developed interactive visualization tools particularly well-suited for comparative analysis of genomic structure [87, 88]. These tools display SNPs, shared haplotype blocks, and subspecific origin over multiple collinear genomes, highlighting similarities and differences across the genome.

While existing genome browsers provide an effective interface to analyze genotype information and annotation for individuals, they lack the ability to visualize and analyze collinear genomes in an informative way. The framework I have developed supports the simultaneous visualization of multiple collinear genomes (for example, a variety of laboratory mouse strains). I introduce tools to support dynamic interaction and automatic comparison tools between genomes. These tools and datasets are derived from my model of compatible intervals

such that we can use regions of local phylogeny to compare samples based on their local inheritance structure. This type of interaction makes the phylogeny browser an excellent tool for analyses such as GWAS in which one would like to discover the relationship between regions of shared inherited genotypes and the phenotypes of a study sample.

My genome browser is provided as a web-based service, taking advantage of client-side as well as server-side computation to provide an interactive and widely accessible interface to this tool. Two instances of my browser have been deployed, one exposing the subspecific and phylogenetic structure of a large set of classical laboratory mice, the other describing the structure of the emerging Collaborative Cross [12] population including how the population has been derived from eight "founder" strains. These resources have seen effective and widespread use since their introduction.

1.6 Conclusions

In this thesis, I will describe my development of effective methods for decomposing genomes into meaningful blocks, referred to as *compatible intervals*, and placing them within the context of a local phylogeny. The point at which my approach departs from the classical model of evolution is that it attempts to infer the history for genomic segments rather than for an entire organism. The advantage of this approach is that it decouples genomic changes brought on by recombination from those originating from mutations. In Chapter 2, I describe my method for efficiently computing compatible intervals which are maximally informative over both haplotype and genotype data. In Chapter 3, I describe a method for genome-wide imputation using local phylogenetic structure and show that this method achieves a very high accuracy. I introduce a visualization tool in Chapter 4 which allows comparative analysis of multiple individuals within a species based on their local phylogenetic structure. Chapter 5 describes an extension of my local phylogenetic model to allow us to better handle haplotype

data with a higher rate of error. Finally, Chapter 6 discusses additional applications of my local phylogeny model and directions of future research.

Chapter 2

Compatible Intervals

I present methods for partitioning a genome into blocks within which there are no apparent recombinations. This provides parsimonious sets of compatible genome intervals based on the four-gamete test. My contribution is a thorough analysis of the problem of dividing a genome into compatible intervals, its computational complexity, and an achievable lower-bound on the number of intervals required to cover an entire genome [85]. In general, such minimal interval sets are not unique. However, I identify properties that are common to every possible solution. I also define the notion of an interval set that achieves the interval lower-bound, yet maximizes interval overlap. I demonstrate algorithms for partitioning haploid data, such as that derived from inbred mice. I will then describe how I extend this method to outbred, heterozygous genotype data using a modification of the standard four-gamete test. These methods allow our algorithms to be applied to a wide range of genomic data sets.

2.1 Introduction

The local block structure of genotypes within a population sheds light on many biological questions [19]. Genotype blocks are central to quantifying and localizing recombinations (both recent and historical) [76, 75, 89], are widely used to identify informative marker sets [101], and are building blocks for constructing genetic maps [73]. Genotype-block structure

also underlies many genome-wide association methods [100], provides evidence for selection [32], and offers a tool for ascertaining ancestral origins [18].

The task of decomposing a genome into meaningful blocks, however, has proven to be ill-defined, inconsistent, and often ambiguous [64, 70]. In part, the problem is due to the ad hoc definition of what constitutes a genotype block. Genotype blocks are often defined to serve a specific purpose. Examples include the minimum number of tagging SNPs sufficient to capture informative genotypes [65, 101], intervals of SNPs that exceed a given threshold of Linkage Disequilibrium (LD) [68], and maximal regions whose genotype diversity falls below a threshold [19]. Partitioning genotypes into blocks supporting perfect phylogenies [76, 31] and, similarly, the selection of blocks lacking evidence for recombination [91] are also used to construct Ancestral Recombination Graphs (ARGs).

I propose unambiguous definitions for haplotype and genotype blocks and efficient methods for computing them. Where ambiguity is unavoidable, I provide properties common to all solutions. My haplotype-block definition directly supports, and has been used for, association mapping [63], construction of genetic maps [103], and determining ancestral origins within local genomic regions [104]. My proposed genotype blocks can be used in much the same way.

Dense genotype data sets that are homozygous at every allele are readily available for many inbred mammal [28] and plant [13, 60] models commonly used for association mapping. However, haplotype data is not directly available for use in human studies. Using my approach, it is unnecessary to phase such data sets. I show how, using heterozygous genotypes, blocks can be identified for exploring the local phylogenetic structures [8, 46, 59, 61] and ancestral origins [99]. Like others [31, 91, 89], my blocks are chosen for their lack of historical recombination evidence.

My methods can be used as an alternative to other block methods such as those in [25, 102, 30]. Particularly, the genotype block methods I introduce may be used to inform

phasing [47] and feasibly extend these methods to be applicable to unphased genotype data. Block association methods such as Blossoc [57] and QBlossoc [6], which utilize small regions that admit perfect phylogenies, could potentially benefit from my methods to compute regional perfect phylogenies rather than single-marker phylogenies. These tools could also be extended to unphased genotype data rather than exclusively haplotype data.

I define blocks in terms of SNP compatibility according to the four-gamete test (FGT) [40]. The FGT is of interest because of its close relation to perfect phylogeny [44]. Specifically, a necessary and sufficient condition for a perfect phylogeny is that all pairs of SNPs satisfy the FGT [37]. For unphased genotype data, I further define the notion of optimistic and pessimistic compatibility based on if a region is *possibly* or *necessarily* passes the FGT. I partition the genome into a set of potentially overlapping maximal compatible intervals, each of which admits a perfect phylogeny, and whose union covers the full data set. I address the question of what is the fewest number of such intervals required and identify suspect SNPs whose removal reduces the overall complexity of the block structure (perhaps indicating genotyping errors, homoplasy, or gene conversions).

My contribution is an analysis of the problem of dividing a genome into compatible intervals based on genotypes and its complexity. I provide an achievable lower-bound on the number of such intervals. While in general there are numerous ways of dividing a genome into a minimum number of compatible intervals (a fact overlooked by others [57, 89, 92]), I also identify non-overlapping core subintervals common to all valid solutions. I define an interval set that achieves the interval lower-bound, yet maximizes the overlap between adjacent intervals, thus minimizing the number of perfect phylogeny trees, while providing the richest possible set of SNPs to each tree.

2.2 Related Work

There are three common approaches for partitioning haplotypes into blocks. The first employs LD measures [30, 68] and assigns blocks to regions with high pairwise LD within, and low LD between, blocks. A second class assigns blocks to regions of low sequence diversity [65]. Lastly, there are approaches that look for direct evidence of recombination, by either applying the FGT [40] and defining blocks as regions free of apparent recombination or homoplasy, or during the construction of ARGs, denoting supporting regions' component subtrees [76]. Schwartz et al. [70] performed an analysis of approaches and concluded that the block assignments of various methods differed markedly. Of these methods, the block boundaries of the FGT were better correlated to both the LD and diversity-based methods than these two methods were to each other.

My approach partitions the genome into blocks satisfying the FGT. This is not new. The seminal work of Hudson and Kaplan [40] provides a sketch of a greedy algorithm that processes SNPs in sequence order looking for runs of compatible intervals that are broken at points of incompatibility. This method is widely used [57, 70, 89, 92]. A disconcerting feature of this approach is that one arrives at a different interval set if the genome is scanned in the reverse order (Figure 2.4b). Alternative sets of compatibility intervals arise when the region is grown maximally around each SNP [57]. Moreover, there appear to be many other possible partitions, begging the question of which block sets have the fewest intervals, and, of these sets, which minimizes haplotype diversity. In my model, each block is compatible with a perfect phylogeny (a side effect of the FGT) and overlaps between adjacent intervals are allowed.

I extend these basic methods to unphased genotype data. Modern genotyping technologies often cannot distinguish between alleles in the genotypes of diploid individuals. There exist methods for computationally inferring the component haplotypes, called "phasing", but

these methods introduce significant inaccuracy [47]. Little work has been done on partitioning genotype data without first phasing; however, there has been considerable work on the related topic of phasing by perfect phylogeny [3, 10, 26, 34, 22]. Such methods determine if a given genotype block admits a perfect phylogeny. My contribution is to apply the basic insights of these methods to extend the notion of a haplotype "scan" to the genotype case. Similar work has been done [23] in which local phylogenies are built over unphased genotype data to inform association mapping; however, this work does not take full advantage of compatible blocks, employing a single-marker approach rather than a global block structure.

Past attempts at using perfect phylogeny to analyze genotypes assume they are given a region which admits a perfect phylogeny or does so within an error model. Most previous work ([22, 34, 26, 77, 3]) determines if the given set of genotypes does in fact admit a perfect phylogeny, and then solves the Perfect Phylogeny Haplotyping problem (PPH) [34]. In this context, "haplotyping" refers to phasing - determining the component haplotypes from a genotype. Recent extensions allow data to fall within some error model and handle cases where the data does not fit a perfect phylogeny. Error models include Missing Data (MD) and Character-Removal (CR), and the algorithms attain a global perfect phylogeny while dealing with erroneous point cases ([37, 35]).

No previous approach considers the possibility of different PPH solutions as determined by the choice of block partition. While I do not propose haplotyping by perfect phylogeny, I use related techniques to partition the genome into blocks which satisfy a perfect phylogeny that could, in practice, then be haplotyped using any one of several previous algorithms. Introducing a genome-wide approach to perfect phylogeny rather than filtering out data as in [37, 35] considers many biological factors previously overlooked. The notion of recombination-free blocks in the genome is well-documented in humans and mice, as well as other species [30, 102, 89, 63, 60]. In many cases, regions of the genome on either side

of a recombination point should realistically admit different phylogenies based on hybridization between subspecies. Simply removing presumed erroneous data and forcing regions separated by historical recombination into a global phylogeny ignores their biological relevance and produces a misleading solution. My method of partitioning allows for biologically meaningful, though limited, regions with which to perform further analyses.

2.3 Definitions

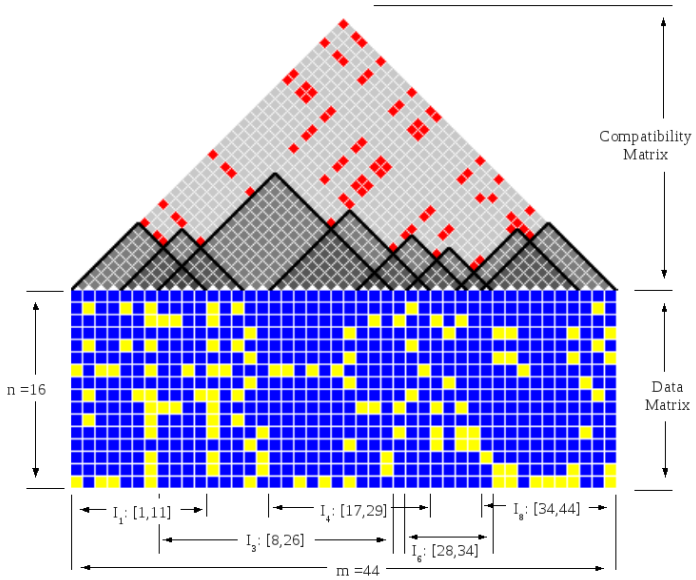
Throughout this discussion, we assume a data set of M SNPs spanning N haplotypes (or genotypes) that are represented as either a binary data matrix S or a ternary matrix S_g where each column corresponds to a SNP, and each row is a haplotype or genotype (Figure 2.1). Alleles 0 and 1 represent alternative homozygous alleles and 2 represents heterozygous alleles.

A compatible interval over a set of haplotypes is a sequence of contiguous SNPs over S for which there are no violations of the FGT between any SNP pair. A compatible interval, $I_x = [b_x, e_x]$, includes all SNPs between the beginning SNP s_{b_x} and ending SNP, s_{e_x}. Figure 2.1a shows a data set of 16 haplotypes and 44 SNPs, together with eight compatible intervals, $\{I_1 \ldots I_8\}$. Each interval covers a consecutive set of SNPs. For example, I_3 covers from s_8 to s_{26}. The triangular matrix above the SNP matrix is the pairwise compatibility matrix. If two SNPs exhibit four gametes, the corresponding matrix element is marked incompatible (red). Darkened triangles indicate sub-matrices corresponding to SNP pairs in the compatible intervals $\{I_1 \ldots I_8\}$. Note that no triangles enclose red elements.

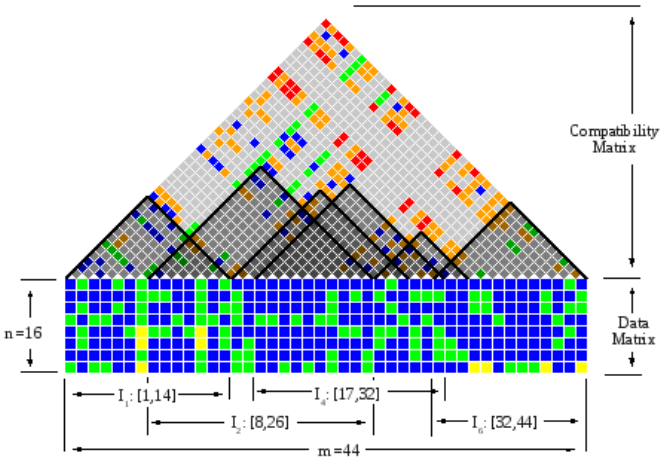
Compatible intervals over genotypes S_g are less straightforward due to ambiguities caused by heterozygous alleles. I define the notion of optimistic and pessimistic compatibility, whether genotypes are possibly or necessarily compatible, respectively. Resolving genotype intervals requires more considerations when performing the FGT. Pairs of SNPs are evaluated

to determine which gametes phasing could produce. In cases of homozygous-homozygous and homozygous-heterozygous pairs, the possible gametes are trivially determined. For example, the 0-0 produces the 0-0 gamete and 0-2 produces the 0-0 and 0-1 gametes. Ambiguity is caused only by the 2-2 case - when there exist heterozygous alleles in the same sample at two different loci. These cases can produce two different sets of gametes, either 0-0 and 1-1, which we call *consistent* gametes, or 0-1 and 1-0, which we call *inconsistent* gametes. The compatibility between 2-2 pairings can be categorized in three ways according to the restrictions necessary to make them compatible. If one of the 0-1 or 1-0 gametes are not present, all pairs must be *consistent*. If one of the 0-0 or 1-1 gametes are not present, all pairs must be *inconsistent*. If there are no other gametes present, all 2-2 pairs must simply produce the same set of gametes since it is always possible to produce four gametes with opposite phasings of two 2-2 pairs. These three states are indicated by green, orange, and blue, respectively, in the compatibility matrix (Figure 2.1b).

The optimistic algorithm forms a graph with a vertex representing each locus and an edge representing the relative phasing (consistent, inconsistent, or ambiguous) between two vertices. An interval is optimistically compatible iff there exists a bipartition of the graph into two sets A and B such that no edge within A is *inconsistent*, no edge within B is *inconsistent*, no edge between sets A and B is *consistent*, and all ambiguous edges are uniquely resolvable to as either in phase or out of phase. We use an algorithm similar to [26] to partition the genome into blocks of genotypes which admit a perfect phylogeny. Similar to the haplotype "scan", we introduce SNPs one-by-one and test whether the resulting interval is internally compatible. For haplotypes, this is accomplished by pairwise comparisons of previous SNPs with every newly introduced SNP. For the genotype case, we define two interval types. For *optimistic* intervals, we use the idea of what Eskin et al. [26] refer to as *equal* and *unequal* resolution to create a bipartite graph for each proposed interval. We "scan" SNPs as described in the haplotype case, adding these SNPs as vertices to the graph until the graph is no



(a) C_{Uber} cover over haplotypes



(b) C_{Uber} cover over diploid genotypes

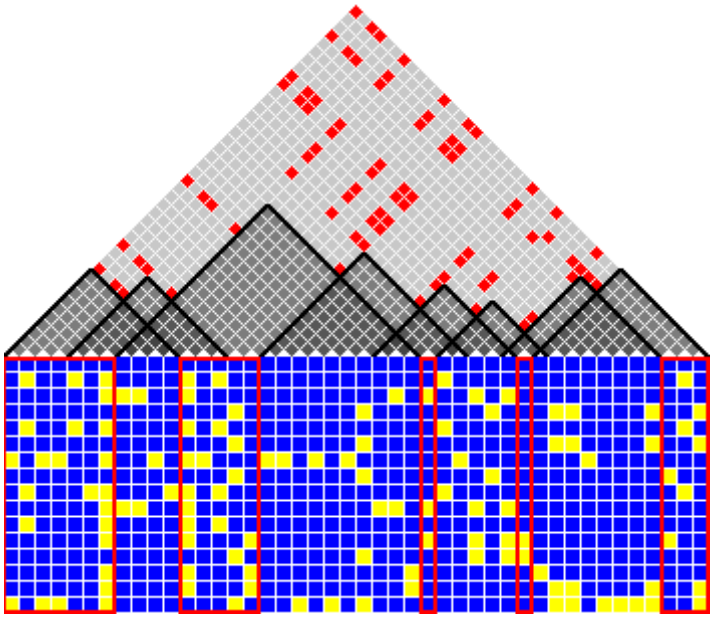
Figure 2.1: Example data sets and Uber-cover. The data sets used in this figure will be used as running examples. The lower portion of the figure is the data matrix. The columns correspond to SNPs, and the rows correspond to haplotypes (a) and genotypes (b). Blue and yellow boxes represent homozygous alleles normalized such that the first sample is homozygous blue. Green boxes represent heterozygous alleles. The large triangles above the data matrices are the compatibility matrices, showing the compatibility of each pair of SNPs. Gray boxes indicate that a pair of SNPs definitely does not violate the four-gamete rule. Red boxes indicate that a pair of SNPs creates four gametes. Green boxes indicate that a pair must be *consistent* to be compatible. Orange boxes indicate that a pair must be *inconsistent* to be compatible. Blue boxes indicate that a pair must be either *consistent* or *inconsistent* to be compatible.

longer realizable, thus ending an interval. *Pessimistic* intervals are unambiguously compatible regardless of the choice of haplotype phasing. When the scan is performed, an interval is ended as soon as it reaches a SNP which is possibly incompatible with any previous SNP in the interval. This is equivalent to considering all non-gray points as incompatible (red) and performing a haplotype scan to produce the pessimistic genotype intervals.

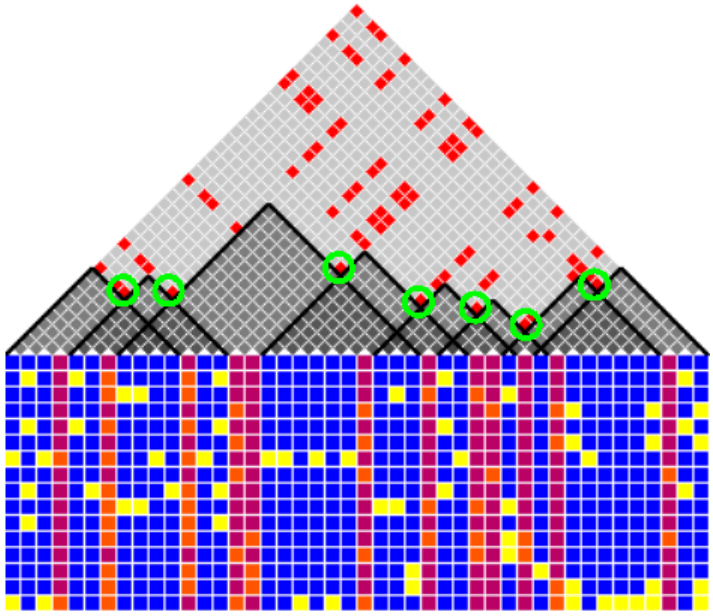
Figure 2.1b shows a data set of eight genotypes and 44 SNPs, together with four of its optimistic compatible intervals. It remains true that no interval may enclose an incompatible SNP pair. However, unlike the haplotype case, intervals are not necessarily bounded by red elements. As described, SNPs may be implicitly incompatible with a given interval if their addition forms an unrealizable graph.

A compatible interval is maximal if it cannot be extended in either direction. All intervals in Figure 2.1a (I_1, I_2, I_3, I_4, I_5, I_6, I_7, and I_8) are maximal, since further extension includes one or more incompatible SNP pairs. We denote the set of all maximal compatible intervals as C_{Uber}. Throughout, we will denote a cover over a genome generically by C. A cover of a set of haplotypes will be represented by $C(h)$. An optimistic cover of a set of genotypes will be represented by $C(g)$ and a pessimistic cover by $C(p)$. The darkened triangles in Figure 2.1a depict $C(h)_{Uber}$. The two SNPs adjacent to a maximal compatible interval, s_{b_x-1} and s_{e_x+1}, are the flagging SNPs of the interval (Figure 2.2b). Note that flagging SNPs are incompatible with at least one SNP of the maximal compatible interval that it flanks.

A *cover*, $C_{x,y}$, is an ordered set of intervals, $C_{x,y} = I_1, I_2, \ldots, I_c$, where $b_i \leq b_{i+1}$, and every SNP in the range $[x, y]$ is covered by some interval in C but no SNP outside $[x, y]$ is covered by any interval in $C_{x,y}$. $C_{x,y}$ also satisfies $e_i \leq e_{i+1}$, since otherwise I_{i+1} is a fully contained subset of I_i. We call $C_{1,m}$ a *complete cover* of S, and $|C|$ is its cardinality. I will frequently refer to a cover, C, where $|C| = c$, as a c-interval cover, or simply as a c-cover. In addition, I will refer to special instances of complete covers by using descriptive subscripts, in which case a range from $[1, m]$ is implied. For example, in Figure 2.1a, I_1, I_3,



(a) C_{Uber} with cores



(b) Flagging SNPs

Figure 2.2: This figure shows the C_{Uber} intervals. (a) includes the cores, highlighted in red. (b) shows flagging SNPs highlighted in red. Incompatibilities between flagging SNPs of adjacent maximal compatible intervals are highlighted with green circles.

I_4 is a $C(h)_{1,29}$ cover, $\{I_1, I_3, I_4, I_6, I_8\}$ is a complete 5-cover. We are particularly interested in complete k-covers, where k is a reachable lower bound on the number of intervals for the given SNP set.

In the following sections we provide an effective method for finding minimum-length complete covers for a given SNP set. This establishes k as a tight lower bound. In general, the k-cover for a given data set may not be unique. I provide several simple linear-time algorithms that generate various k-covers. In addition, we examine features which are common to all k-covers of a given data set. I then present a linear-time algorithm for finding a cover composed entirely of maximal compatible intervals from C_{Uber}, where $|C_{Uber}| \geq k$. Finally, I present an algorithm for finding the k-cover with maximal overlap, the Maximal-k-Cover (C_{Max}). The cover C_{Max} is of particular interest since it leads to the construction of a parsimonious set of perfect phylogeny trees where each incorporates maximal information (i.e. the maximum number of SNPs per tree). Finally, I present an algorithm for finding critical SNPs in S whose removal reduces $|C_{Max}|$ from k to $k-1$ or smaller. This can be accomplished using a number of tests proportional to $|C_{Uber}|$ rather than m.

2.3.1 Constructing Phylogenetic Trees from Compatible Intervals

Every compatible interval admits an unambiguous phylogenetic tree, by definition [33]. While we are often concerned only with the haplotypes represented within an interval, we also often interested in constructing the actual phylogenetic tree which represents the variation in an interval. This can be done simply and efficiently for binary haplotypes using the method described in [33]. We wish to construct a perfect phylogeny tree in which each edge represents an SDP in the compatible intervals. A rooted tree is constructed with the all-zero haplotype at its root (there may be no actual representative of this haplotype). SDPs are considered in decreasing order of the number of minority alleles present in all samples. For each unique SDP, the leaf containing the samples which have the minority allele is identified.

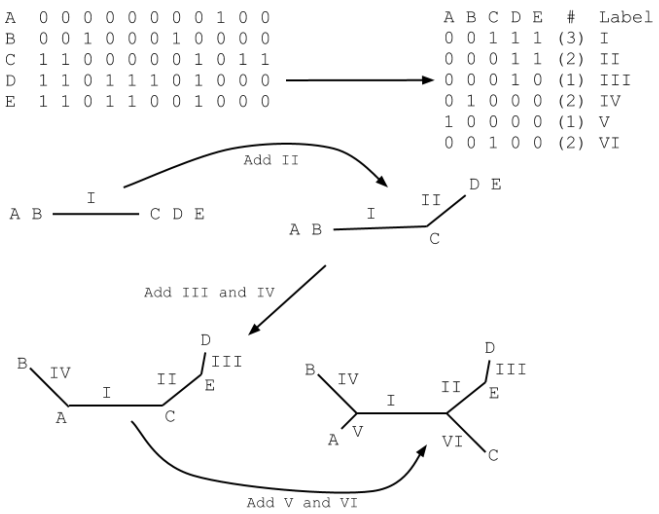
A subtree is added to this leaf containing those samples with 1s (the minority allele). Figure 2.3a illustrates this tree construction procedure. When constructing local phylogenetic trees in this way, the outgroup is most often unknown, so the resulting tree is considered unrooted after it is fully constructed.

It is not trivial to construct a meaningful tree over genotypes unless they are phased and the corresponding haplotypes admit a perfect phylogeny. However, when perfect phylogeny trees are unreasonable or we wish to construct trees over intervals which do not strictly admit a perfect phylogeny (in some circumstances, I consider the collection of several adjacent intervals), I use other tree construction methods. The neighbor-joining method [69] allows the efficient construction of parsimonious phylogenetic trees based on pairwise distances (in our case, SNPs). Given a population of samples and pairwise distances between them, we repeatedly merge the samples with the smallest distance (see Figure 2.3b). After merging two samples, we remove them from further consideration and add a virtual sample with distances to the remaining samples equal to the average distance from the two merged samples. These "merges" represent the roots of successive subtrees until all samples are merged into the root. This method accurately represents the relationships between samples when genotypes do not admit a perfect phylogeny [69].

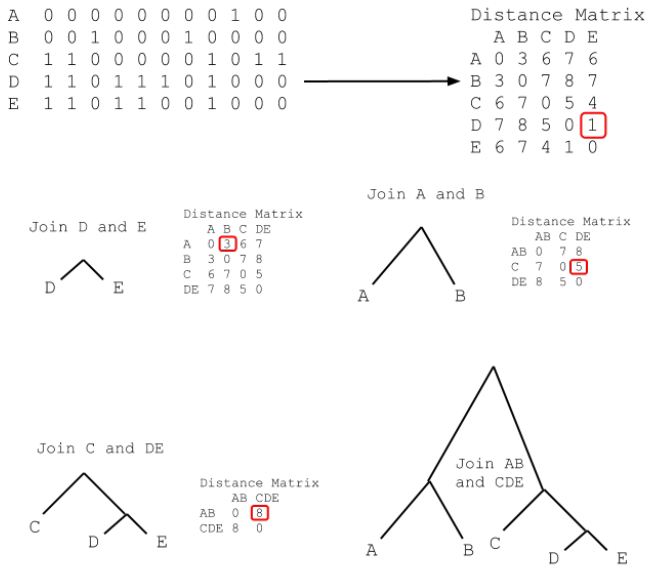
2.4 A Lower Bound

2.4.1 LR and RL Covers

We first define two non-overlapping covers, the Left-to-Right cover ($C(h)_{LR}$), and the Right-to-Left cover ($C(h)_{RL}$). A simple greedy algorithm, LRScan (Algorithm 1), which has been previously described in [89], finds $C(h)_{LR}$ over a set of haplotypes. It begins at the leftmost SNP (s_1), and either extends or terminates the current active interval as it considers each SNP in sequence order. LRScan performs FGTs of the candidate SNP against those SNPs



(a) Perfect Phylogenetic Tree Construction



(b) Neighbor-Joining Tree Construction

Figure 2.3: Construction of perfect phylogeny (a) and neighbor-joining (b) trees. Perfect phylogeny trees are constructed from a set of SNPs admitting a perfect phylogeny by partitioning the samples into subtrees/leaves strictly according the the distribution of alleles in each SNP. In (a), SDPs are used to partition samples in decreasing order of the frequency of the minority allele. Neighbor-joining trees do not require a perfect phylogeny. Leaves are successively merged based on a distance matrix between samples. In (b), leaves are merged at each step if they have the lowest distance in the distance matrix (circled in red).

already in the active interval. If four gametes occur between the SNP under consideration and a previous SNP found in the interval, the active interval is closed, and a new interval begins from the candidate, otherwise the SNP is added to the active interval. This continues until the last SNP is reached, thus closing the final interval (see Figure 2.4a).

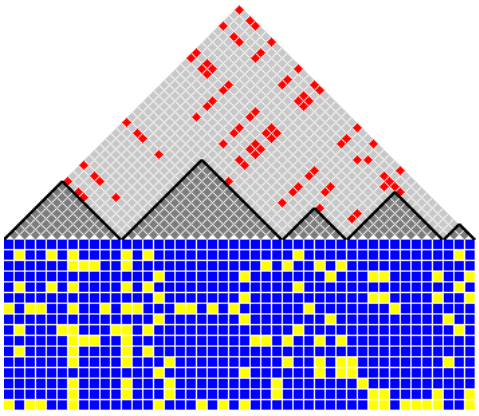
The run-time of LRScan depends on the number of SNPs, m, and the number of the FGTs performed for each SNP. Since the maximum number of distinct compatible SNP patterns that can be mutually compatible among n haplotypes is $2n - 3$ [77], the FGT requires only $O(n)$ operations per SNP, assuming a constant-time overhead for each FGT. Therefore, the complexity for LRScan is $O(mn)$, and thus is linear in the size of the data matrix.

Algorithm 1: $C(h)_{LR}$=LRScan(SNP)

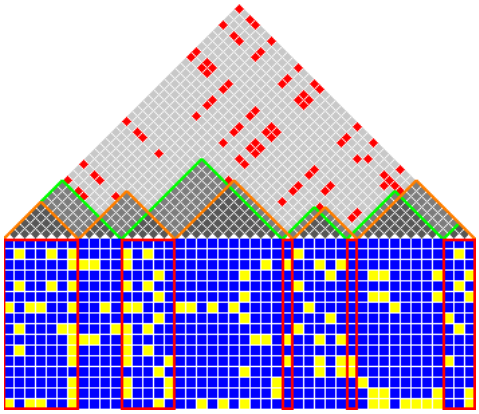
Input: $SNP = [snp_1, ..., snp_m]$: an array of m markers
Output: $C(h)_{LR}$: a list of intervals covering 1 to m
Variables: l : list of unique SNP patterns we have seen for the current interval
s : the start of the current interval

```
s ← 1
for i = 1 to m do
    if snp_i ∉ l then
        for j = 1 to ||l|| do
            if l_j is not compatible with snp_i then
                append to C(h)_LR interval [s, i − 1]
                s ← i
                l ← ∅
                break
    add snp_i to l
append to C(h)_LR interval [s, m]
return C(h)_LR
```

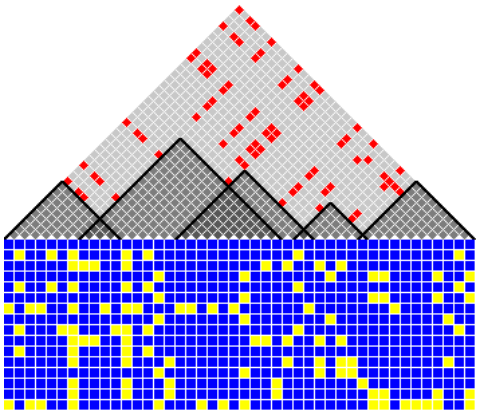
A similar greedy Right-to-Left scan algorithm (RLScan) generating $C(h)_{RL}$ can be defined via straightforward modifications to (LRScan). Likewise $C(h)_{RL}$ can be generated by merely reversing the input sequence, applying LRScan, and adjusting the indices of the resulting intervals, including their starting and ending positions. Note that a cover's interval ordering is defined consistently according to the sequence order, regardless of the scanning



(a) C_{LR}



(b) C_{LR} (green) and C_{RL} (orange)



(c) C_{Max}

Figure 2.4: This figure shows three covers. (a) depicts C_{LR} created by the LRScan algorithm. (b) shows C_{LR} (in green) and C_{RL} (in orange) together. The intersection of these two covers creates the set of cores (highlighted in red below the data matrix). In (c), each C_{Max} interval encloses exactly one core.

direction.

I define a similar notion over a set of genotypes. As described in Section 2.6, the only difference is the manner in which the FGT is performed. We find an optimistic left-to-right ($C(g)_{LR}$) and right-to-left ($C(g)_{RL}$) cover by closing an interval only when the subsequent SNP will be definitely and unambiguously incompatible with a SNP in the interval (regardless of the phasing chosen). Likewise, a pessimistic interval ($C(p)_{LR}$ and $C(p)_{RL}$) is closed off if there exists a phasing of the genotype set for which the next SNP will be incompatible with SNPs already in the interval.

The run time of the pessimistic genotype scan is also $O(mn)$. Like the haplotype case, there is a limit on the number of distinct SNPs that can be compatible among n genotypes which is linear in n. So, the adjusted FGT requires only $O(n)$ operations per SNP to determine if there exists any possible incompatibility.

The run time of the optimistic genotype scan is more complex. Eskin et al ([26]) propose an algorithm with $O(nm^2)$ complexity to determine if a single region admits a perfect phylogeny. I use a similar algorithm, adding SNPs incrementally. Since each interval is bounded and scan proceeds linearly across the genome, this allows for an $O(nm^2)$ algorithm to partition the entire genome.

2.4.2 Properties of C_{LR} and C_{RL}

In this section I present properties of compatible intervals and provide proofs. My first theorem states that "The covers C_{LR} and C_{RL} have the same number of intervals k, and k is the minimal number of intervals possible for any complete cover." A second theorem identifies certain core subintervals are common to all complete k-covers of a given SNP set (Figure 2.2a). These subintervals are the intersections of corresponding intervals from C_{LR} and C_{RL} which I have called cores (Figure 2.4b). This implies that a k-cover must contain k intervals each containing one core exclusively. This leads to two corollaries. The first is that

any interval that does not contain an entire core is not part of any complete k-cover and the second is that the ith core is only contained within the ith interval of any k-cover. Cores have several interesting properties worth noting. All SNPs in a core are compatible because each core is an intersection of two compatible regions. Adjacent cores must contain at least one pair of incompatible SNPs.

LEMMA 1. *Any c-interval cover covering the range $[1, e_c]$, with $e_c \leq m$, satisfies $e_c \leq e_c^L$, where e_c^L is the endpoint of the c^{th} interval of C_{LR} over the same sequence.*

PROOF. By induction, a single-interval cover must end at, or short of, e_1^L (an overage would indicate LRScan closed the interval prematurely). Assume that the Lemma holds for an i-interval cover, thus $e_i \leq e_i^L$. This implies for any $(i+1)$-interval cover $b_{i+1} \leq b_{i+1}^L$. We now prove $e_{i+1} \leq e_{i+1}^L$. Since there exists a SNP, s, in the $(i+1)$th interval of C_{LR} ($i.e.$ within $[b_{i+1}^L, e_{i+1}^L]$) where s is incompatible with $s_{e_{i+1}^L+1}$, $[b_{i+1}, e_{i+1}]$ cannot be a compatible interval if $e_{i+1} > e_{i+1}^L$. Therefore, we have $e_{i+1} \leq e_{i+1}^L$. □

A symmetric Lemma for C_{RL} states that any C-interval cover covering the range $[b_1, m]$, with $b_1 \geq 1$, satisfies $b_c \geq b_c^R$, where b_c^R is the start of the c^{th} interval of C_{RL} over the same sequence.

THEOREM 1. The covers C_{LR} and C_{RL} have the same number of intervals k, and k is the minimal number of intervals possible for any complete cover.

PROOF. Assume $\|C_{LR}\| = i$. According to Lemma 1, for any c-interval cover, C, with $c < i$ and starting from the left-most SNP, the end of the cover's range $e_c \leq b_c^L < b_i^L = m$. Therefore, C cannot be a complete cover if $\|C\| < \|C_{LR}\|$, implying that a complete cover must have at least $\|C_{LR}\|$ intervals. A similar conclusion can be drawn for $\|C_{RL}\|$ using the symmetric version of Lemma 1. Therefore, we have $\|C_{LR}\| = \|C_{RL}\| = k$, and k is the lower bound on the number of intervals for a complete cover. □

LEMMA 2. *For all* $i = 1 \ldots k$, $Core_i \neq \emptyset$.

PROOF. By induction. Assume $C_{LR} = \{I_1^L, \ldots, I_k^L\}$, and $C_{RL} = \{I_1^R, \ldots, I_k^R\}$. By definition $Core_i$ is $[b_i^L, e_i^R]$; therefore, it is sufficient to prove $b_i^L \leq e_i^R$. This is trivially the case for $i = 1$; $b_1^L \leq e_1^R$. Assume that $b_i^L \leq e_i^R$ holds for i, we now prove it holds for $i + 1$. From $b_i^L \leq e_i^R$, we know $b_i^L < b_{i+1}^R$. If $b_{i+1}^L \leq e_{i+1}^R$ does not hold, implying $e_{i+1}^R \leq e_i^L$, then we know $I_i^L \supset I_{i+1}^R$, and SNP $s_{e_i^R}$ ($\in I_i^L$) must be compatible with all SNPs in I_{i+1}^R. Figure 2.5b illustrates this proof. $s_{e_i^R}$ is outside and adjacent to I_{i+1}^R. By definition of the Right-to-Left cover, SNP $s_{e_i^R}$ must be incompatible with at least one SNP in I_{i+1}^R, which results in a contradiction. □

THEOREM 2. *For any complete* k*-cover* $C_{1,m}$= $\{I_1, \ldots, I_k\}$*, the* i*th interval contains the entire* i*th core:* $Core_i \cap I_i = Core_i$*, and, it does not contain any part of another core* $Core_j \cap I_i = \phi$, $1 \leq j \leq k, j \neq i$.

PROOF. Since $Core_i = [b_i^L, e_i^R]$, to prove the ith interval $I_i = [b_i, e_i]$ contains $Core_i$, it is sufficient to prove $b_i \leq b_i^L$ and $e_i \geq e_i^R$. Since $\{I_1, \ldots, I_{i-1}\}$ is an $(i-1)$-cover covering the range $[1, e_{i-1}]$ starting from the leftmost SNP, according to Lemma 1, we have $e_{i-1} \leq e_{i-1}^L$, which means $b_i \leq b_i^L$. Similarly, we can prove $e_i \geq e_i^R$ (see Fig. 2.5c). To prove $Core_j \cap I_i = \emptyset$ for any other core $Core_j$, it is sufficient to prove $e_i < b_{i+1}^L$ and $b_i > e_{i-1}^R$. Since $\{I_1, \ldots, I_i\}$ is an i-cover covering the range $[1, e_i]$, according to Lemma 1, we have $e_i \leq e_i^L < b_{i+1}^L$. Similarly, we can prove $b_i > e_{i-1}^R$. □

Stated formally, $Core_i = I_i^L \cap I_i^R$. According to Lemma 1, $e_i^R \leq e_i^L$ and $b_i^L \geq b_i^R$ (Figure 2.5a), therefore $Core_i = [b_i^L, e_i^R]$. Theorem 2 states, "For any complete k-cover, $C\{1, m\} = \{I_1, \ldots, I_k\}$, the ith interval contains the entire ith core: $Core_i \cap I_i = Core_i$, and, it does not contain any part of another core $Core_j \cap I_i = \Phi, 1 \leq j \leq k, j \neq i$". This is due to the interleaving of the non-overlapping intervals of C_{LR} and C_{RL}. $Core_i$ is necessarily compatible with both I_i^L and I_i^R and cannot be extended beyond the outside boundary of either. Therefore, no part of any two cores may be a part of the same interval.

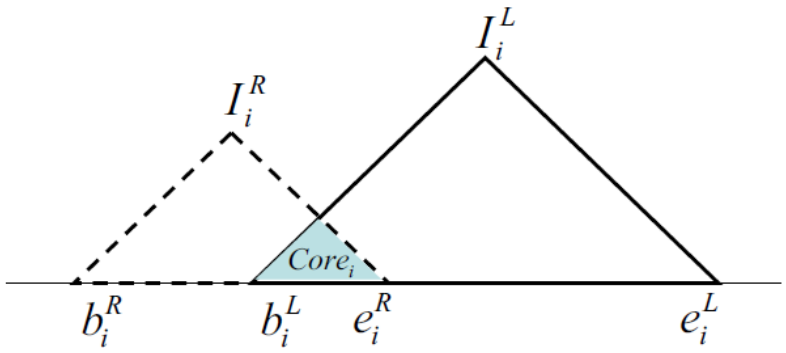
PROPERTY 1. *All SNPs in a core are compatible, since each core is an intersection of two compatible intervals*

PROPERTY 2. *Adjacent cores must contain at least one pair of incompatible SNPs*

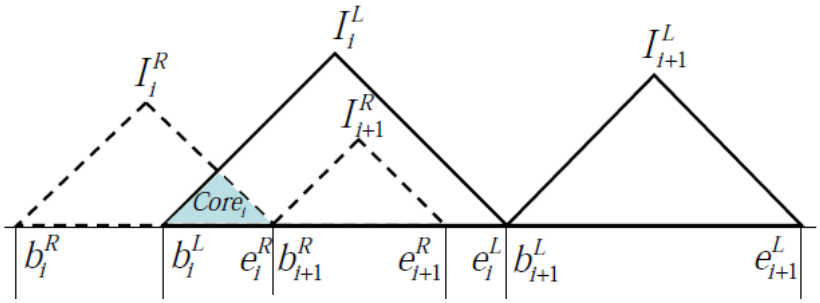
PROOF. This can be demonstrated by contradiction. Assume that two adjacent cores, $core_i$ and $core_{i+1}$, are compatible. By definition, $core_i = [b_i^L, e_i^R]$. Since $core_i \subseteq I_i^L = [b_i^L, e_i^L]$, $core_i$ is compatible with $[e_i^R + 1, e_i^L]$. Similarly, $core_{i+1}$ is compatible with $[e_i^R + 1, e_i^L]$. Therefore, $core_i \cup [e_i^R + 1, e_i^L] \cup core_{i+1} = [b_i^L, e_{i+1}^R]$ is a compatible interval containing both cores and contradicting Theorem 2 (Fig. 2.5d). □

2.5 Max-k Interval Set

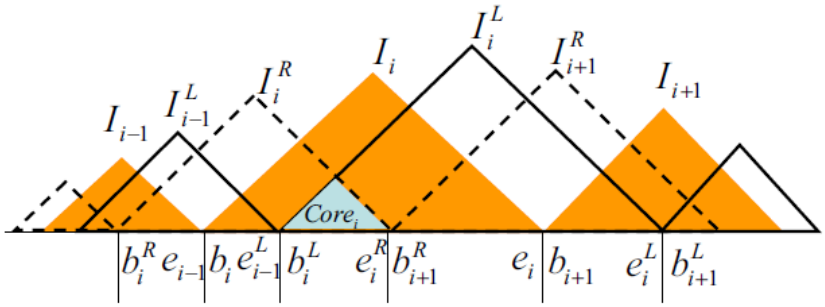
First we introduce UberScan, which generates the set of all the maximal compatible intervals, C_{Uber}. UberScan, shown in Algorithm 2, is similar to the LRScan. Whenever a compatible interval ends at SNP s_i, instead of starting the next interval from $s_i + 1$ as LRScan does, UberScan finds the nearest SNP $s_j (j < i + 1)$ that is incompatible with $s_i + 1$, and the following SNP, $s_j + 1$, begins the next interval. Note that $s_i + 1$ is a flagging SNP of the previous maximal compatible interval and s_j is a flagging SNP of the next maximal compatible interval. UberScan is a simple modification of LRScan with added bookkeeping to track of the index of the last occurrence of each unique SNP pattern. Recall that the maximum number of distinct SNP patterns within a compatible interval is $2n - 3$, or $O(n)$. An analysis similar to that of LRScan shows that UberScan also takes $O(mn)$ time. UberScan generates C_{Uber}, containing all maximal intervals of S, and generally, $|C_{Uber}| \gg k$. C_{Uber} contains all candidates for the Maximal-k-cover, C_{Max}, since a cover with maximal overlap must be composed of maximal intervals.



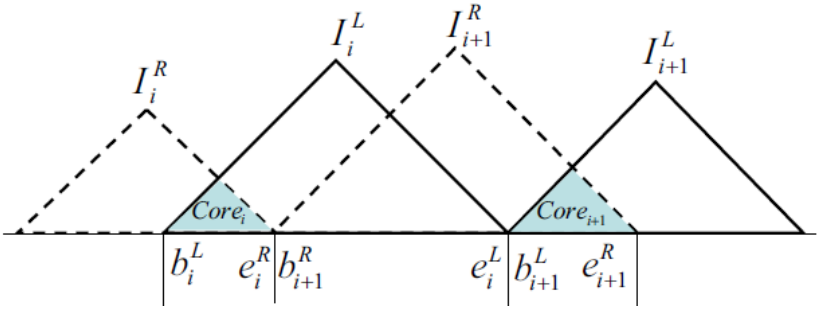
(a) Shows the definition of core ($Core_i = I_i^L \cap I_i^R$)



(b) Illustration of the proof for Lemma 2

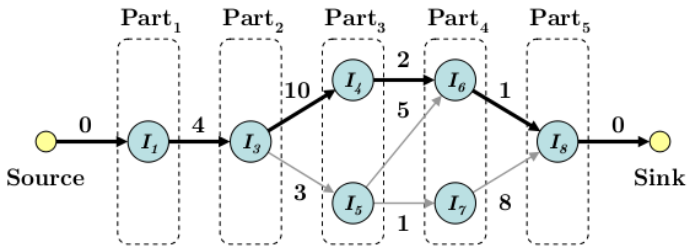


(c) Illustration of the proof for Theorem 2

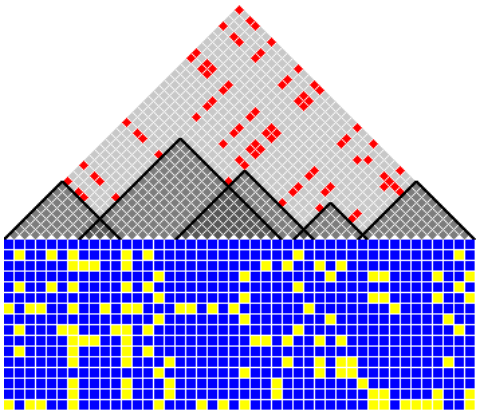


(d) Illustration of the proof for Property 2

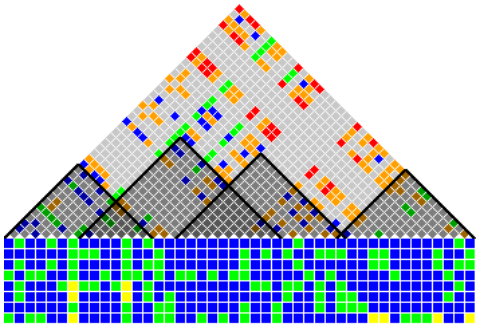
Figure 2.5: Illustration of proofs



(a) Max-k graph



(b) C_{Max}



(c) $C(g)_{Max}$

Figure 2.6: (a) shows the k-partite graph used to find C_{Max} for the running example. The node represents the interval, and the edge connects intervals which overlap, with weight representing the number of shared SNPs. The longest path (bold) is computed from source to sink which is C_{Max}. It contains intervals I_1, I_3, I_5, I_6, and I_8 from C_{Uber}, with a total overlap of 17. (b) is the Maximum-k-cover of the set of haplotypes, $C(h)_{Max}$. (c) is the optimistic Maximum-k-cover of the set of genotypes, $C(g)_{Max}$.

Algorithm 2: C_{Uber}=UberScan(SNP)

Input: $SNP = [snp_1, ..., snp_m]$: an array of m markers
Output: C_{Uber} : a list of intervals covering 1 to m
Variables: l : sorted list of (position, SNP) tuples for the SNPs we have seen for the current interval
s : the start of the current interval

```
s ← 1
for i = 1 to m do
    for j = ||l|| downto 1 do
        if l_j is not compatible with snp_i then
            append to C_Uber interval [s, i − 1]
            s ← l_{j_position} + 1
            remove l_1 to l_j from l
            break
    append snp_i to l
append to C_Uber interval [s, m]
return C_Uber
```

2.5.1 Finding the Maximal-k-cover

A Maximal-k-cover, C_{Max}, is of particular interest as it covers the entire SNP set using the fewest, k, maximal intervals. While, C_{Max} is not necessarily unique, alternate solutions are generally similar. Next we provide a fast graph algorithm to compute all Maximal-k-covers.

We consider only those maximal intervals in C_{Uber} that entirely enclose a single core and no part of a second core. According to Theorem 2, these intervals are the candidates for Maximal-k-covers. Next, we organize the candidates into k groups according to the core it contains. Each core is contained within at least one maximal interval, thus, no group is empty. We then examine the overlap between groups. A candidate interval in group i can only overlap candidates from groups $i - 1$ or $i + 1$, because, if any two candidates enclose non-adjacent cores (say $Core_x$ and $Core_y$), at least one of them contains part of the core between $Core_x$ and $Core_y$, contradicting Theorem 2.

The Maximal-k-cover problem is solved by recasting it as finding the longest path in a directed k-partite graph (Figure 2.6a). Specifically, each candidate maximal interval is mapped

to a node and each group as a part, with part i containing all the candidates covering $Core_i$. An edge connects nodes corresponding to overlapping candidates. Each edge's weight is the amount of overlap between the two intervals. The edge is directed towards the candidate that contains the next core in the sequence. Because intervals only overlap adjacent groups, edges only exist between adjacent parts. A source is added with edges to all nodes in part 1 and a sink with edges from all the nodes in part k; both types of edges have weight 0. Finding a C_{Max} solution corresponds to finding the longest path in this directed graph with $k+1$ edges from source to sink. Note that the greedy approach of taking the largest interval that encloses each core does not always yield a correct answer, as shown in the fourth core of our running example (compare Figure 2.2a and Figure 2.4c).

The problem is a single-source shortest path problem for a weighted directed acyclic graph (DAG), except that we search for longest path (maximizing instead of minimizing the sum of weight on the path) with a constraint on the number of steps. The constraint can be ignored since all edges lead from one part to the next, thus any path from source to sink will have $k+1$ steps. The problem can be solved using dynamic programming and requires only $\Theta(|V|+|E|)$ time, where V is the set of nodes, and E is the set of edges [15].

2.5.2 Critical SNPs

A critical SNP is any SNP whose removal reduces k, the minimum number of intervals required to cover the given SNP set. To check whether a SNP is critical, one could simply remove each SNP and recalculate k by either a LRScan or an RLScan. This naive approach requires m scans, and takes $O(nm^2)$ time. However, it is unnecessary to test every SNP. In fact, only flagging SNPs of maximal compatible intervals need to be considered. A flagging SNP bounds an interval on one side and prevents the interval from growing toward an adjacent interval on that side, therefore a flagging SNP must be removed to allow any interval to grow, a necessary condition of reducing k.

THEOREM 3. *Critical SNPs are flagging SNPs of C_{Uber} intervals.*

PROOF. Consider two adjacent maximal compatible intervals, I_i^{Uber} and I_{i+1}^{Uber}. Their flagging SNPs $s_{e_i^{Uber}+1}$ and $s_{b_{i+1}^{Uber}-1}$ must be incompatible (see Fig. 2.2b for an example). This incompatibility makes it infeasible for e_i^{Uber} to be larger and b_{i+1}^{Uber} to be smaller. Without removing at least one of these two flagging SNPs, it is impossible for either of these two intervals to grow towards each other. Since $C_{Max} \subseteq C_{Uber}$, a necessary condition of reduction in k is that at least one interval in C_{Uber} grows in size. Therefore, being a flagging SNP is a necessary condition for being a critical SNP. □

Since each maximal compatible interval has two flagging SNPs, one on each side of the interval, the total number of flagging SNPs is $O(|C_{Uber}|)$. The running-time for computing critical SNPs is $O(nm|C_{Uber}|)$.

2.6 Genotypes

Determining four-gamete compatibility and compatible intervals over unphased genotype data is ambiguous in that there are many possible interpretations (phasings) of a set of genotypes as haplotypes. I define two approaches for determining compatibility among genotypes without explicitly phasing. The optimistic method determines intervals for which there might exist a phasing such that the interval is four-gamete compatible. The pessimistic method determines intervals by choosing a phasing which introduces the maximum possible incompatibility.

I compared the haplotype and genotype intervals on three data sets. First, I created simulated genotype data using a simple infinite-sites model of mutation with cross-over recombination - this served as a data source devoid of confounding factors such as experimental error and homoplasy. Second, I used real data from F1 crosses between isogenic mouse strains. Last, I used two populations of HapMap data.

2.6.1 Relating Genotype and Haplotype Covers

In simulated data, one can explore relationships between the compatible intervals of genotypes and the compatible intervals of their "source" haplotypes. These haplotyeps can be used as ground truth for the corresponding set of genotypes. The number of intervals in an optimistic genotype cover, $||C(g)_{LR}||$, is less than or equal to the number of ground truth intervals (Theorem 4) and the size of the pessimistic genotype cover, $||C(p)_{LR}||$, is greater than or equal to the number of ground truth intervals (Theorem 5). Thus, the number of intervals in an optimistic and pessimistic genotype scan are lower and upper bounds, respectively, on the true number of intervals.

THEOREM 4. $\|C(g)_{LR}\| \leq \|C(h)_{LR}\|$

PROOF. By contradiction. Assume $C(h)_{LR}$ exists such that $\|C(h)_{LR}\| < \|C(g)_{LR}\|$. There must exist some incompatibility in S_g not in S. By definition, it must be impossible to phase S_g to make S. Therefore, S must not be a true phasing of S_g. □

THEOREM 5. $\|C(h)_{LR}\| \leq \|C(p)_{LR}\|$

PROOF. By contradiction. Assume $C(h)_{LR}$ exists such that $\|C(p)_{LR}\| < \|C(h)_{LR}\|$. There must exist some incompatibility in S not in S_p. By definition, it must be impossible to phase S to make S_p. Therefore, S_p must not be a true phasing of S. □

COROLLARY 1. $\|C(g)_{LR}\| \leq \|C(h)_{LR}\| \leq \|C(p)_{LR}\|$

In Figure 2.7, 'a' represents the distribution of the set of all possible genotype "pairings" of a fixed haplotype set into genotypes. The "ground truth", or the number of intervals required to form a cover using the source haplotypes, is 5. Notice that the covers resulting from every optimistic genotype scan fall closer to the "ground truth" than those from every pessimistic scan. From our experimental results, we observe this is commonly the case.

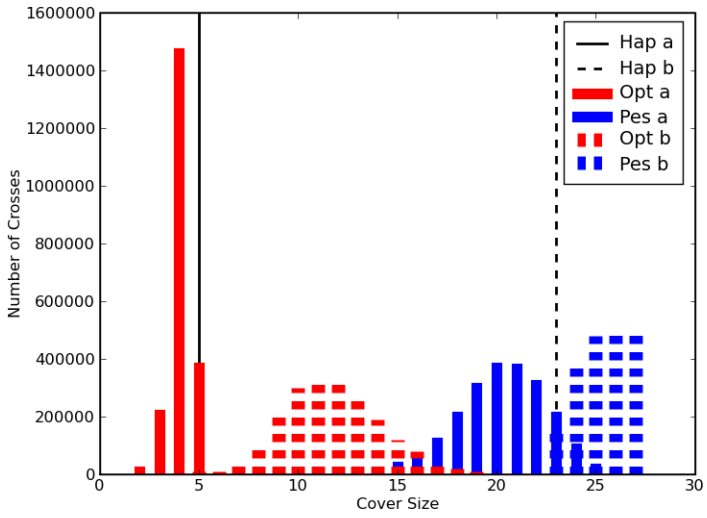


Figure 2.7: The distribution of cover sizes for genotypes resulting from all pairings of a set of haplotypes. For 'a' distributions, data was simulated using the infinite sites model and recombination. The source haplotypes cover size (the "ground truth") is represented by the solid black line. For 'b' distributions, a contrived haplotype set was made by phasing a pessimistic genotype result of 'a'. The source haplotypes cover size is the dashed black line.

In contrast, 'b' represents the same plot for a contrived, non-biologically-based, haplotype data set. The distribution of the cover size of all genotypes can be formed by pairings of the haplotype set that achieves one of the pessimistic covers from 'a' (this can always be achieved, as discussed in Section 2.6.2). Specifically, the "true" cover size for this contrived set is 23. Notice that the distribution is different from the biology-based model. In particular, the optimistic and pessimistic distributions are closer together and the "ground truth" is nearer the pessimistic estimations.

2.6.2 Achieving Genotype Covers by Phasing

In many circumstances, it is useful to determine if a particular genotype cover or interval set is achievable by phasing. Pessimistic genotype covers can always be achieved (see Algorithm 3). However, there does not always exist a phasing to accomplish a given optimistic genotype interval. In practice, many candidate covers can be proposed and it is trivial to

verify candidate intervals using existing PPH methods ([34, 26, 3, 10]).

2.7 Experiments and Results

We tested the performance of our algorithms on real datasets. The first is based on 8 inbred mouse strains and selected F1 (first generation) crosses between these strains using a newly developed genome-wide genotyping platform ([99]). The second and third are human datasets from the International HapMap Project [83]. The first HapMap dataset describes a population of Utah residents from northern and western Europe (CEU). The second is a Yoruba population from Ibadan, Nigeria (YRI).

The mouse genome contains 20 chromosomes (chromosome 1-19 and chromosome X). The number of SNPs per chromosome in our MDA genotypes varies from 15K to 50K and includes strains: 129SvlmJ, A/J, C57BL/6J, CAST/EiJ, NOD/LtJ, NZO/HILtJ, PWK/PhJ, and WSB/EiJ along with 37 F1 crosses. The inbred founder mouse strains were used in the haplotype analysis, while the F1 crosses were used in the genotype analyses. With this mouse data set, we are able to ascertain an approximate ground truth, sans genotyping errors, with real-world rather than simulated data. SNPs discriminating among the 8 strains were used in our analysis, reducing the 500K total SNPs to 340K.

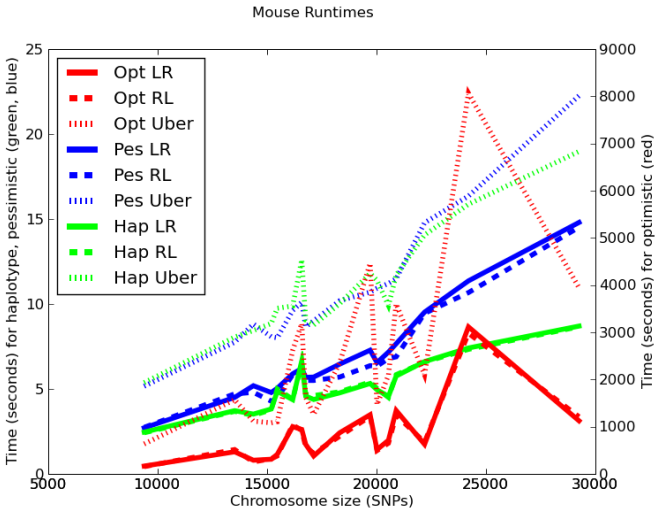
I found compatible intervals for 23 human chromosomes from the phased HapMap data (Chromosome 1-22 and Chromosome X). The CEU dataset has 348 haplotypes (174 individuals) with 34K - 222K SNPs per chromosome and the YRI dataset has 348 haplotypes (174 individuals) with 38K - 252K SNPs per chromosome. In my experiement, also used SNPs imputed as a result of the HapMap phasing method. The final phased CEU data set has 2.6M SNPs and YRI has 2.9M SNPs.

Our algorithms were implemented in Python 2.6 and experiments were performed on a 2.67GHz Intel Core i7 processor with 8.0GB of RAM.

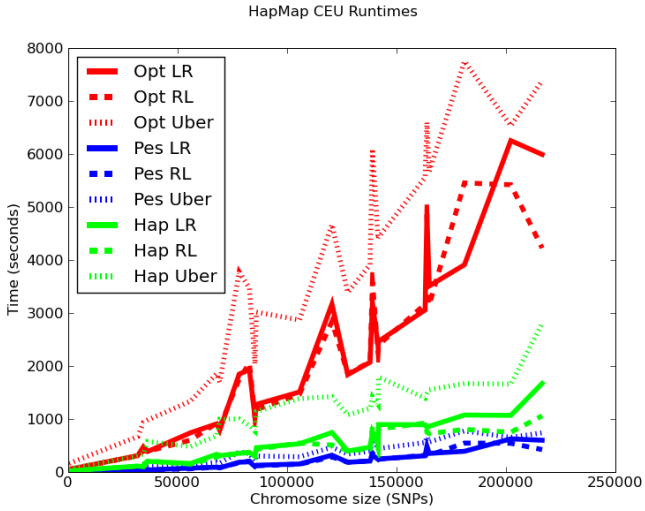
Algorithm 3: S=PGPhase(Sg)

Input: $S_g = [snp_1, ..., snp_n]$: an array of n SNPs with m markers,
$C(p) = [(start_1, end_1), ..., (start_n, end_n)]$: an array of intervals
Output: S : an array of n SNPs with $2m$ markers, initially empty

```
for i = 1 to n do
    i1 = 2i, i2 = 2i + 1
    for j = 1 to ||C(p)|| do
        if i == end_j + 1 or i == start_j − 1 then
            for l = start_j to end_j do
                l1 = 2l, l2 = 2l + 1
                if S_g[i] is consistent with S_g[l] then
                    for k = 1 to m do
                        if S_g[i][k] == 2 and S_g[l][k] == 2 then
                            S[i1][k] = S[j2][l]
                            S[i2][k] = S[j1][l]
                            break
                else if S_g[i] is inconsistent with S_g[l] then
                    for k = 1 to m do
                        if S_g[i][k] == 2 and S_g[l][k] == 2 then
                            S[i1][k] = S[l1][k]
                            S[i2][k] = S[l2][k]
                            break
                else
                    phase = 0
                    for k = 1 to m do
                        if S_g[i][k] == 2 and S_g[l][k] == 2 then
                            if phase == 0 then
                                S[i1][k] = S[l1][k]
                                S[i2][k] = S[l2][k]
                                phase = 1
                            else
                                S[i1][k] = S[l2][k]
                                S[i2][k] = S[l1][k]
                                break
                            break
            break
    for k = 1 to m do
        if S_g[i][k] == 2 then
            S[i1][k] = 0
            S[i2][k] = 1
        else
            S[i1][k] = S_g[i][k]
            S[i2][k] = S_g[i][k]
return S
```



(a) Mouse haplotype and genotype run-times



(b) HapMap CEU cover run-times

Figure 2.8: Run-times to calculate covers over real data sets. (a) shows C_{LR}, C_{RL}, and C_{Uber} run-times versus the number of size of chromosome in SNPs for haplotype, optimistic genotype, and pessimistic genotype covers over the mouse data set. (b) similarly depicts run-times of the HapMap CEU population.

2.7.1 Run-times

The performance of interval scanning algorithms (LRScan, RLScan and UberScan) is linear in the data size, which enables us to compute C_{LR}, C_{RL}, and C_{Uber} efficiently. The run-times of all three scans and the Max-k-cover algorithm was recorded for all the data sets. Figure 2.8a gives the times for calculating the haplotype covers $C(h)_{LR}$, $C(h)_{RL}$, $C(h)_{Uber}$, and $C(h)_{Max}$ on the mouse data set (inbred founders) and times for calculating genotype covers for the F1 crosses. Figure 2.8b shows run times for all covers over the HapMap data sets. As shown, the run-times for all scan types are linear in the number of SNPs for a fixed population size. In addition, the Max-k-cover finding is much faster than all three scans. Since the LRScan and RLScan are symmetric procedures, they have similar run-times. UberScan involves more bookkeeping and, thus, has higher times than the other scans. The optimistic genotype scan times (Figure 2.8) are much higher than the haplotype and pessimistic scans due to the computationally intensive graph algorithm which must be performed at each step.

The k-partite graph component of the Max-k-cover algorithm takes the intervals of C_{Uber} as input. Figure 2.10b shows the run time of the Max-k-cover algorithm as a function of the number of SNPs. The pessimistic genotype scans admit substantially intervals to achieve a cover, consistent with the linear relationship between the run time of the Max-k-cover algorithm and the cardinality of C_{Uber}.

2.7.2 Interval and Core Statistics

We also collected various statistics over the C_{Max} intervals and cores, including interval lengths in terms of SNPs and genomic position, and the number of distinct haplotypes. The interval lengths shown in Figure 2.9 illustrate the prevalence of many long conserved regions punctuated by smaller intervals indicative of recombination hot spots. Numbers of distinct haplotypes (Figure 2.10c) indicate the relative diversity within each block. Note that the maximum number of distinct haplotypes is $Min(n, s + 1)$, where s is the number of unique

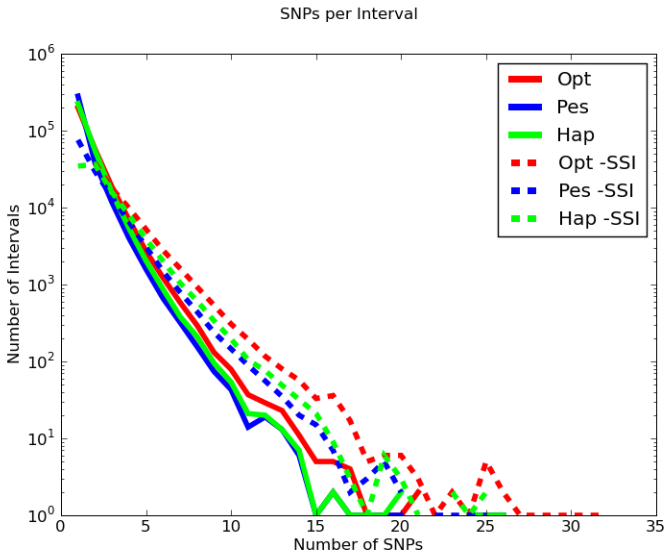
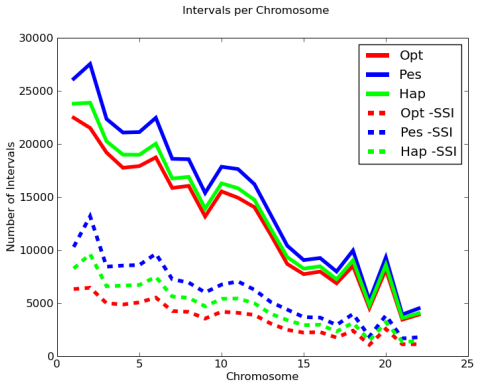


Figure 2.9: Interval size statistics for the CEU human population. A distribution of the number of SNPs per interval over this data set under optimistic, pessimistic, and haplotypes scans. "-SSI" indicates a similar distribution after the removal of single-SNP intervals (SSIs).

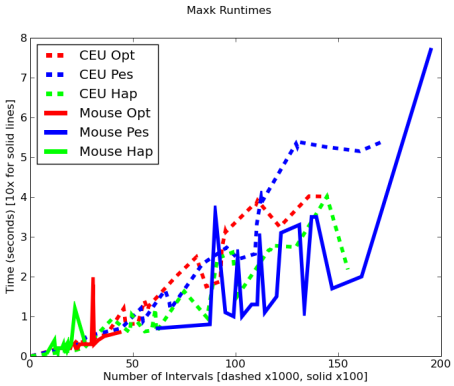
SNPs in the interval.

A large percentage of Max-k intervals in the optimistic genotype cover of the human CEU data set contain only a single SNP. By definition, these are Critical SNPs - removing one reduces the total number of intervals k by at least one. Such one-SNP intervals, which are incompatible with SNPs immediately adjacent, are not likely to be informative regarding recombinations and probably represent other biological or experimental artifacts such as genotype errors, homoplasy, or gene conversions. This explains the prevalence of cores with two unique haplotypes as shown in Figure 2.10c, and the large number of single-SNP cores, as shown in Figure 2.9. Figure 2.9 also includes the distribution of interval sizes in SNPs of the Max-k cover after the SNPs making up these single-SNP intervals have been removed. Notice that the average interval size is greater for all three cover types after removing these data.

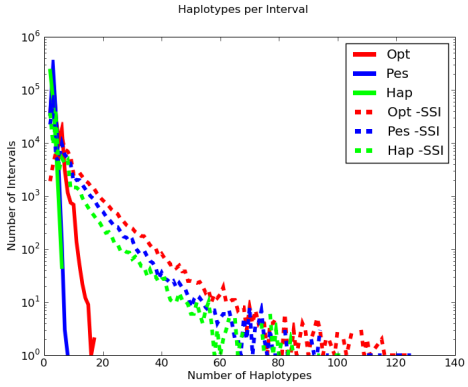
The relationship between the haplotype covers and the optimistic and pessimistic genotype covers is illustrated in Figure 2.10. Figure 2.10a shows the number of intervals in a



(a) Max-k cover sizes per chromosome.



(b) k-partite graph run times



(c) Unique Haplotypes per Interval

Figure 2.10: (a) shows the number of intervals required to form a k-cover over each chromosome before and after removing the SNPs composing single-SNP intervals (SSI) in the optimistic genotype cover. (b) shows the run times of the k-partite graph component of the Maximum-k-cover algorithm for the haplotype, optimistic genotype, and pessimistic genotype intervals for all three data sets. The algorithm is linear in the number of Uber intervals $|C_{Uber}|$. The runtimes of the haplotype and optimistic genotype algorithms are similar while the pessimistic genotype run times are higher because since the pessimistic cover contains many more, smaller intervals. (c) represents the distribution of the number of unique haplotypes per interval. All of these figures plot both Max-k intervals and Cores over the CEU human data set.

k-cover of the CEU human data set for each chromosome. These covers demonstrate Theorems 4 and 5 that $|C(g)| \leq |C(h)| \leq |C(p)|$. Thus, the genotype scans serve as effective upper and lower bounds on the "ground truth".

2.8 Conclusion

This basic notion of compatible intervals is described by myself and collaborators in [85]. By providing an effective means of partitioning haplotypes and genotypes into meaningful blocks on a genome-wide scale, these methods enabled several new areas for exploration. An obvious application of FGT compatible intervals is to find sets of frequently recurring local perfect phylogenetic trees [94]. These common phylogenies are of particular interest in model organisms such as laboratory mice that are thought to derive from small set of founders (i.e., fancy mice) [99], and in communities where there are ongoing efforts to generate new model populations for systems biology [12]. Our method was used effectively by Yang et al. [98] to identify meaningful blocks over which the historical subspecific origin of laboratory mice can be analyzed. Both local phylogenetic trees derived from compatible SNP intervals and the limited haplotype diversity of compatible SNP intervals can be incorporated into disease association studies, as has been recently demonstrated [82, 57, 63]. With our introduction of a method for finding compatible intervals over outbred populations, it may be possible to gain the same benefits working with less controlled populations, including humans.

The prevalence of single-SNP cores in our results also suggests new methods for cleaning data. There are several possible sources for these small local features including genotyping errors, gene conversions, and homoplasy. In addition to the obvious benefits of eliminating putative errors from a given data set, the other two sources for single-SNP cores are of great interest to biologists, but are not well-characterized. One would expect that a systematic

greedy reduction of an interval set from k to $k-1$ or $k-2$ intervals would expose larger scale structure and phylogenetic trees with improved support. I describe an effective approach to this problem in Chapter 5.

Chapter 3

Imputation using Local Phylogeny

In this chapter, I describe the method I developed to impute full-genome genotypes within populations based on the local phylogenetic structure [86]. I discuss the results of this method applied to 100 classical laboratory mouse strains. Imputation is the inference of unknown variant alleles, in a low-density data set or missing genotypes. Using genotypes consisting of 549,683 SNPs obtained with the Mouse Diversity Array (MDA), I partitioned the genome of 100 mouse strains into 40,647 intervals that exhibit no evidence of historical recombination, as described in Chapter 2. For each of these intervals, I inferred a local phylogenetic tree. These data were combined with 12 million SNPs recently discovered by whole genome sequencing in a common subset of 12 classical laboratory strains. For each phylogenetic tree, I identified strains sharing a leaf node with one or more of the sequenced strains. I then imputed high and medium confidence genotypes for each of 88 non-sequenced genomes. Among inbred strains, I imputed 92% of SNPs genome-wide, with 71% in high confidence regions. This method produced 977 million new genotypes with an estimated per-SNP error rate of 0.083% in high-confidence regions and 0.37% genome-wide. The analysis identified which of the 88 non-sequenced strains would be the most informative for improving full-genome imputation, as well as which additional strain sequences will reveal more new genetic variants. These imputed sequences, quality scores, and interactive visualizations are

publicly available and have since been used in a variety of genetic studies of this population.

3.1 Genome Imputation

Among the many advantages of inbred strains in genetic studies is that each strain needs to be genotyped only once, and that information can be reused in many experiments. Moreover, as more genotype data become available for a given inbred strain, the analysis can be updated. This cycle can continue until, ultimately, all inbred strains are fully sequenced. In the meantime, there is a need to leverage the handful of inbred strains that have been sequenced using robust imputation methods to maximize the value of existing data. High-quality imputed sequence has many potential applications including identification of functional variants and the creation of accurate scaffolds for the analysis of next generation RNA-seq and bisulfite sequencing data. Until affordable deep sequencing becomes a reality, a balanced approach that combines targeted sequencing with accurate imputation offers the best of both worlds: high quality genomic data today at little additional cost.

A recent sequencing effort by the Wellcome Trust/Sanger Institute has made available dense genome sequences for a set of 17 inbred mouse strains, including 13 common laboratory strains, three wild-derived mouse strains from different subspecies of *Mus musculus* and a single strain from a different species, *Mus spretus* [43]. This set of samples is expected to capture much of the variation found in common laboratory mouse strains and, therefore, provides a foundation for sequence imputation. A complementary resource is the recent release of MDA genotypes from 162 mouse strains [98]. The density of SNP genotypes available on the MDA is believed to exceed the density of recombination events accumulated over the development of the classical inbred strains [97] and as such the MDA SNPs can provide a framework for imputation of the underlying whole genome sequence.

Imputation can be used to increase the effective resolution of a lower density SNP panel to match that of a higher density panel when there is a subset of samples common to both

sets. Previous imputation methods use variations of a hidden Markov model (HMM) [4] to infer sequence similarities and likely transitions between haplotypes [80, 50]. These methods employ probabilistic models based on local sequence similarity to infer the state of missing genotypes. Missing genotypes arise from two sources. No-calls (Ns) indicate either technical noise or an unexpected sequence variant such as a nearby SNP or an indel that interferes with probe hybridization. A second, and more extensive, source of missing genotypes is due to differences in the density of marker sets between platforms.

There have been two recent imputation efforts in the laboratory mouse [80, 50]. Both used the NIEHS/Perlegen SNP set [28] consisting of 8.27 million SNPs over 16 common mouse strains as a source of high-density source genotypes. Szatkiewicz et al. imputed genotypes by combining low-density genotypes from 51 classical and wild-derived inbred mouse strains [99] with a filtered subset of the Perlegen SNPs set containing 7.9 million loci. The authors imputed each locus consecutively across the genome using an HMM to predict the most likely genotype among the possible alleles. Using this locus-by-locus method, they reported a 10.4% error rate over the entire genome and 4.4% error in high confidence regions. High confidence regions in this study were defined by high posterior probability and cover 71% of the genome.

Kirby et al. imputed genotypes in 94 classical and wild-derived laboratory strains for the entire Perlegen SNP set [28] using a different HMM method that predicts genotypes by estimating haplotype blocks from the smaller set of samples with high-density genotypes (EMINIM, [41]). The hidden states in Kirby's model correspond to the 16 NIEHS/Perlegen strains or a 17th unknown state. Their method models recombination between haplotype blocks rather than transitions between adjacent allele types. The authors imputed 657 million genotypes with a reported error rate of 2.4% over the entire genome and 0.27% in regions with high confidence based on posterior probability in the HMM.

These two methods do not explicitly take advantage of the local phylogenetic relationships present in classical inbred strains. This shortcoming is particularly significant given the strong population structure and the limited amount of haplotype diversity present in classical laboratory strains [98]. My approach estimates both haplotype blocks and the relatedness between them in the form of a local phylogenetic tree. Because my method is fundamentally different from these HMM methods, my notion of confidence is also fundamentally different. My confidence is determined by the structure of the local phylogenies with which I impute, whereas the HMM methods use posterior probability to define confidence level. In addition, my haplotype blocks and trees are inferred from a larger set of genotypes. This has the advantage that the larger set of samples can capture haplotype diversity that is not sampled in the smaller high-density set. The success of this approach requires that the SNP density in the larger sample set is sufficient to detect haplotype blocks, which is the case for the MDA genotypes [98]. Moreover, the trees provide a measure of difference between haplotypes that is consistent with their evolutionary history.

I use the MDA and Wellcome Trust/Sanger Institute resources to impute the genotypes of 88 common inbred strains (Figure 3.1). This approach takes advantage of the local phylogenetic relationships among inbred strains to determine the confidence of local imputation. Based on this imputation, I discuss strategies for future sequencing and SNP discovery in the lab mice and the efficient use of this resource for association studies. These data are publicly available at `http://www.csbio.unc.edu/imputation/`.

3.2 Materials and Methods

3.2.1 MDA genotype data

All genotype and haplotype data as well as the phylogenetic trees were reported in [98]. This study is based on local phylogenetic trees for 100 classical laboratory strains (Figure 3.1)

based on genotypes from MDA. MDA is an Affymetrix-based 6.5M probe platform with over 600,000 SNP markers uniformly spaced across the non-repetitive regions of the mouse genome [97]. I used a subset including 549,683 high-quality SNPs out of the 600k markers. I also identified additional alleles in these markers including residual heterozygosity, deletions and other copy-number variation, and variable-intensity oligonucleotides (VINOs) [98, 20]. VINOs and deletions were incorporated into haplotype and tree estimations by treating them as additional marker loci with binary alleles (i.e. with and without VINO, or with and without deletion) at positions coincident with the probe where they were detected [98].

3.2.2 C57BL/6J reference genome

The Wellcome Trust/Sanger Institute reports SNPs relative to the reference mouse sequence [62] and uses the NCBI Genome Reference Consortium's build 37 (MGSCv37) [11]. The reference genome is derived from C57BL/6J and, thus, I included this strain as a 12th high-density sequence along with the 11 Sanger sequenced strains for which we have MDA genotypes (Figure 3.1).

3.2.3 Wellcome Trust/Sanger Institute genotype data

Our imputation incorporates the set of high-confidence SNPs recently discovered during the Wellcome Trust/Sanger Institute's sequencing of 17 inbred strains [43] (which I will hereafter refer to as the Sanger set). Of these 17 sequenced strains, 12 are "classical" laboratory strains for which we have MDA genotypes: 129S1/SvlmJ, A/J, AKR/J, BALB/cJ, C3H/HeJ, C57BL/6N, C57BL/6J, CBA/J, DBA/2J, LP/J, NOD/ShiLtJ, and NZO/HlLtJ. I impute the remaining 88 of the 100 MDA strains, for which we do not have sequence data. The Wellcome Trust study identified over 65 million high confidence SNPs. However, 82% of these represent alleles private to wild-derived strains, so they do not vary among the subset of 12 classical strains (Figure 3.1). I impute the remaining 12,054,616 SNP loci where they are

medium or high confidence. I did not include wild-derived strains in the imputation, for reasons discussed later (see Section 3.4).

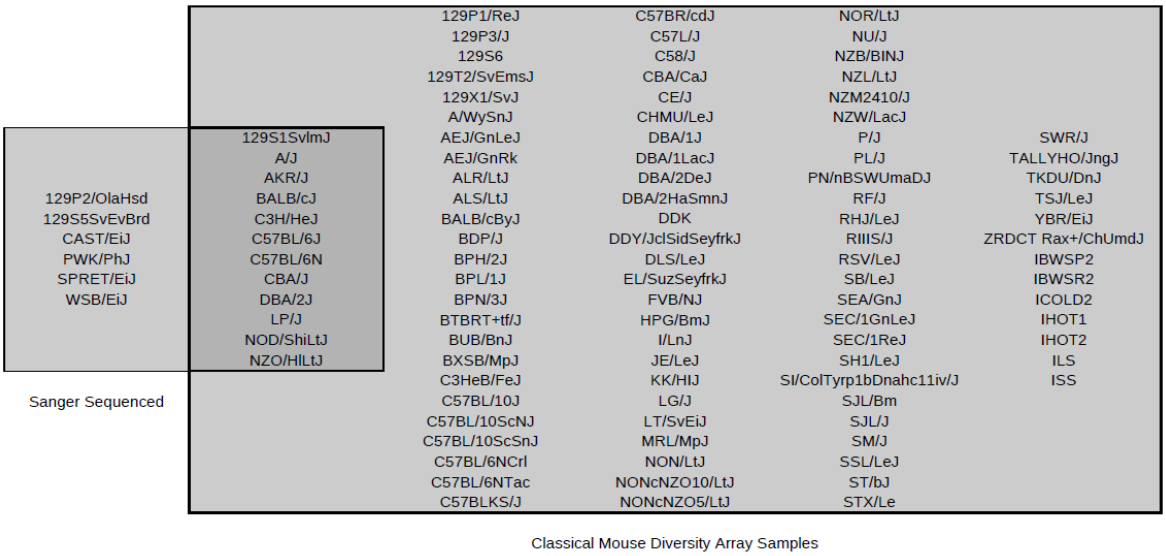


Figure 3.1: The set of mouse strains genotyped at medium-density on the MDA are shown on the right. The strains in the Sanger set are shown on the left. The overlapping set we used as the source of high-density genotypes in our imputation.

3.2.4 LG/J and SM/J validation genotypes

I verified my imputation method using an independent set of high-throughput sequence data of two mouse strains, LG/J and SM/J. Whole-genome sequencing for the LG/J (20X haploid coverage) and the SM/J (14X haploid coverage) strains [51] was completed by the Washington University School of Medicine Genome Sequencing and Analysis Center using Illumina sequencing in two steps as described in [58] and [21]. Illumina reads from DNA extracted from the livers of a single LG/J female and a single SM/J female were aligned to the July 2007 assembly NCBI build 37 reference genome using Mapping and Assembly with Quality [55]. SNPs for each strain were called using SamTools [54], requiring a minimum of three reads and a SNP quality score $\geq$20. For chromosomes 14 and 15, 305,114 SNPs were identified between LG/J and the reference and 422,879 SNPs were identified between SM/J and

the reference. The LG/J and SM/J SNPs have been submitted to dbSNP [72] under the handle Cheverud. For our validation method, I excluded ambiguous SNP calls, leaving 292,051 for LG/J and 416,589 for SM/J, relative to the reference genome.

3.3 Imputation Method

My imputation method uses the notion of compatible intervals, described in Chapter 2, to define haplotype blocks that admit a local perfect phylogeny. As described previously, I compare all pairs of SNPs and identify a minimal set of maximum size contiguous intervals covering the genome in which no pair of SNPs violates the four-gamete rule (Figure 3.3). This results in 40,647 intervals covering the entire genome. The median interval size is 71kb and covers 12 SNPs. Intervals had an average and median of 5 unique haplotypes. For each interval, I construct the local phylogenetic tree as described previously [98]. Briefly, I computed a pairwise genotype similarity score among strains as the proportion of the matching variants in each interval. I identify shared haplotypes by connecting pairs of strains with similarity score > 0.99. I constructed phylogenetic trees by connecting these haplotypes (leaves in the tree) using neighbor-joining over the mean pairwise similarity of strains in each leaf. The general workflow of my imputation method is shown in Figure 3.2.

I compared the local phylogenetic structure in our MDA genotypes to the strain allele distribution patterns (SDPs) in the Sanger set. Among the Sanger SNPs differentiation among classical laboratory strains, 96.66% are entirely consistent (four-gamete compatible) with the phylogenetic trees computed using the MDA genotypes. An additional 2.09% further subdivide leaves in the trees, but are essentially consistent. Only 0.17% of Sanger SNPs are inconsistent with the corresponding phylogeny. This validated that my local phylogenies matched the SDPs for the appropriate Sanger SNPs. This validation prior to imputation supports our assumption that MDA SNPs are sufficient to define representative haplotype

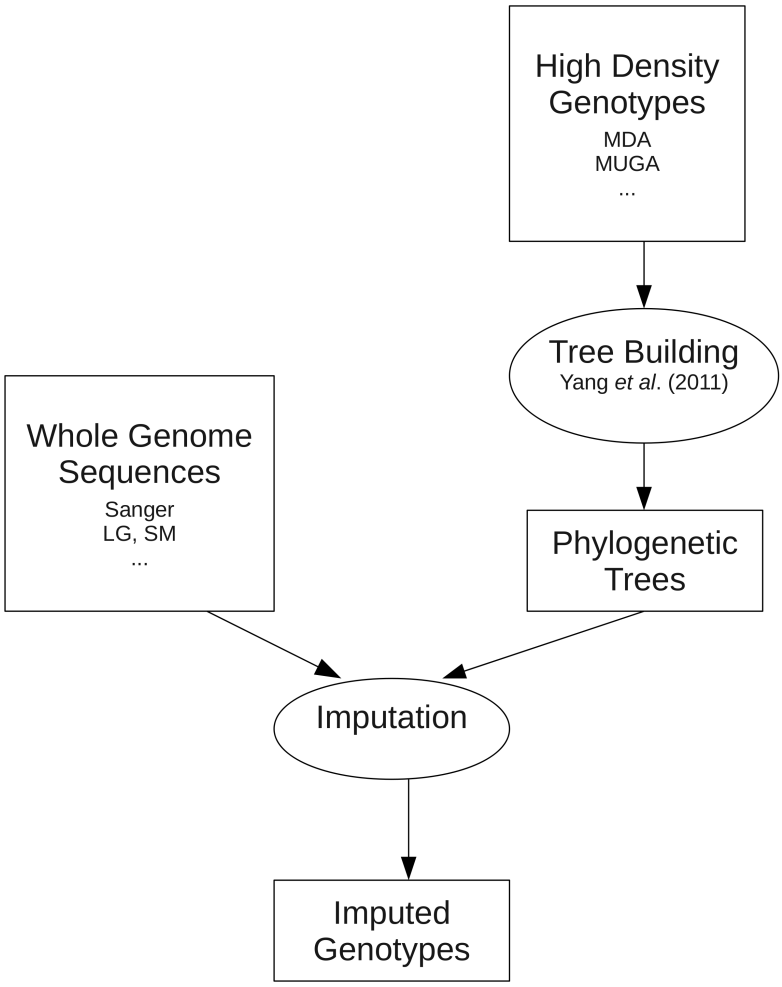


Figure 3.2: A diagram of the workflow showing the flow of data through our imputation method. Trees are constructed from the MDA genotype data (Figure 3.3). These trees are then used to inform the imputation using high-density sequence data (Figure 3.4).

blocks.

I imputed each sample for which we have only MDA data by filling in genotypes of Sanger set samples in regions where they share a haplotype that is identical-by-descent (IBD) as indicated by a shared leaf in the local phylogenetic tree. I assign confidences to each imputed strain over each interval according to whether the imputed strain shares a haplotype block with a Sanger set sample. Figure 3.4 shows an example of imputation and the correspondence between phylogenetic trees and imputation confidence. High confidence was

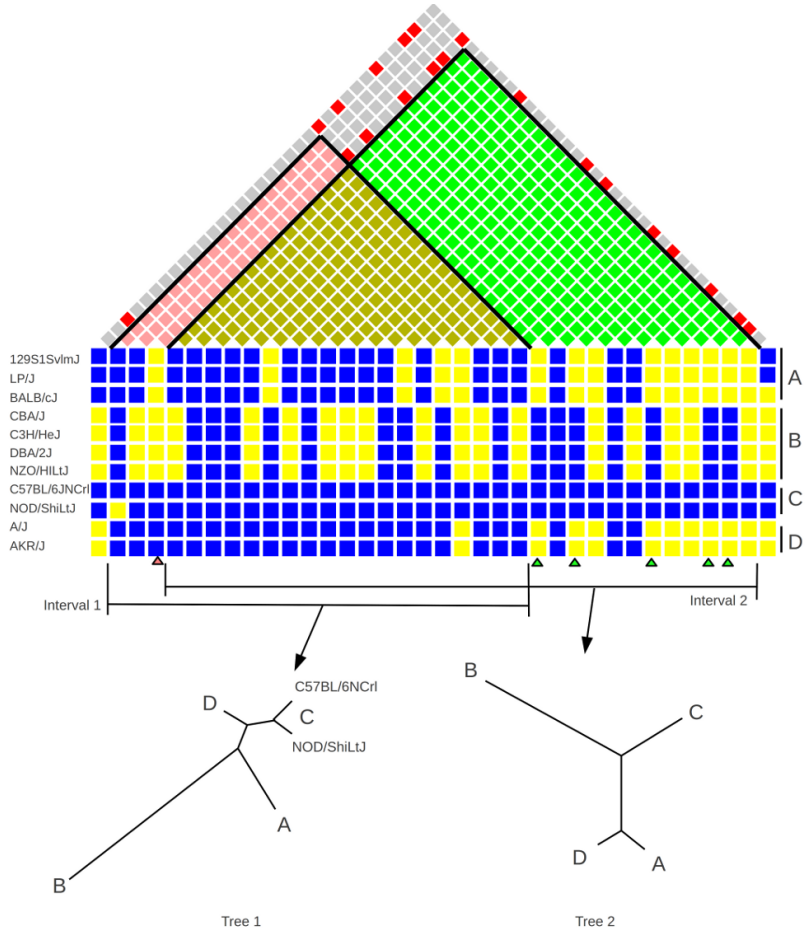


Figure 3.3: Identification of recombination intervals (see Chapter 2) and phylogenetic tree construction. Two compatible intervals and associated phylogenetic trees (73.7 - 73.8 Mb on chromosome 19) are shown. The left interval (interval 1) is shown in pink and right interval (interval 2) in green. Below the SNP matrix, the SNPs involved in violations of the 4-gamete between these two intervals are indicated by pink and green arrowheads. The four-gamete incompatibilities between these intervals mean we cannot construct a single perfect phylogeny tree. Strains are grouped by haplotype into groups A, B, C, and D - shown to the right of the haplotypes. These haplotypes are the basic unit with which I perfom my imputation. Notice both the structural changes between the adjacent trees and the subdivision of the haplotype in leaf C.

assigned to haplotypes in an interval for which there are one or more concordant Sanger sequences. Where multiple samples from the Sanger set share a haplotype, but exhibit different alleles (i.e. evidence that our tree leaf could be further subdivided), I assign medium confidence. In medium confidence cases, I resolve the allele at each locus independently by further subdividing the leaves using the haplotype structure in neighboring intervals until there is a consensus among remaining shared-haplotype samples. This method captures

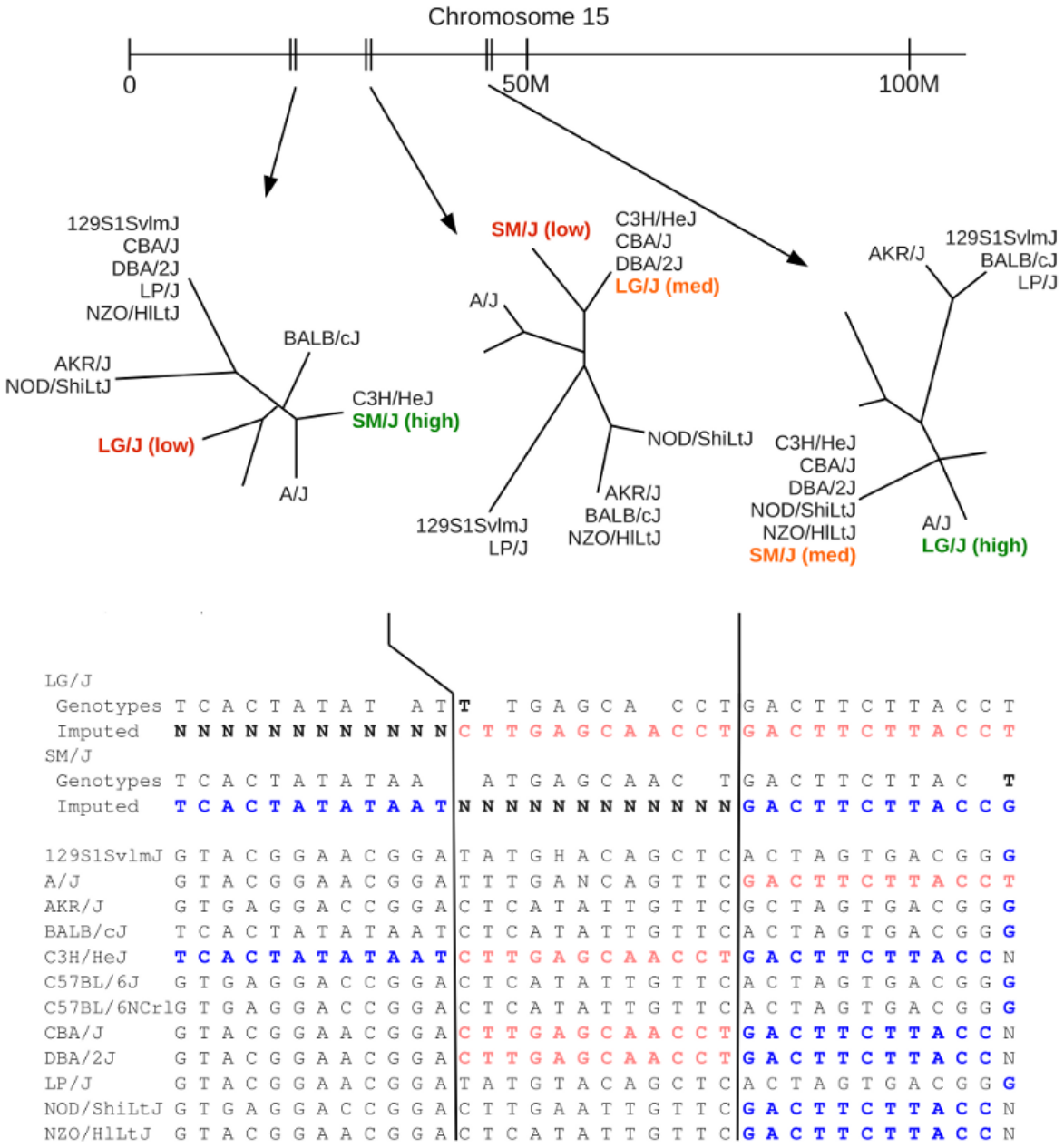


Figure 3.4: A representative example of our imputation method. Three trees on chromosome 15 (22.8 - 23.0 Mb, 31.3 - 31.4 Mb, and 45.1 - 45.5 Mb) which exhibit all three levels of confidence for LG/J and SM/J are shown. At the bottom, a sampling of SNPs in these regions is shown with the alleles contributing to the imputed sequences highlighted. Allele shown in blue contribute to SM/J, and those in pink contribute to LG/J.

nearby haplotype structure where intervals are too small to fully differentiate samples.

There are many intervals where classical inbred strains do not share a haplotype with

any strain from the Sanger set, thus no high-density imputation source is available. These intervals are assigned low confidence for strains not sharing a haplotype. Our approach provides no satisfactory imputation method in these regions and I will show in the following section that they cannot be imputed with accuracy substantially better than 50%. As a result, these regions are not assigned a call, indicated by 'N' (see Figure 3.4). Since maximal compatible intervals can overlap, two intervals might cover a SNP. In this case, the interval with higher confidence is used. If the two intervals have the same confidence, the union of strains with a shared haplotype in each intervals is used.

3.3.1 Validation

To assess the accuracy of our imputation method, I used a leave-one-out approach and compared the imputed genotypes directly to the Sanger sequence data. In the leave-one-out method, I removed the sequence for one of the 12 Sanger strains and imputed SNPs using only the remaining 11 strains in the Sanger set. This method has the advantage that it allows us to consider the entire genome when determining accuracy. In addition, I performed external validation using sequence data for chromosomes 14 and 15 obtained from strains LG/J and SM/J. I determined imputation accuracy by comparing genotypes over the intersection of our imputed SNP set (Sanger set SNPs) and the SNP sets in the external validation sequences.

3.4 Results and Discussion

I imputed 12 million SNPs over each of 88 samples for a total of 977 million new genotypes, having excluded those found to be low confidence (Table 3.1). On average, I imputed 70.76% of the genotypes with high confidence and 21.36% with medium confidence. The remaining 7.88% were low confidence. There is a wide range of variation in the fraction of SNPs

imputed with different confidence among these 88 strains. The greatest fraction of high confidence SNPs is observed in substrains derived from a common ancestor that only differ at loci harboring new mutations (97.00%) [98]. These are followed by the 129T2/SvEmsJ and TSJ/LeJ strains that have very large fractions of their genome imputed at high confidence (94.21% and 93.29%, respectively). On the other hand, KK/HlJ and TALLYHO/JngJ have the smallest fraction of genome imputed at high confidence (39.52% and 42.10%, respectively). However, it is the CE/J strain that has the greatest fraction of the genome with low confidence (23.48%).

Using a leave-one-out method, we can estimate our imputation accuracy using the Sanger set. In these samples, 1.05% of genotypes could not be imputed because the haplotype structure is unknown for some strains in regions due to deletion or copy-number variation [98]. An additional 2.25% of genotypes are uncalled (N) in the Sanger set to which I compared the leave-one-out imputed genotypes and, therefore, cannot be verified. On average, I imputed $73.8 \pm 19.1\%$ of the remaining genotypes with high confidence and $18.0 \pm 11.8\%$ with medium confidence (Table 3.2). High confidence genotypes had an average error rate of $0.083 \pm 0.019\%$ and the error rate for medium confidence genotypes was $1.57 \pm 0.75\%$. The remaining $8.2 \pm 7.5\%$ were low confidence. In these regions, methods based on the haplotype similarity and phylogeny trees performed no better than a consensus sequence among all strains in the Sanger set because there is no single representative haplotype to use for imputation. The consensus genotypes had an error rate of $44.5 \pm 10.1\%$ compared to the Sanger genotypes. Since our accuracy in low confidence regions is little better than chance, I do not impute these genotypes, and indicate them by Ns.

In addition to leave-one-out analysis, I performed validation with the chromosome 14 and 15 DNA sequence data for LG/J and SM/J, two strains that are not included in the Sanger set. I compared imputed genotypes on chromosomes 14 and 15, containing 1,316,845 imputed Sanger SNPs, with high-density genotypes for LG/J and SM/J containing 292,051 and

416,589 SNPs, respectively. For SM/J, there were 362,362 SNPs in common with the SNPs Sanger reported. Of these, 67.90% were imputed with high confidence and had an error rate of 0.07% (Table 3.3). An additional 21.25% were imputed with medium confidence at an error rate of 3.14%. The remaining were low confidence. Similar results were found for LG/J (Table 3.3). We could attempt to validate more markers if we included all loci in our imputed genotypes and assume that these markers have the reference allele in our validation sequences where a SNP is not present. This is undesirable because the SNP density is much lower in the validation sets than our imputed sequences, so there are likely many unreported SNPs in LG/J and SM/J. While LG/J has 292,051 SNPs and SM/J has 416,589 SNPs in the validation genotypes, any pair of two strains in the Sanger SNP sets has, on average, 908,493 SNPs across chromosomes 14 and 15.

My imputation method produced very low error rates in regions of high and medium confidence with both validation approaches. It provides several improvements over existing techniques and resources. It outperforms previously published imputation methods in regions in which we can impute with high confidence. In addition, I identified regions that cannot be accurately imputed with our samples because they do not share common haplotypes with any sequenced strain based on local phylogeny. Furthermore, my analysis can inform the selection of strains for which full genomic sequence would substantially improve the ability to confidently impute other strains.

Kirby et al. [50] imputed 657 million genotypes over 94 strains consisting of 65 classical and 13 wild-derived low-density sequenced strains and 12 classical and 4 wild-derived high-density sequenced strains. They imputed the 78 low-density sequences (121,433 SNPs) with the high-density NIEHS/Perlegen data set (8.27 million SNPs) [28] and missing genotypes in the 16 high-density sequences. In addition to including wild-derived strains in the sample set, 64% of SNPs in the full Perlegen data set include private alleles only seen in wild-derived strains. Szatkiewicz et al. [80] imputed 269 million genotypes using a cleaned

subset of 7.9 million NIEHS/Perlegen SNPs over 51 strains including 39 classical and 12 wild-derived. These previous imputation efforts attempt to impute a mixture of classical laboratory strains and wild-derived strains including SNPs with private alleles in wild-derived samples. Including wild-derived strains contributes many SNPs that are non-varying among classical strains and results in inflated estimates of accuracy because many of the imputed variants are actually constant among the classical population. Here I exclude wild-derived strains because they exhibit few haplotype blocks seen among classical strains and many haplotypes not seen in any classical strain. I have imputed many more SNPs that segregate among classical inbred strains. In addition, my estimated error rate in high confidence regions is 0.083% compared to 0.27% and 4.4% reported in previous studies. The error rate in high confidence regions is in line with sequencing error and the rate of recurrent mutations at highly mutable sites (homoplasy). These improvements are due, in part, to the use of a more complete set of sequence data and the fact that our lower-density set, at over 500,000 markers, is considerably denser than in previous studies. Furthermore, the MDA platform, designed specifically to highlight the diversity among our sample set, is better equipped to identify appropriate imputation genotypes than the sparser 135,000 marker Broad SNP set [50] used in previous efforts.

Due to the different SNP and strain sets used for imputation in my work compared with previous imputation methods, it is useful to analyze differences against a common set of sequence data. I identified 174,891 SNPs common to our imputed genotypes, previous imputation results, and our LG/J and SM/J validation genotypes on chromosomes 14 and 15 against which we can directly compare (Table 3.4). In SM/J, my method imputed 64.95% of SNPs with high confidence (by haplotype sharing), Kirby et al. [50] imputed 57.26% with high confidence (they define this as posterior probability > 0.98), and Szatkiewicz et al. [80] imputed 72.39% with high confidence (posterior probability > 0.9). Using our validation genotypes as the ground truth, my method achieved a per-SNP error rate of 0.03% while

previous methods achieved error rates of 1.60% and 5.76%, respectively (Table 3.4).

Because the measure of confidence is different between my method previous HMM-based approaches, one may consider whether the difference in error rate is due only to confidence assignments. My method, in some cases, imputes a lower fraction of SNPs with high confidence, perhaps contributing to the lower rate of error. However, we may include those imputed genotypes I consider medium confidence (Table 3.3), achieving an error rate of 1.29% over 90.78% of the genome, showing that my method performs better across the range of corresponding confidence values in previous methods.

My imputation method highlights an important feature of the imputed MDA sample set. This method of haplotype identification and assignment is based on the notion that classical laboratory mice are derived from a small set of recent common ancestors. To impute missing genotypes, I described previously how I identify intervals with no evidence of ancestral recombination, in which shared haplotype ancestry can be assumed (Chapter 2, [85, 98]), and identify sequenced strains that share these haplotypes with strains that are to be imputed. In some cases, there exists no evidence of a shared haplotype with a sequenced strain; these we consider low confidence (Figure 3.4). Since these intervals are not derived from a haplotype common to one of our high-density sequences, no method can produce accurate genotypes given the data on hand. This feature allows us to suggest a method for improving our imputation power by identifying those sequences that share haplotypes unrepresented in our high-density genotypes.

The strains that would provide the greatest improvement in imputation accuracy are those which share the greatest number and size of unrepresented haplotype blocks with the greatest number of other strains. These intervals are currently identified as low-confidence. If we had whole genome sequence for even one sample with the shared haplotype in these intervals (Figure 3.5), we could impute these genotypes with high-confidence. In other words, the discriminating function can be described as the greatest total number of genotypes changed

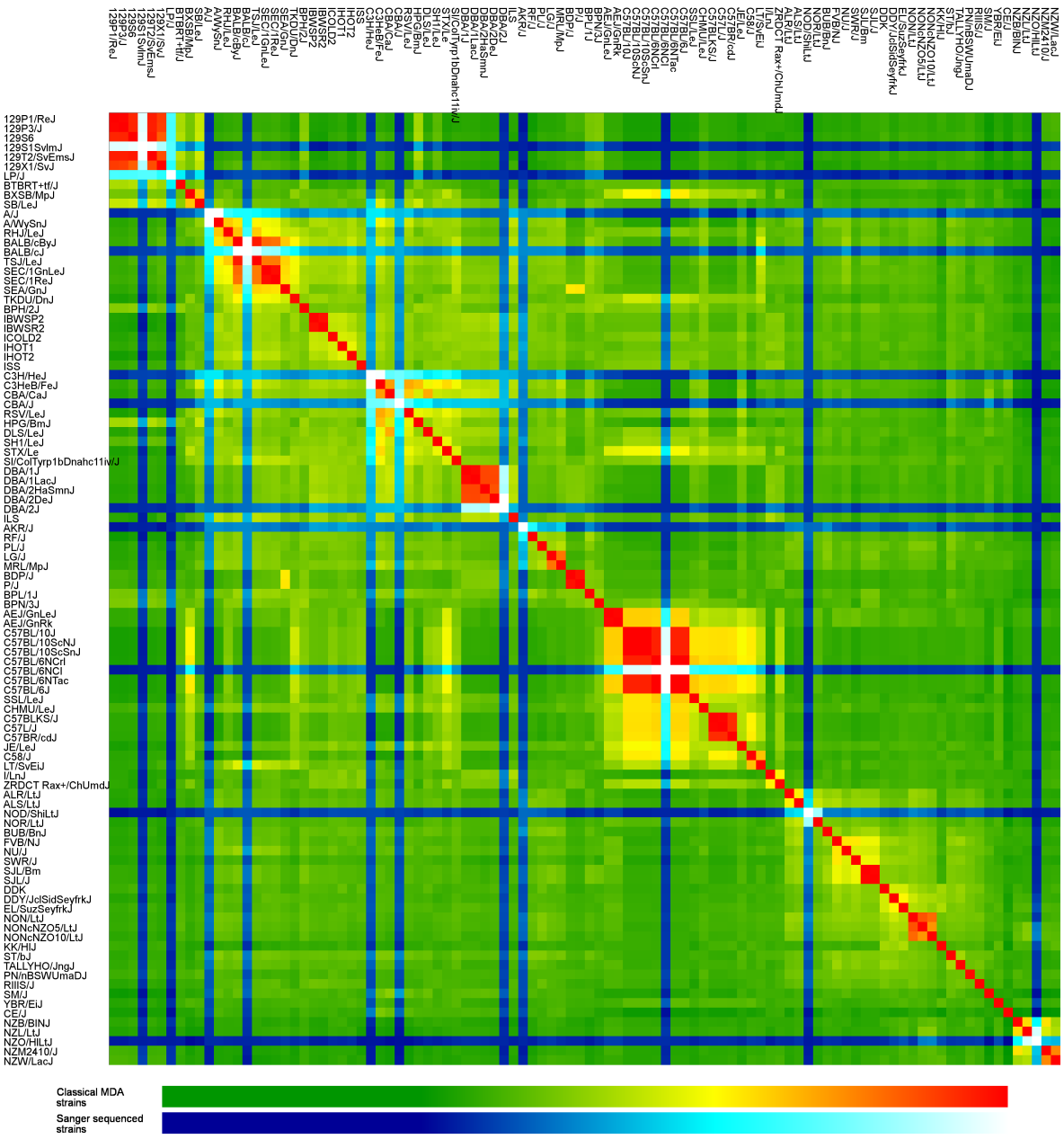


Figure 3.5: Frequency of leaf/haplotype sharing among the set of 100 MDA genotyped samples is shown as a heat map. Green to red intensity colors indicate similarity among only classical MDA strains. Blue to white colors indicate similarity with and between strains in the Sanger set.

from low-confidence to high-confidence by introducing a new fully sequenced sample into these haplotypes (Table 3.5). The strain that would contribute the greatest number of new high confidence genotypes is SWR/J, which would contribute an additional 12,973,012 high

and medium confidence imputed alleles, reducing overall low confidence regions from 7.88% to 6.65%.

Imputation, by its nature, cannot increase the number of SNPs since all imputed genotypes are derived from existing sequence. However, using our local phylogenies, we can predict samples that would contribute to the greatest discovery of additional sequence variation. While high confidence imputation power is related to the haplotype group membership (Figure 3.5) in the local phylogenetic structure, the level of sequence variation can be inferred from the edge length in the local phylogenetic trees. The edge lengths in my local phylogenetic trees are derived from the sequence differences in each compatible interval. The longer the edge, the further a leaf/haplotype is from the high-density samples and the greater sequence variation we expect to see in the unrepresented sequence. This is unlike the metric for imputation power since the haplotype frequency and sharing with other samples are not relevant. The sample that would contribute the greatest additional sequence variation is KK/HlJ. This strain has the most unrepresented sequence variation, an average sequence variation of 5.55% from the nearest Sanger sequenced sample across the genome. The top 10 candidates and their sequence variation are shown in Table 3.6.

I deliberately omitted wild-derived strains from my imputation since wild-derived strains do not share a recent history with the classical laboratory strains [5]. We do not have sufficient marker density to catalog the variants among wild-derived strains and they are likely so widely divergent that a local phylogeny cannot be constructed. However, using wild-derived strains sequenced by the Sanger Institute to impute classical inbred strains could potentially improve our current design in regions of contamination in the wild-derived strains (i.e., CAST/EiJ and PWK/PhJ [98]) or in putative regions in which a few classical strains share a haplotype that is rare or absent in fancy mice. I conclude that this could be worth doing but is unlikely to have significant impact and is not consistent with our tree-based approach. Imputing wild-derived strains would be of little value as only a single wild-derived

strain from each species or subspecies would be used to impute SNPs of all additional strains in the same taxon. The work by Yang et al. [98], in which I contributed to the phylogenetic analysis, demonstrates that there is far more sequence variation among wild-derived than classical mice, making the imputation of wild-derived samples an exercise in futility.

3.5 Conclusion

This imputation model can be further fine-tuned to better identify the appropriate intervals over which we assign a local phylogenetic structure. The optimal haplotype blocks should be small enough to accurately represent only a single indivisible haplotype but large enough to capture all appropriate variation in these haplotypes. As more samples are incorporated into the haplotype derivation model, it will be especially important to accurately represent the structure and how it relates to high-density sequenced samples. Chapter 5 describes a modification of the compatible intervals model which allows us to incorporate additional and more variable sources of data.

Imputed genotypes are evolving resources (`http://cgd.jax.org/datasets/popgen/imputed.shtml` [80]; `http://mouse.cs.ucla.edu/mousehapmap/` [50]). As the pace of sequencing increases for the mouse genome, the need for a codified and coherent resource will be even more important. Using my method of imputation, additional full genome sequences can be easily incorporated and will further improve the imputation accuracy (Figure 3.4). Since my model explicitly takes advantage of local phylogeny and haplotype structure and accounts for multiple instances of a single derived haplotype, we can incorporate multiple and varying sequences representing a single sample or strain. This will help consolidate and derive a consensus from possibly discordant sources. My phylogeny modeling can be extended to include classical strains genotyped with MDA, such as the upcoming Collaborative Cross strains [14], to provide an even larger and more diverse resource

of imputed genotypes. The method I present here is not limited only to the laboratory mouse, but could be extended to any organism for which inbred populations exist, such as rats and dogs, as well as many plant species. Imputation in any species will be most effective when inbred populations are derived from a common and relatively small set of ancestral populations.

Strain	High confidence %	Medium confidence %	Low confidence %
129P1/ReJ	92.54	4.05	3.41
129P3/J	90.13	6.35	3.52
129S6	90.97	6.90	2.13
129T2/SvEmsJ	94.21	4.29	1.51
129X1/SvJ	90.14	6.38	3.48
A/WySnJ	97.26	1.41	1.33
AEJ/GnLeJ	76.11	18.14	5.76
AEJ/GnRk	75.57	18.57	5.86
ALR/LtJ	53.46	35.68	10.86
ALS/LtJ	57.47	32.78	9.75
BALB/cByJ	97.63	1.27	1.11
BDP/J	58.81	26.78	14.41
BPH/2J	79.92	15.79	4.29
BPL/1J	70.77	23.24	6.00
BPN/3J	68.72	25.08	6.20
BTBRT+tf/J	71.04	21.71	7.26
BUB/BnJ	56.14	32.43	11.42
BXSB/MpJ	88.38	9.36	2.26
C3HeB/FeJ	92.39	6.07	1.54
C57BL/10J	85.72	12.44	1.85
C57BL/10ScNJ	85.63	12.53	1.85
C57BL/10ScSnJ	85.66	12.50	1.84
C57BL/6NCrl	98.98	1.01	0.02
C57BL/6NTac	98.98	1.01	0.02
C57BLKS/J	70.87	21.04	8.09
C57BR/cdJ	65.74	25.67	8.59
C57L/J	70.84	21.06	8.10
C58/J	67.74	23.85	8.42
CBA/CaJ	85.47	11.39	3.14
CE/J	46.04	30.48	23.48
CHMU/LeJ	79.16	17.97	2.88
DBA/1J	84.31	12.50	3.19
DBA/1LacJ	84.70	12.09	3.21
DBA/2DeJ	96.76	1.48	1.76
DBA/2HaSmnJ	76.99	18.95	4.06
DDK	46.12	37.57	16.31
DDY/JclSidSeyfrkJ	42.14	41.76	16.11
DLS/LeJ	80.43	15.43	4.13
EL/SuzSeyfrkJ	43.96	41.46	14.58
FVB/NJ	49.52	34.92	15.56

HPG/BmJ	88.64	9.12	2.24
I/LnJ	51.49	32.43	16.08
JE/LeJ	77.22	17.42	5.36
KK/HIJ	39.52	42.42	18.06
LG/J	60.78	27.65	11.57
LT/SvEiJ	78.25	15.70	6.05
MRL/MpJ	61.72	28.72	9.55
NON/LtJ	50.87	36.24	12.89
NONcNZO10/LtJ	46.36	40.08	13.56
NONcNZO5/LtJ	56.95	32.28	10.77
NOR/LtJ	84.40	12.19	3.42
NU/J	54.68	32.14	13.18
NZB/BINJ	58.76	29.49	11.75
NZL/LtJ	81.44	13.62	4.93
NZM2410/J	49.12	36.00	14.88
NZW/LacJ	49.09	35.58	15.33
P/J	58.39	27.11	14.50
PL/J	60.00	30.83	9.18
PN/nBSWUmaDJ	46.78	37.32	15.90
RF/J	66.97	25.02	8.01
RHJ/LeJ	82.57	14.57	2.86
RIIIS/J	45.13	37.46	17.41
RSV/LeJ	87.86	10.43	1.71
SB/LeJ	84.83	12.08	3.09
SEA/GnJ	72.31	20.11	7.58
SEC/1GnLeJ	82.88	11.96	5.17
SEC/1ReJ	83.13	11.72	5.15
SH1/LeJ	83.49	13.85	2.66
SI/ColTyrp1bDnahc11iv/J	75.42	19.96	4.62
SJL/Bm	53.84	32.52	13.65
SJL/J	53.89	32.49	13.62
SM/J	51.79	29.97	18.24
SSL/LeJ	82.21	13.46	4.34
ST/bJ	53.69	33.28	13.04
STX/Le	89.96	8.69	1.35
SWR/J	47.74	33.68	18.58
TALLYHO/JngJ	42.10	42.16	15.75
TKDU/DnJ	85.40	10.63	3.97
TSJ/LeJ	93.29	5.29	1.42
YBR/EiJ	47.36	37.29	15.35
ZRDCT Rax+/ChUmdJ	65.05	23.42	11.54
IBWSP2	67.83	25.23	6.94
IBWSR2	65.80	26.70	7.50
ICOLD2	70.84	22.58	6.59
IHOT1	70.31	24.01	5.68
IHOT2	69.29	24.49	6.22
ILS	76.18	16.99	6.83
ISS	74.25	20.01	5.74
Average	70.76	21.36	7.88
Standard Deviation	16.58	11.46	5.49

Table 3.1: The percentage of SNPs imputed with high, medium, and low confidence in 88 classical inbred strains. I impute 71% of genotypes with high confidence, on average. However, strains range from 40% to 99% depending on the frequency of shared haplotypes.

Strain	HC %	HC error %	MC %	MC error %	LC %	LC error %
129S1SvlmJ	76.83	0.07	15.35	1.74	7.82	43.75
A/J	81.26	0.09	13.47	2.30	5.27	44.98
AKR/J	58.50	0.11	29.98	1.85	11.52	44.18
BALB/cJ	82.07	0.08	13.39	1.86	4.55	37.81
C3H/HeJ	85.29	0.09	11.92	0.82	2.79	53.76
C57BL/6J	97.05	0.08	2.89	0.54	0.06	33.87
C57BL/6NCrl	97.21	0.04	2.77	0.13	0.02	19.94
CBA/J	81.91	0.08	13.56	1.52	4.54	52.73
DBA/2J	66.46	0.10	22.54	1.60	11.00	50.67
LP/J	79.05	0.07	14.32	1.76	6.63	55.25
NOD/ShiLtJ	42.00	0.10	38.60	1.84	19.40	45.85
NZO/HlLtJ	38.04	0.10	37.12	2.83	24.84	51.20
Average	73.81	0.08	17.99	1.57	8.20	44.50
Standard Dev.	18.32	0.018	11.30	0.71	7.20	9.64

Table 3.2: The percentage of genotypes imputed with high, medium, and low confidence in my leave-one-out analysis, and the error rates for each confidence level. On average, 74% of genotypes are imputed with high confidence with an error rate of 0.08%

Strain	HC %	HC error %	MC %	MC error %
LG/J	65.58	0.08	25.20	4.43
SM/J	67.90	0.07	21.25	3.14
Average	66.74	0.075	23.23	3.79
Standard Dev.	1.16	0.005	1.98	0.65

Table 3.3: The percentage of genotypes imputed at high and medium confidence, and error rates, in LG/J and SM/J. I imputed LG/J and SM/J using the high-throughput Sanger sequence SNPs and validated using SNPs derived from an external sequencing effort [51]. This comparison provides additional validation of my imputation method, achieving only 0.075% error in high confidence regions.

	This study	Kirby et al. 2010	Szatkiewicz et al. 2007
LG/J HC %	60.91	65.88	73.07
LG/J HC error %	0.06	3.75	6.57
SM/J HC %	64.95	57.26	72.39
SM/J HC error %	0.03	1.60	5.76

Table 3.4: Performance of imputation methods based on percentage of SNPs imputed at high confidence and fraction of error in chromosomes 14 and 15. I compare my imputation method to previous imputation approaches [80, 50]. In this normalized comparison, my method achieves an error rate orders of magnitude lower than previous studies.

Strain	HC genotypes	% increase
SWR/J	12,873,012	1.21%
SJL/Bm	12,806,101	1.21%
SJL/J	12,699,148	1.20%
BDP/J	12,592,551	1.19%
P/J	12,497,039	1.18%
DDY/JclSidSeyfrkJ	12,002,104	1.13%
FVB/NJ	11,985,984	1.13%
DDK	11,892,009	1.12%
KK/HIJ	11,546,347	1.09%
I/LnJ	11,497,933	1.08%

Table 3.5: These strains contribute the most high confidence genotypes not represented in the Sanger set. Including high-density genotypes from these strains with the most unrepresented haplotypes would significantly increase the power of the imputation model.

Strain	% unrepresented variation
KK/HIJ	5.55
NZM2410/J	3.47
EL/SuzSeyfrkJ	3.30
DDK	3.09
DDY/JclSidSeyfrkJ	3.08
BDP/J	2.79
P/J	2.77
SWR/J	2.57
SJL/Bm	2.22
SJL/J	2.22

Table 3.6: These strains have the highest sequence variation not represented in the Sanger set. The percentage of unrepresented variation indicates the average "distance" in the local phylogenetic trees of these strains from the nearest Sanger haplotype. Sequencing of these strains is likely to reveal the greatest number of new variants.

Chapter 4

Visualization of Local Phylogeny

In this chapter, I describe a novel tool for comparative genomics based on my model of local phylogenetic structure. I have developed a web-based tool for visualizing and analyzing multiple collinear genomes. My tool illustrates genome-sequence similarity using a mosaic of compatible intervals and the higher structure derived from them, including subspecific origin and haplotype identity. Comparative analysis is facilitated through reordering and clustering of genomic data which can vary between individuals. A significant feature of my genome browser is that I provide local phylogenetic trees as an alternate visualization to assess the relatedness of local haplotypes.

Genome browsers are a common tool used by biologists to visualize genomic features including genes, polymorphisms, and many others. However, existing genome browsers and visualization tools are not well-suited to perform meaningful comparative analysis among a large number of genomes. With the increasing quantity and availability of genomic data, there is an increased burden to provide useful visualization and analysis tools for comparison of multiple collinear genomes such as the large panels of model organisms like those commonly used in modern genetic research.

Unlike previous genome browsers and viewers, this tool allows for simultaneous and comparative analysis. The browser provides intuitive selection and interactive navigation

around features of interest. Dynamic visualizations adjust to scale and data content making analysis at variable resolutions and of multiple data sets more informative. I describe the design and implementation of this genome browser based on my compatible intervals and demonstrate these features using an extensive set of genomic data sets composed of almost 200 distinct mouse laboratory strains.

4.1 Classical Genome Browsers

Genome browsers are one of the most common bioinformatics tools used by biologists. Browsers allow biologists to visualize genomic features such as genes, SNPs, CG islands, and transcription factor binding sites, and to place these features in their genomic context. They are also useful in adding and viewing genome annotations and feature-specific information. Generally, genome browsers support analysis of a single genome, but there is often a need to compare features between two or more genomes. Existing tools are not well-suited for this. Many visualization methods have been developed to support comparative genomics of animals from different species. These include phylogenetic trees, alignment viewers, Circos diagrams [52], and dot-matrix methods [7]. Tools that perform comparative analysis include BLAST (pairwise alignment analysis) [1] and VISTA [29]. Generally these methods support only comparisons between a small number of genomes. There is a need for comparative analysis and visualization tools supporting populations from the same species with largely collinear genomes. These genomes are considered "collinear" in that they can largely be considered identical in structure and content except for point variations. This is a common assumption within species. My goal was to develop a system that supports simultaneous and dynamic analysis of many (10s to 100s) collinear genomes and to display the context and extent of compatible intervals and their corresponding phylogenetic trees.

A web-based resource for investigating genomic data from multiple samples simultaneously would aid many common comparative genome analyses including disease association

studies and expression analysis. My system supports any generic genomic data set, allowing it to be an extensible framework for analysis, not simply a data resource. Like existing genome browsers and viewers, I represent different categories of genomic data as horizontal "tracks" covering a particular region of the genome. Unlike previous work, color is used to better indicate important regions and facilitate comparison. In addition, my tools allows dynamic sorting and local reordering of tracks.

Comparisosn between genomes of different samples of the same species, particularly the analysis of local haplotype and phylogeny, can provide insight into gene origins and individual variations. They can also aid in understanding population structure. Understanding local genomic variations and population structure is the key to studies of individual genes and their association with disease. We need to be able to not only determine similarities and differences between samples genome-wide, but also at the level of individual loci (Figure 4.6).

There are many genome browsers and viewers that can integrate multiple data sets pertaining to a particular genome sequence whether it is specific or a species consensus. These browsers, including the UCSC genome browser [45], GBrowse [78], Ensembl [39], NCBI Map Viewer [24], and JBrowse [74], display multiple tracks of data and support a variety of useful navigation techniques that allow the genome to be traversed and visualized at various resolutions. However, existing browsers are limited in their ability to support dynamic and comparative analysis between multiple genomes.

The UCSC Genome Browser [45] is the standard and most prevalent web-based genome browser. The UCSC browser originally targeted the human genome data as a part of the Human Genome Project. It has since been extended to numerous other species. The goal of the UCSC browser is to make a particular set of data broadly accessible and navigable. It does not focus on any particular analysis but is a comprehensive resource for integrating, displaying, and navigating publicly accessible genome data. The browser supports standard

functions including navigation by panning and zooming. Data sets of interest can be displayed in tracks and reordered manually by the user. The UCSC browser functions as a window into very comprehensive sets of data for many different species, but does not support comparisons between either inter- or intraspecific genomes. The UCSC browser does not support dynamic interactions with the displayed data. Instead, pages must be reloaded in their entirety any time that new data is requested. Due to this limitation, data retrieval is necessarily limited to a small window or few data types to allow quick and easy analysis.

The Generic Genome Browser (GBrowse) [78] is another widely used web-based genome browser available for human, mouse and other model organisms. The main difference between GBrowse and the UCSC browser is extensibility. GBrowse is designed to be extended with new and user-provided data sets, and as such it provides a flexible framework for displaying and navigating arbitrary genome information. Otherwise, GBrowse uses the same basic navigation and display structure as the UCSC browser. Data sets can be individually selected and are displayed as horizontal tracks stacked on top of one another and aligned to a common genomic scale. Unlike the UCSC browser, GBrowse supports asynchronous retrieval and navigation of data, meaning the entire page does not need to be reloaded to update the genomic regions displayed. This reduces the computational overhead on the server, refreshing only those parts that need to be changed. However, GBrowse is limited in its ability to display small-scale details at high resolutions. Since the representation and visualization of data is essentially fixed, fine details such as SNPs are often omitted when viewing large regions.

The Ensembl genome database project [39], a joint venture between the Sanger Institute and the European Bioinformatics Institute (EBI), was initiated with a goal of providing full genome data along with various annotation as a public resource for researchers. The Ensembl genome browser serves as a publicly available web-based browser for this data. Although initially focusing on the human genome, the browser now includes many model-organism

genomes with annotations including genes, DNA and RNA alignments, and many others. The browser function itself is similar to the UCSC browser, supporting traditional navigation techniques. Ensembl also uses asynchronous data requests to retrieve data when it is needed. In addition, detailed annotations and links to more thorough information are displayed when a feature such as a gene or contig is selected.

The National Center for Bioinformatics Information (NCBI) provides the NCBI Map Viewer [24] as an online tool for browsing genomes. Unlike others, the NCBI Map Viewer displays the genome vertically with tracks for only the assembly, contigs, and genes while focusing on detailed description and annotation for these features linking to other useful NCBI tools for directly accessing related genes, SNPs, proteins, and more. Map Viewer also does not provide any dynamic navigation mechanism, therefore the entire page must be reloaded each time the genome window is adjusted. The browser serves best as a hub through which other resources are accessed by genomic position and is not a viable analysis tool by itself.

JBrowse (Javascript-based genome browser) [74] is a more recent web-based tool designed to allow navigation and analysis of genomes and is available as a framework that researchers can set up and fill with their own data. JBrowse takes advantage of the dynamic features available to modern browsers and allows for more interactive and dynamic visualizations. JBrowse's focus is on supporting dynamic and fluid transitions between displayed windows, for example showing a smooth sliding transition as a user pans in one direction along the genome. JBrowse also supports client-side dynamic rendering rather than the server-side image or block rendering as in most other browsers. This reduces the server-side computation and time and cost associated with data transfer of full images between the server and client. JBrowse leverages the computational power on the user's browser to draw and dynamically shift and rescale the visualization, leading to a more intuitive understanding of the relationship between genomic features as a user shifts the frame of reference. JBrowse is

a good tool for generic genome annotation analysis, but, as with other existing browsers, it does not provide suitable techniques for visualizing multiple genomes simultaneously.

Existing genome browsers are well suited for generic genome annotation and are useful for analysis of the specific data sets to which they are tailored, but there are many limitations. Available data is essentially static. In many cases, users have the ability to customize the browser to use different data or display only that in which they are interested, but the underlying information representation remains constant. The visualization is essentially static, where the current region of interest is displayed to the viewer. Data can be viewed at multiple resolutions, but no further attempt is made to improve upon the organization and layout of the visualization for a particular purpose. It is hard to quickly glean information and understanding from the visualization. These tools are frequently used to provide access to publicly available data sources rather than to support novel visualizations for analysis. Our browser addresses the following limitations of existing genome browsers: it supports simultaneous exploration of multiple aligned genomes, it allows for dynamic rearrangements of tracks to support comparisons, and it provides alternative visualization modes based on the current displayed scale.

4.2 Browser Design

The Mouse Phylogeny Viewer (MPV) is available as a public website allowing users to view, explore, and analyze multiple genomic data without requiring a standalone application (http://msub.csbio.unc.edu). Data is stored on the web server and the client side executes within a web browser. It has been tested and works on most modern web browsers and operating systems. Browsers using Webkit, Mozilla, and other renderers/Javascript engines conforming to the HTML5 W3C specification are known to work. Platform interoperability and constant availability make it an easy and useful tool for genetic analysis.

Figure 4.1: An overview of the visualization interface. Basic navigation controls are shown at the top, allowing users to zoom and pan within the viewing region. Three track types are shown, a histogram representing the density of SNPs, subspecific origin (see Figure 4.3), the underlying compatible intervals (Fig. 4.11), and a single phylogenetic tree representing the local phylogeny in the interval covering 7.8 - 8.9 Mb on chromosome 19.

The design of MPV focused on the visualization of multiple simultaneous components from aligned data sets (Figure 4.1). To this end, the browser supports a multi-row display, where individual data sets are displayed as vertically stacked tracks that be compared vertically, along with derived data tracks which integrate information over the selected subset of samples. MPV assumes collinearity, common local coordinates, of all feature tracks of interest. The displayed samples are selected or deselected by clicking the strain name in the selector region. Data sets can also be individually shown or hidden depending on the analysis performed by toggling the show/hide button next to each track group.

4.2.1 Navigation

MPV supports various navigation techniques including manually selecting a region of the genome, panning backward and forward through the genome, and zooming in and out (Figure 4.2). Clicking and dragging over any track highlights the region over which the pointer is dragged on all displayed tracks (Figure 4.4). This allows users to highlight regions of interest to easily compare between track groups. In addition, once a region is selected, a button appears to allow zooming in to the selected region such that it fills the entire viewing window (Figure 4.4). This allows for precise navigation to features of interest. There are also navigation buttons to zoom out by fixed small (two) and large (ten) ratios of the displayed window size. Panning side to side is supported by four buttons; two pan in each direction, one a short distance and one a long distance, 10% and 50% of the viewing window size, respectively. Panning small distances allows the user to fine tune the display to focus on a region of interest. Further panning allows users to scan the genome for nearby features while maintaining a local frame of reference. In addition to panning and zooming, when a point on any track is clicked, a vertical cursor line is highlighted to allow visual alignment of features at that point. The display window may then be re-centered around the selected position to best show the chosen feature and its surrounding area (Figure 4.5).

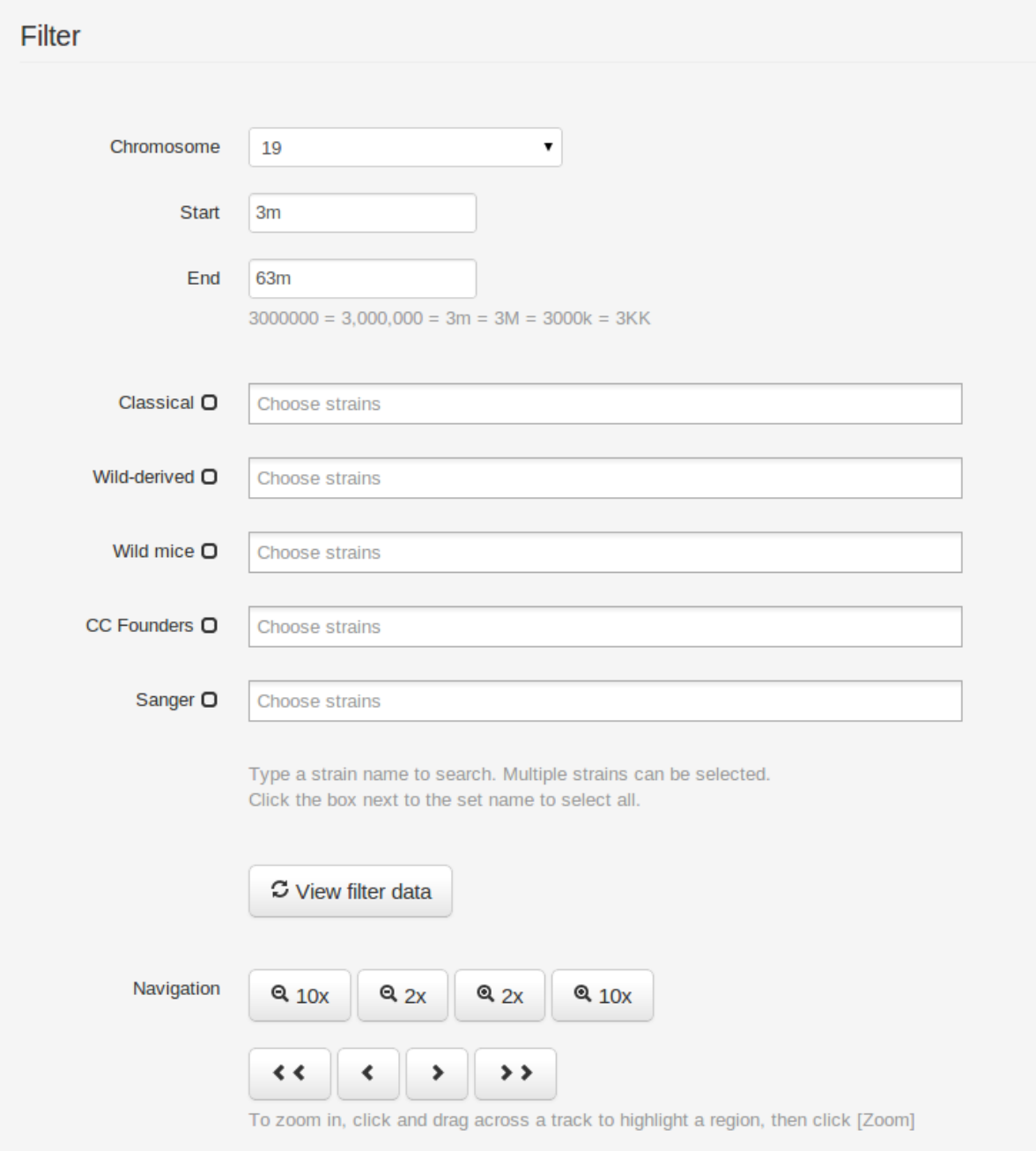


Figure 4.2: Strain selection and navigation. Multiple strains can be selected from several groups to visualize and compare. The displayed window reflects the chosen chromosome and base-pair coordinates. Navigation actions include zooming in and out, and panning to the left and right varying amounts.

The user interface provides access to the underlying genetic data. For most data types, the displayed information can be retrieved as a delimited text file by clicking the output button below each track, which retrieves the underlying data for the currently selected sets of active

Figure 4.3: Data tracks for subspecific origin and haplotype similarity. Some data sets, such as the haplotype coloring, show only data for the selected classical laboratory strains. The user can drag tracks within a group to reorder the display of samples. This reordering is reflected in all grouped tracks. Users can navigate the genome by manually entering positions (Figure 5.1) or use the navigation buttons to zoom in and out and pan side to side across the genome. Data tracks can also be collapsed and expanded using the ^button above each track. In the subspecific origin group, the colored tracks indicate the subspecies origin of each strain. Throughout, blue indicates *Mus musculus domesticus*, red indicates *M. m. musculus*, and green indicates *M. m. castaneus*. Shared colors in the haplotype tracks indicate a common haplotype.

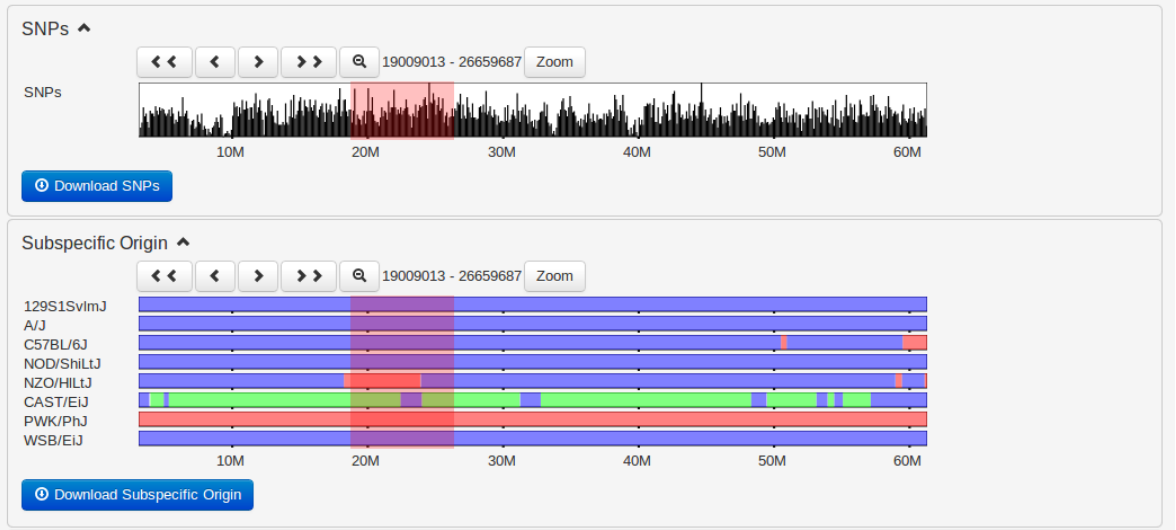


Figure 4.4: A region of mouse chromosome 19 selected by clicking and dragging across the genome. The region is highlighted on all tracks and the user is given the option to zoom in to the chosen region.

genomes and within the displayed window so that no further filtering is required.

The visualization technique that is used changes based on both content and scale. Unlike

Figure 4.5: The results of zooming into a highlighted region. These two tracks show a single point in the genome selected to allow the user to center the viewing window on the chosen position. The position selected by clicking one track is reflected in all other tracks to pinpoint aligned features.

previous browsers, the visualization changes to aid analysis based on the subset of samples chosen, the order of samples, and the viewing resolution of the data. The basic data representation used by the browser is a set of possibly overlapping genomic intervals specified by their genomic coordinates (typically chromosome and position). Intervals are displayed as horizontal blocks that are displayed along the viewing window based on the bounding positions of the interval. Overlapping intervals are displayed on adjacent stacked tracks. Dynamic visualization techniques are also applied to the subset and order of comparative samples selected. Samples can be displayed simultaneously and their tracks reordered such that they can be easily compared. In addition, a variety of visualizations are computed dynamically based on the current subset of selected samples, such as intervals of sequence identity among the selected set of samples.

Another feature facilitates similarity analysis at a particular position by allowing sorting of tracks within all groups at a user-selected position within the displayed genomic window. Strains are sorted vertically according to the color at the selected position such that strains

with identical features are grouped together. In addition, strains are further sorted at increasingly distant positions radiating in both directions from the selected position until either the edge of the displayed window is reached or all strains are distinct. This is particularly useful in assessing local similarity among a large set of samples.

4.2.2 Multiscale Visualization

When examining a small region of fine detail, features such as SNPs and genotype intervals are displayed as discrete blocks or points such that individual features and their exact relationship can be determined (Figure 4.6). Because data may be visualized at a wide range of resolutions relative to the genome size, the number of intervals to display may exceed the display resolution. To combat this, regions with multiple intervals are presented, at a course resolution, as histograms. This resolution-dependent representation (RDR) supports a wide variety of genome annotations and allows the browser to be easily extended to novel data sets while providing a more useful high level interpretation of the data (SNPs in Figure 4.3). The use of this visualization mode is well supported by my model of local phylogeny.

4.2.3 Use of Color

The genome browser uses color to allow for more intuitive visualization. Intervals for various data types are displayed as variable-width colored bars across the genome, highlighting similarities and differences between genomes by their respective color pattern. To allow users to further customize the visualization, the browser supports dynamic recoloring of intervals (Figure 4.7) as well as dynamic sorting of samples at a user-selected position (Figure 4.8). Dynamic coloring and reordering tools facilitate comparison of features by visually aligning regions where genomes are similar and different. At any position along the genome, the relationships among the displayed strains can be understood visually as dividing strains into groups according to their color such that strains with similar genomic features are the same

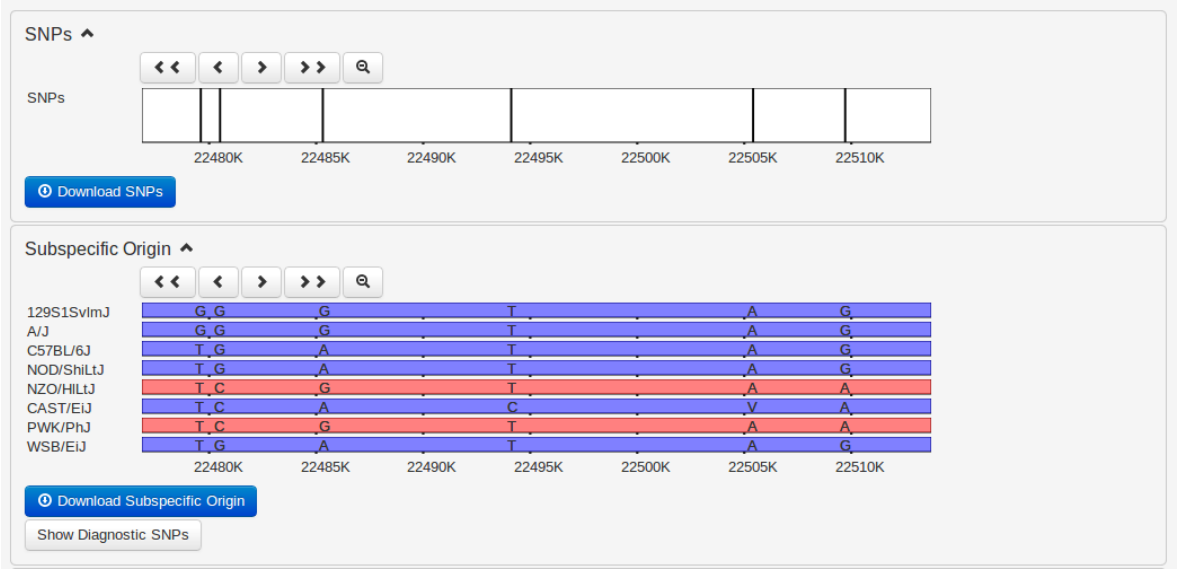


Figure 4.6: Mouse Diversity Array SNPs (above) and subspecific origin (below) shown at a fine resolution. A high density of SNPs is represented as a histogram across the genome. As the user zooms in, the histogram's bar heights dynamically adapt to display the relative SNP densities in each genomic region. Individual SNPs are displayed as vertical ticks along the SNP track as the display resolution approaches an individual base-pair. Alleles for each strain are shown overlaying the subspecific origin to enable detailed analysis. Alleles are also shown overlaid on the haplotype track at low resolutions.

color. Over larger regions, similarity is represented by shared color patterns.

When viewing only a small sample of strains, this coloring can be simplified, essentially changing colors only when there are feature changes among the selected strains. Colors can be dynamically reassigned according to the order of the selected strains such that colors are assigned in descending order (Figure 4.8). The topmost displayed strain is assigned a single color across the genome. The second strain is assigned the color of the previous strain where its content/color matches the first and a second color where it differs. This process is repeated for subsequent strains, introducing a new color in regions that match none of the preceeding strains. This has the effect of, for example, highlighting all regions where the first selected sample shares some feature with subsequent samples by using the same color. In this way, the coloring scheme can be substantially simplified for a small sample of strains allowing more intuitive analysis.

Figure 4.7: In the haplotype track, strains are assigned colors based on shared haplotypes - equivalent to a shared leaf in the local phylogenetic trees. The default haplotype block mosaic, which minimizes total color transitions across the entire genome, is shown above. Below is the selected subset of strains recolored according to their displayed order. The topmost strain is assigned a single color and subsequent strains are assigned the same color where their haplotypes match the first strain. A strain is assigned a second color where it does not match the first strain and subsequent strains are assigned the color where they match the new strain but not the first. This process is repeated for all remaining strains in displayed order. This recoloring highlights the haplotype similarities over extended genomic regions (60 Mbases as shown) between the selected strains.

4.3 Browser Implementation

There are many critical design and resource allocation decisions which arise when handling very large sets of data. In traditional genome browsers, a relatively small amount of data needs to be handled at any one time. Existing browsers only need to handle a single sequence. In order to visualize multiple sequences simultaneously (10s to 100s) as in the case of our implementation, it is important to consider different methods for efficient data transfer, management, and visualization. In addition to handling multiple sequences, my tool also supports dynamic visualizations that vary based on the scale and local context. Existing browsers, such as the UCSC Genome Browser [45], do not support large-scale visualization

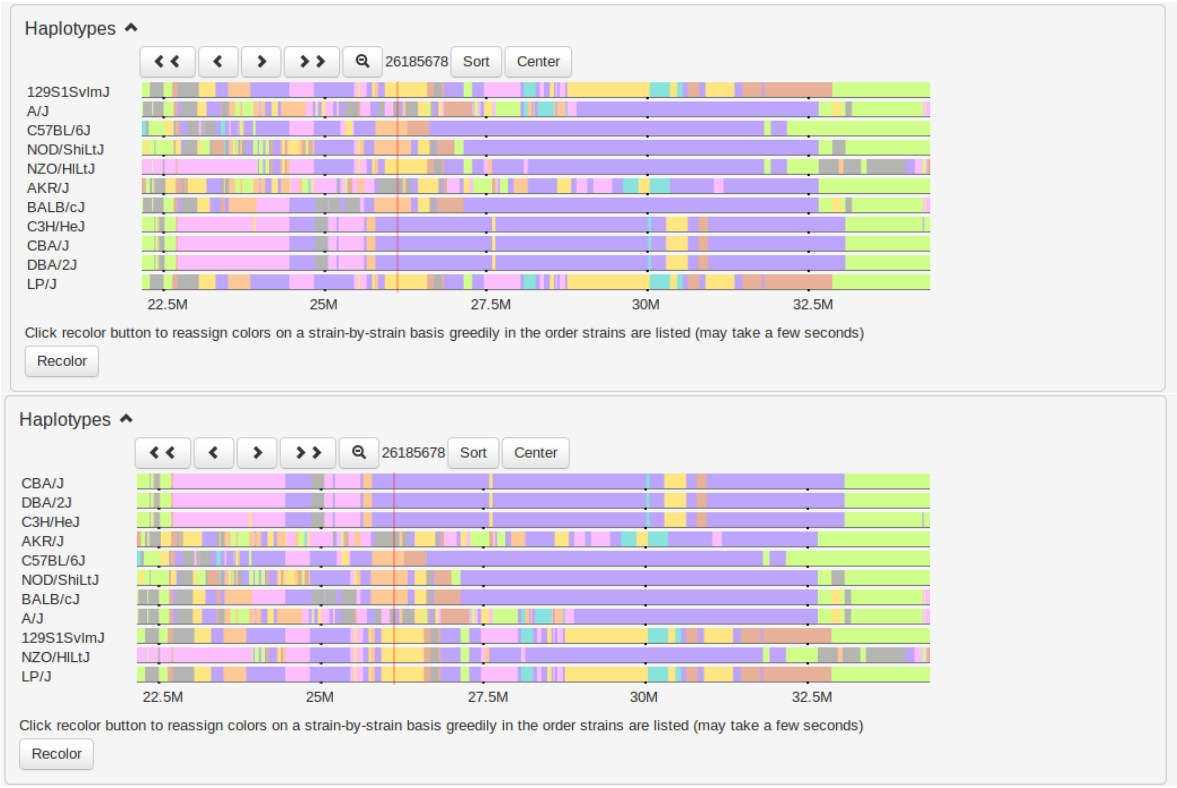


Figure 4.8: The top track group shows haplotype similarities with strains shown in the default order. The lower track shows this track after automatic sorting according to the haplotype similarity at the selected cursor position. Strains are sorted by haplotype at the selected position. Strains with the same haplotype are further sorted by adjacent positions until all strains are distinct. Other tracks are reordered to match.

of fine-scale features, like SNPs.

To support faster and more interactive visualization while dealing with remote data, I addressed issues of data transfer and efficiency and how to best allocate the rendering tasks. My implementation loads data as it is needed into the page using asynchronous requests (using AJAX [84]) to the server. To reduce data transfer costs in memory and speed, the page is loaded only once at the beginning of a session and, subsequently, only data is loaded. In addition, visualization and display are handled in the browser by dynamic scripts on the page so that complete images do not have to be transferred from the server. Data rescaling, panning, and drawing are all handled by the client. Requests are made asynchronously so that the tool is available to the user even while new data is transferred.

In order to increase usability as well as allow dynamic content and interaction, my implementation requires a set of web-based technologies that would allow wide platform interoperability and dynamic client-server interaction. The user interface and interactive components are constructed in Javascript using the JQuery (`http://jquery.com`) library to simplify implementation and abstract the numerous issues arising due to variable cross-browser syntax.

4.4 Browser Usage

I have deployed an instance of the visualization tool to aid analysis and interpretation of a Nature Genetics paper [98] that I coauthored. This browser analyzes a set of 100 classical laboratory and 62 wild-derived mouse strains along with 36 wild-caught mice. This study answers open questions regarding the subspecific origin of the laboratory mouse and provides the first detailed view of the haplotype diversity in most common laboratory mouse strains. This instantiation is used to visualize ten different data types to aid in comparative analysis of these 198 mouse samples.

Several data sets are included to aid in analysis by placing features in a genomic context. The browser supports display of SNPs from the Mouse Diversity Array [97] that used in genotyping the analysis presented in the paper. SNPs use the RDR approach in a dedicated track. In addition, alleles at each SNP for each strain are displayed at fine-scale resolutions overlaying the subspecific origin and haplotype coloring tracks to allow for direct comparison (Figure 4.6). Known genes [38] are displayed in a similar manner. In the case where genes overlap, overlapping genes are displayed in additional stacked horizontal tracks.

A second data set of interest is the local subspecific origin of each sample's genome. The genomes of classical laboratory mouse strains arose through interbreeding of pet mice from three different mouse subspecies. In [98], the subspecific mosaic structure of each

genome was determined (Figure 4.3). Subspecies are assigned to each strain as a mosaic of intervals representing *Mus musculus domesticus*, *Mus musculus musculus*, or *Mus musculus castaneus* regions of origin. SNPs were assigned their diagnostic status based on each allele's distribution among wild and wild-derived mouse strains of known subspecies. In addition to diagnostic alleles, diagnostic values were also assigned to SNPs based on the distribution VINOs [20]. Subsequently, genomic regions of mouse strains from unknown subspecies were assigned a subspecies and assigned a confidence based on these diagnostic values. A Hidden Markov Model (HMM) was used to delineate subspecies intervals across the genome which integrated the diagnostic strength of the SNP markers, a data-error model, and minimized the number of transitions (see [98]). Subspecific origin is visualized as a horizontal track made up of a mosaic of colored bars representing domesticus (blue), musculus (red), or castaneus (green) regions for each selected strain. At fine scales, diagnostic SNPs are shown above the subspecies assignment for each strain, the height and color of the bar represent the relative diagnostic value and implied subspecies, respectively (Figure 4.9).

Another data set annotates regions of heterozygosity, which are displayed for each selected strain (Figure 4.10). This is particularly important for outbred and wild-caught mice since inbred laboratory mouse strains have little or no heterozygosity. Heterozygous blocks are computed using a method similar to the subspecific origin HMM [98] to detect large heterozygous regions. Heterozygosity is displayed using the basic RDR approach I described.

Genome mosaic representations, such as subspecific origins and heterozygous regions, reveal the phylogenetic structure and identify introgressions between mouse strains. Existing laboratory and wild-derived strains are a mosaic of ancestral genomes that were selected for desired traits and subsequently inbred. The genome browser provides the first tool for exploring this genomic diversity at both a high level and a fine scale.

For classical laboratory strains, several additional data tracks are displayed to show local variation and haplotype structure. These data include a mosaic of possibly overlapping

Figure 4.9: Subspecific-origin assignments with diagnostic SNPs overlaid within a fine-scale window. Diagnostic SNPs are similarly indicated by color. Diagnostic SNPs were determined by [98] based upon whether each SNP variant was unique to a particular subspecies, and categorized as either fully informative (common to all members of the subspecies) or partially informative (occurring within some members of the subspecies). The SNPs with diagnostic alleles are displayed as individual bars with the height representing the confidence with which that allele indicates the particular subspecies. The subspecific origin intervals are computed using an HMM over the set of diagnostic SNPs for each strain, as described in [98].

intervals or compatible haplotype blocks that show no evidence of recombination. Within these blocks, the tool dynamically computes identity-by-descent between the selected set of strains. The MPV provides an innovative visualization of haplotype identity among classical laboratory strains based on these compatible blocks. Importantly, local phylogeny trees can be displayed for each interval.

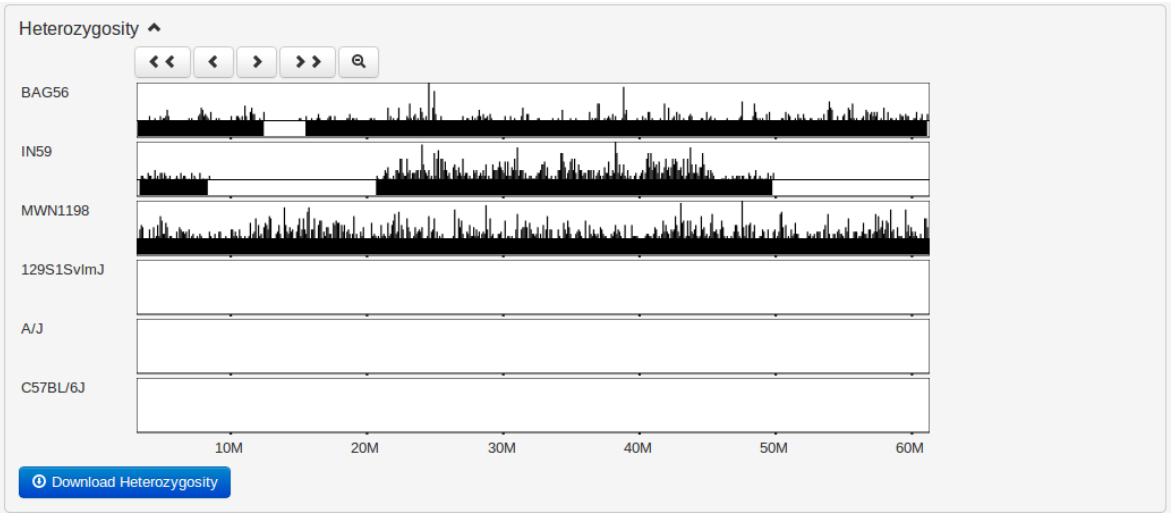


Figure 4.10: The heterozygosity track highlights heterozygous regions of the genome, which occur primarily in outbred and wild strains. Similar to SNPs, independent heterozygous calls are represented as a histogram at coarse scales and individual loci at finer scales. Blocks of heterozygosity are inferred from the individual calls using an HMM, and represent suspected outbred regions. Classical laboratory and wild-derived strains are largely inbred however they may maintain small regions of residual heterozygosity.

I include a visualization of intervals that show no evidence of historical recombination [85], as described in Chapter 2, among the classical laboratory strains (Figure 4.11). These blocks serve as not only a visualization of their own representing ancestral haplotypes but also as a framework over which other data sets and visualizations are based. This visualization is useful when analyzing other data sets in order to place those data in the context relative to these intervals. We expect that significant genomic features, specifically differences between strains, should fall within these breakpoints. Compatible intervals, as described by a Maximal-k scan in Chapter 2, can overlap at most with one other interval on each side, so intervals are displayed as horizontal bars in one of two stacked horizontal tracks. The visualization uses RDR to allow users to assess the local interval structure at both course and fine resolutions.

The compatible intervals are pre-computed in order to represent diversity our full set of classical laboratory strains. The intervals are not recomputed dynamically based on the selected subset of strains. The relationships between a small subset of samples is better

represented by the dynamically computed identity-by-descent (IBD) track, which displays the intervals in which the selected strains share a common haplotype.

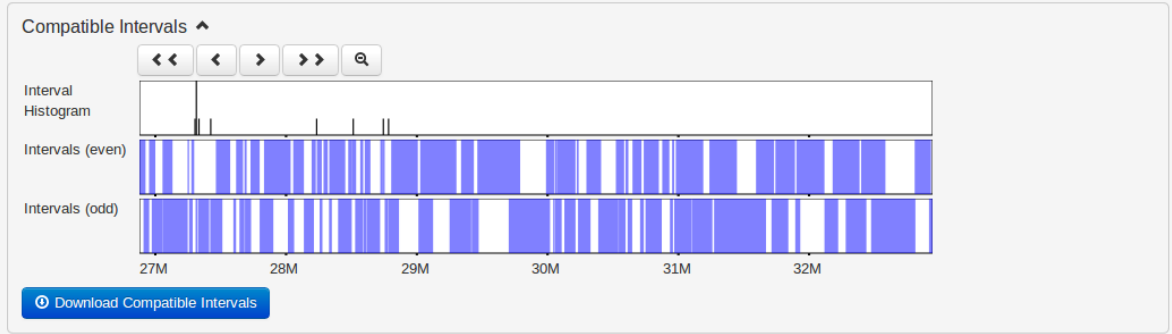


Figure 4.11: Intervals are represented as blue blocks or a density histogram, using RDR. Compatible intervals are computed as maximal overlapping four-gamete compatible regions across the genome (see Chapter 2). Such intervals have the property that no more that 2 adjacent intervals can overlap. Overlapping "even" and "odd" intervals are displayed on alternating stacked tracks.

Intervals of sequence identity are computed dynamically based on where the subset of selected strains share a common haplotype (Figure 4.12). Strains are divided into haplotype identity groups within each compatible interval based on sequence similarity (see [98]). We compute intervals of identity by descent (IBD) over the user-selected set of strains on the fly. We consider a region IBD if all selected strains are in the same haplotype group over that region. As we will see in the following data description, regions of IBD should correspond directly to identical haplotype coloring patterns as shown in Figure 4.7.

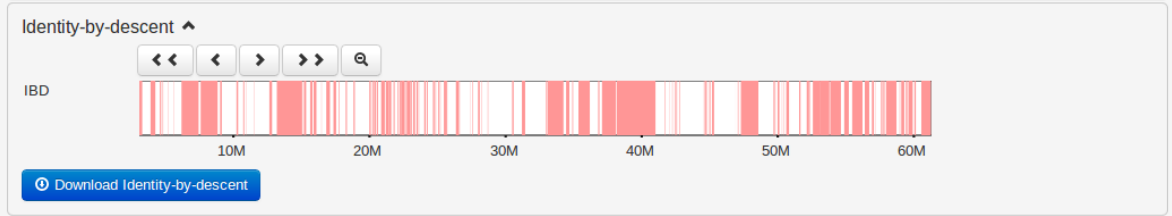


Figure 4.12: The dynamic IBD track shows intervals of sequence identity computed over the selected subset of strains. Red horizontal bars represent regions of the genome over which all selected strains are substantially identical. IBD is computed from the haplotype group assignments for each interval where strains in the same haplotype group are considered IBD. Regions of IBD are displayed where all selected strains are in the same haplotype group over consecutive compatible intervals. This visualization is computed on the fly based on the current subset of selected strains.

We also support a method for exploring the extent of shared haplotypes among the selected strains. Blocks of color are used to depict haplotype similarity. Colors are assigned and reused so that transitions are minimized. A fixed set of pastel colors is used only to easily distinguish among different haplotypes, avoiding those colors representing subspecies. The haplotype identity among the displayed strains can be understood visually as dividing strains into haplotype groups according to their color such that strains with substantially identical haplotypes are the same color. The compatible intervals shown in Figure 4.11 are the units within which the shared haplotypes (Figure 4.7) and IBD (Figure 4.12) are computed. Shared haplotypes correspond directly to a shared leaf in the local phylogenetic tree of each compatible interval. Initial haplotype colors are precomputed for all classical laboratory strains, leading to frequent color/haplotype changes at a genomic scale. Using the dynamic recoloring feature, the haplotype coloring scheme can be substantially simplified for a small sample of strains allowing more intuitive analysis. Additionally, strains may be automatically sorted by haplotype color, aligning strains with similar features.

Lastly, local phylogenetic trees are displayed by selecting a compatible interval of interest within the genome (Figure 4.13). A tree is computed within the interval based on neighbor-joining on haplotype similarity. Selected strains are highlighted according to the haplotype group they fall in, corresponding to a leaf in the tree structure. In contrast to the haplotype identity and IBD tracks, the phylogenetic trees show relative differences and possible ancestral relationships between similar haplotype groups rather than simply the group membership. Strain names are colored according to their subspecific origin to show the relationship between subspecies assignment and tree structure.

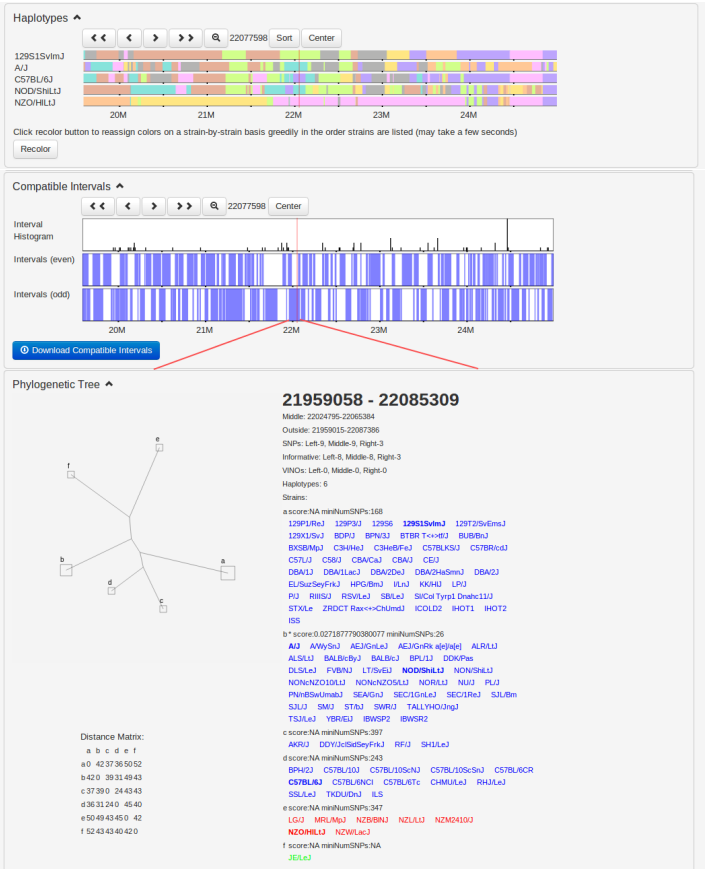


Figure 4.13: The compatible intervals and haplotype coloring are shown for a small region. A local phylogeny tree is also shown for the highlighted interval (denoted by the vertical red bar along the compatible interval track). The local phylogeny tree visualization includes the tree structure, size and location annotation for the interval the tree covers, the leaf descriptions including the strains in each leaf, and the distance matrix used to perform the neighbor-joining between leaves. Letters at the leaves of the phylogeny tree denote nodes that contain strains. The leaf descriptions show the corresponding node letter, a confidence score, and number of supporting SNPs along with the set of strains in that leaf. Each strain is colored according to its assigned subspecific origin within the tree's interval and the strains in the currently selected subset are shown as bold to allow the user to quickly identify them.

4.5 Conclusion

An instance of my genome browser and its dynamic analysis methods has been deployed to display results of our recent publication [98] at `http://msub.csbio.unc.edu`. It is currently used in comparative genome analyses of the mouse genomes presented. In the past twelve months of our tool's availability, we have had over 4,000 users make almost 50,000

queries. The tool is used by researchers to perform comparative analysis between 198 common mouse strains. MPV is particularly well suited for selecting and partitioning strains while simultaneously considering phenotypic variation as it relates to a given gene or genomic region. A recent focus of the browser has been to explore the predictive power of local phylogeny and haplotype assignments. Local comparative genomic analysis has been particularly effective in predicting phenotypic states of the available set of mouse strains given the known state of a small sample. Others have used it to predict copy-number variations in laboratory mice [81] and to narrow the intervals of QTL loci [9]. I have also used the notion of sequence similarity to inform genotype imputation by constructing a haplotype mosaic (see Chapter 3, [86]). Work is continuing to enhances the browser's support in this area.

There are many technical as well as structural improvements that can make the browser more useful, general, and effective for visualization and analysis of multiple genome data. Although the browser is constructed in a modular format, separating our data from the browser itself, to add new user-specified data types or novel visualizations requires modifications to the source configuration. In order to support a larger range of users and wider adoption, it is possible to add a simple web-based user interface for adding new tracks and visualizations within the existing framework. I present the genome browser's application to a specific data set here, but it is suitable to other organisms and other data types where comparative analysis of multiple genomes is useful. I will discuss another genome browser application based on MPV in Chapter 6. Likewise, the tool could provide an API for custom analysis of the existing data set. Improvements to my model of local phylogeny may also be included to enhance the visualization tool. I will discuss an extension of the compatible intervals method useful for incorporating additional sources of data in Chapter 5.

Chapter 5

More Robust Compatible Intervals

In this chapter, I introduce an accurate method for the discovery of local phylogenetic structure among individuals of a species using genotype data with a high rate of genotyping error or confounding biological variation. This is an extension of methods used to partition the genome into regions exhibiting no evidence of historical recombination called compatible intervals. This is accomplished by relaxing the four-gamete rule, which tests for conformance to a perfect phylogeny, in which a number of loci may be ignored. The method I developed is a variant of the Character Removal Problem [71]. This allows my method to accurately discover true compatible regions while masking the effects of possibly erroneous loci. In this chapter, I describe this modified algorithm and demonstrate its effectiveness on simulated and real-world data sets with a range of error rates.

5.1 Introduction

In Chapter 2, I described methods for partitioning the genome into blocks showing no evidence of historical recombination called compatible intervals. Here, I describe an extension of this model which relaxes the four-gamete rule to allow us to construct intervals in the presence of possibly erroneous data. These "errors" may be the result of true errors introduced during the physical genotyping process, be it off-target hybridization on a microarray

or misreading of short shotgun sequencing reads. Errors may also be introduced in the computational derivation of genotype data from raw sequencing or microarray data. This type of error includes misclassification of probe intensities into allelic groups and a wide variety of errors which can occur during sequencing read alignment. Using a scaffold-based alignment approach [16], reads may align incorrectly to distant homologous areas of the genome or the reference genome may not correspond accurately to the sequenced genome, for example omitting polymorphic structural variations.

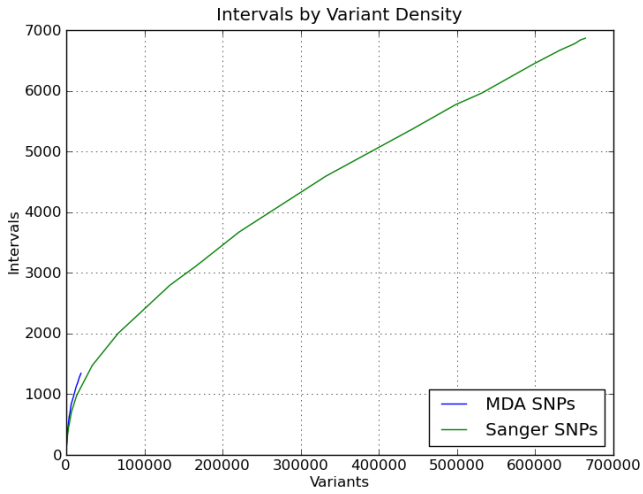
In addition to genotyping errors, there are also rare biological phenomena, such as homoplasy, which, although not strictly errors, violate assumptions made by our model and may confound our ability to accurately determine the phylogenetic structure. Homoplasy is the occurrence of multiple mutations at a single locus, violating the assumption we made in the infinite-sites model that no site may mutate more than once. A homoplasy event or random genotyping error results in the permutation of a single allele which may cause a violation of the four-gamete rule, thus appearing as an incompatibility and implying a recombination breakpoint. My goal is to, in some sense, "cover up" these kinds of perturbations to preserve our ability to find the true underlying phylogeny structure. This type of error tends to have a noticeable signature in that it most often introduces a novel distribution of alleles - or strain distribution pattern (SDP) - within the local genotypes. As discussed in Chapter 2 pertaining to the complexity of computing compatible intervals, any set of mutually compatible SDPs is much smaller than the set of all possible SDPs. Thus, any random SDP is unlikely to be compatible within its current interval. These erroneous SNPs are incompatible with a large proportion of neighboring SNPs uniformly on both sides (see Figure 5.5). In contrast, recombination breakpoints cause SNPs on one side of the breakpoint to be uniformly incompatible with SNPs on the other side, but compatible with proximal SNPs within its own interval. The algorithm I describe for constructing these robust intervals does not explicitly model this signature, but has the effect of ignoring those SNPs which cause the most bidirectional

incompatibility.

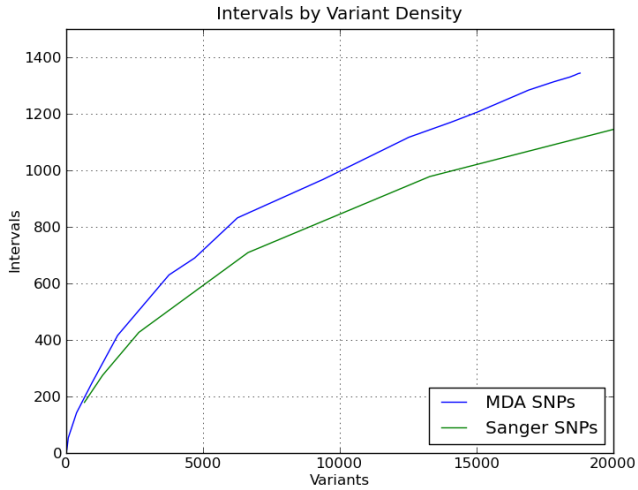
5.2 Effects of Genotyping Error and Homoplasy

In Chapter 2, I described the computation of compatible intervals using a set of dense genotypes discovered using the MDA [97]. These data consist of about 500,000 SNPs across the genome. In an attempt to improve my characterization of the local phylogenetic structure, I recomputed compatible intervals using the Sanger SNP set described in Chapter 3, consisting of over 12 million SNPs, nearly 25 times the density of the MDA. Figure 5.1 illustrates the growth rate of the intervals detected using increasingly dense samples of these two data sets.

As one might expect, more intervals are detected as more SNPs are sampled, increasing our precision in identifying probable recombination breakpoints. As illustrated in Chapter 2 and using my imputation method in Chapter 3, the compatible intervals discovered using the MDA SNPs provide a very accurate characterization of the local phylogenetic structure within the population of classical laboratory mice. However, it is disconcerting to note that, with the incorporation of the much denser set of Sanger SNPs, the number of detected intervals continues to increase. Within this population, the number of true historical recombinations, and therefore compatible intervals, should be fixed. Therefore, I expect the number of detected compatible intervals to asymptotically approach the true number as the sampling density increases. A simulation model supports this hypothesis, as shown in Figure 5.2. With no error (0%), the number of detected intervals asympototically approaches the true number of recombinations (552). When random error is introduced into these simulated genotypes, the number of detectable recombinations continues to grow as additional SNPs are sampled. In fact, I illustrate in Figure 5.3 that the number of intervals detected in excess of the "truth" grows linearly with the error rate.



(a) Interval detection by SNP sampling



(b) Interval detection by SNP sampling (closeup)

Figure 5.1: These plots illustrate the change in number of detected intervals based on the SNP sample size. As we expect, the number of detected intervals increases with the sample size. The increase is non-linear, implying a saturation of informative variants to detect true recombinations.

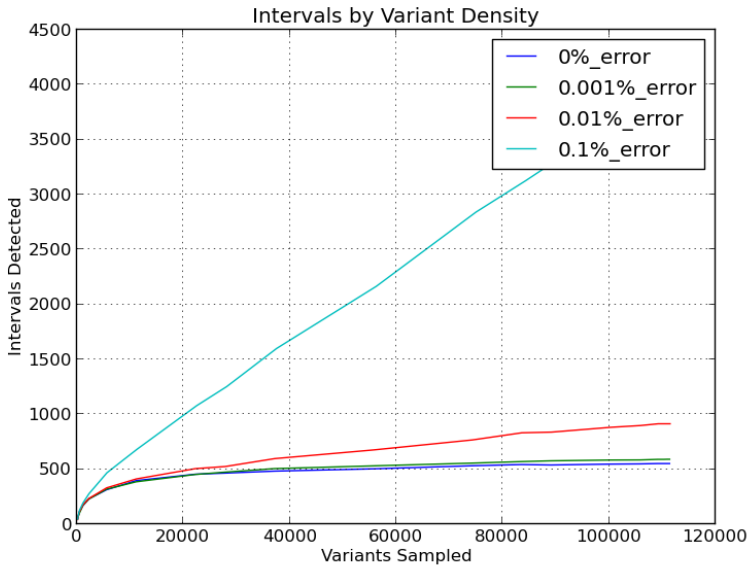


Figure 5.2: Interval detection growth using simulated genotypes. These data use a simulated model of population inheritance under the infinite-sites model with increasing proportions of random error introduced. The growth with 0% error illustrates the asympototic trend indicative of "true" intervals. Those with error introduced are increasingly dominated by the linear error effect.

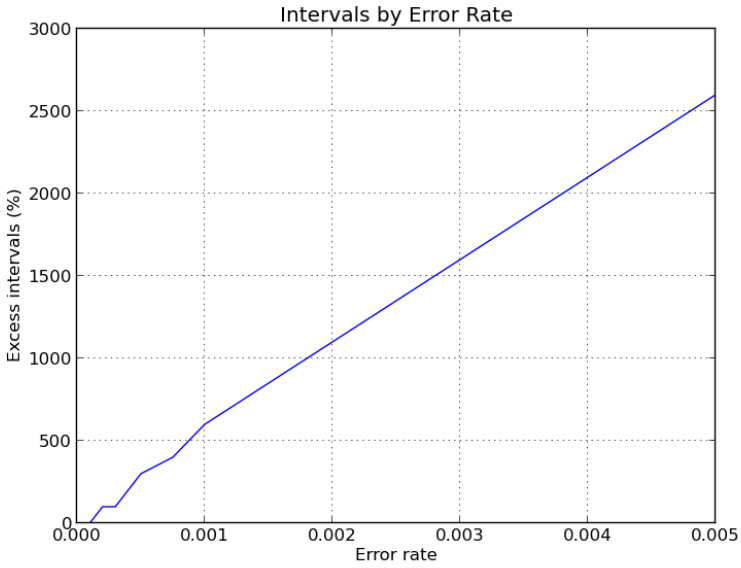


Figure 5.3: The number of intervals detected in excess of the ground truth, in simulation, is shown in relation to the error rate. We see that, as the error rate increases, the proportion of excess intervals detected increases in a roughly linear pattern.

This model represents a set of observed genotypes and corresponding compatible intervals as a combination of "true" and "error" signals. We directly observe this pattern in the growth of detectable intervals, where we expect the "true" recombination breakpoints to contribute to an asymptotic increase in detectable intervals, and the "error" in genotypes to contribute to a linear increase. Using this model, we can decompose an observed interval detection curve into these two components using least-squares regression. This decomposition of the intervals computed over the Sanger SNPs is shown in Figure 5.4.

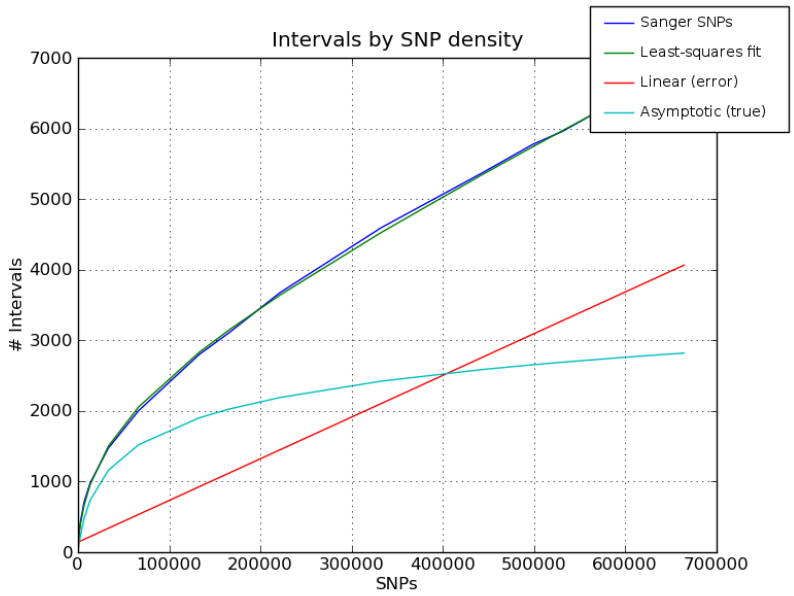


Figure 5.4: The growth rate of detectable compatible intervals at increasing sample sizes over the Sanger SNPs. This growth curve is decomposed into asymptotic and linear components using a least-squares fit. This mixed model provides a good fit of the observed interval growth pattern.

The partitioning of true compatible intervals into many smaller regions due to "error" including genotyping error and homoplasy reduces our ability accurately characterize the local phylogenetic structure, leaving fewer variants in each interval with which to distinguish local inheritance structure. I propose a method to implicitly disregard those "errors" contributing to the linear increase in observed intervals, allowing my method to better approximate the set of "true" compatible intervals bounded by historical recombinations.

5.3 Background

There exist a number of previous studies that consider partitioning a set of genomes into blocks which exhibit high linkage or, equivalently, low levels of recombination. These methods include those based on linkage disequilibrium (LD) [30, 68] and perfect phylogeny [40, 76]. These are described in Section 2.2. Each of these methods has advantages and disadvantages, striking a balance between specificity and susceptibility to error. Methods based on perfect phylogeny err on the side of describing exactly the genomic structure within a population while breaking the genome into unnecessarily small blocks due to genotyping error or due to violations of the infinite-sites model. Linkage disequilibrium measures are necessarily parametric and are thus partially resistant to small-scale fluctuations which may be a result of error, but do not necessarily describe a perfect phylogeny.

The model I have developed retains the notion of perfect phylogeny while relaxing the compatibility rules to better predict the true phylogenetic structure where there is error in the source genotypes. Methods for discovering a single perfect phylogeny over a fixed set of genotypes have been described previously [35, 36, 31]. Gysel et al. [36] describes a general method for solving the Perfect Phylogeny Problem for a set of haplotypes with an arbitrary number of allelic states, missing data, and allowing for character removal. While this method and the others referenced (for a survey, see [31]) go to great lengths to distill a perfect phylogeny from data which may be discordant in many ways, they still attempt to construct a single, global phylogeny to describe an entire data set. The method I describe uses a variant of some of these approaches, notably character removal, in the context of local phylogenies to improve our resistance to error.

The Perfect Phylogeny Character Removal problem can be reduced to the Minimum Vertex Cover problem - basically, to find the fewest SNPs which must be removed such that there are no remaining pairwise incompatibilities. The details of this reduction is given in the next section. The Minimum Vertex Cover problem is a classic NP-complete problem [42] with

a naive solution in $O(2^n)$ time. There also exist many efficient approximations which are appropriate for common cases [95].

5.4 Method

In a standard compatible interval cover of any type (left-to-right, right-to-left, Uber, Max-k), a requirement for any interval is that all pairs of SNPs included in an interval are four-gamete compatible. The optimization described in Chapter 2 was to find the least ambiguous and most informative set of such intervals. In order to overcome erroneous data, I redefine the notion of a compatible interval to allow us to ignore a number of SNPs which may be erroneous. My goal is to find a Max-k set of intervals for which no interval exceeds some number, r, of SNPs that must be ignored such that all remaining SNP pairs are four-gamete compatible. I will use R_r to represent the class of intervals which contain r such SNPs.

I will build upon a simple definition of an interval cover to define a maximal set as I did in Chapter 2. We may perform a left-to-right (LR) scan much as we have done previously. Starting at some SNP, S_i, we add SNPs S_{i+n} to our interval while each SNP is compatible with all previous SNPs $\{S_i, S_{i+1}, \ldots S_{i+n-1}\}$. In addition, we will allow up to r SNPs to be ignored such that if we ignore these SNPs, the remainder will all be pairwise compatible. When we reach a SNP S_{i+n} which contains one or more incompatibilities with previous SNPs, we must attempt to satisfy our condition that no more than r SNPs are ignored. This problem can be reduced to the (weighted) Minimum Vertex Cover Problem. The graph is constructed as follows. Each unique SDP in an interval is a node with weight equal to the number of SNPs with that SDP. If two SDPs are incompatible, we add an edge between the corresponding nodes. Solving for a minimum cover, M, entails finding the minimum total weight of nodes which must be removed such that every edge is removed. The minimum

cover, M, represents the set of SNPs which must be ignored so that there are no more incompatibilities. If $|M| \leq r$, we can satisfy our character removal condition and the SNP set is a valid R_r interval. When a new SNP causes $|M| > r$, we record the interval $[S_i, S_{n-1}]$ and set $i = n$, beginning the next interval immediately after where the last left off.

5.4.1 Minimum Vertex Cover Solution

The Minimum Vertex Cover problem is a well-known NP-complete problem [42] and, as such, has no known polynomial solution. There exist efficient (polynomial) approximations [95]; however, these, by definition, may not discover the optimal solution. Since the number of unique SNPs in an interval is bounded (Chapter 2, [77]), the size of the graph is also necessarily bounded. My method uses an exponential branch-and-bound algorithm to solve for a Minimum Vertex Cover. This algorithm is guaranteed to find an optimal solution, and is tractible, in practice, because the size of the graph is bounded.

I perform a recursive depth-first traversal of the incompatibility graph, bounding the search based on the removal threshold, r. My method considers the removal of each subset of vertices such that there are no remaining edges, backtracking if the total weight of removed nodes exceeds r. The search is further bounded if any solution is found in which the total weight of removed vertices is under the threshold r. If such a solution is found at any point during the search, I stop and declare the interval under consideration achievable given r or fewer removals. If no solution is found, there exists no subset of r SNPs which can be removed to make the interval fully compatible, so the interval is closed at the preceding SNP.

5.4.2 R_r Covers

We can trivially compute R_r left-to-right (LR) and right-to-left (RL) R_r covers by scanning in either direction. Similarly, the Uber scan described in Chapter 2 can be applied, in a modified form, to construct an R_r cover containing all possible intervals with r ignored SNPs.

The Uber cover is constructed by scanning left to right, but with a trailing start rather than beginning the next interval where the last left off. The first phase of the scan is the same as the LR scan (Algorithm 1), adding SNPs to the interval as long as we can satisfy our removal criteria (by adding incompatibilities to our graph and solving the Minimum Vertex Cover). The Uber cover should contain all possible intervals that satisfy R_r. When a SNP is added which increases our minimum cover over r, we store the interval $[S_i, S_{i+n-1}]$ and add S_n and all associated incompatibilities to the compatibility graph. The minimum cover must now be $r+1$, so the interval starting position S_i, is increased, removing the corresponding nodes and incompatibility edges from the compatibility graph until the minimum cover is again r. This represents the beginning of the next possible adjacent R_r interval. The end position is then extended again until the minimum cover exceeds R_r, completing the interval.

The portion of the compatibility matrix "under" the R_r Uber cover represents those pairs of SNPs which are either compatible or can be ignored as possibly erroneous with the removal condition r. I will refer to this division as the R_r "horizon", where all SNP pairs under the R_r horizon are compatible if we allow up to r SNPs to be ignored (Figure 5.5). In simpler terms, we can consider everything under this horizon compatible if we allow for some error term r. I use this model to redefine what I consider "compatible" when computing a Max-k cover. I recompute C_{LR}, C_{RL}, C_{Uber}, and C_{Max-k} using this pseudo-compatibility model. These covers have the same properties as the standard covers described in Chapter 2, including, importantly, $|C_{LR}| = |C_{RL}| = |C_{Max-k}| \leq |C_{Uber}|$. It is also the case that every interval must necessarily conform to R_r. Additionally, our Max-k cover meets the same criteria: it contains the minimum number of R_r intervals necessary to cover the genome, and each such interval is maximal in size.

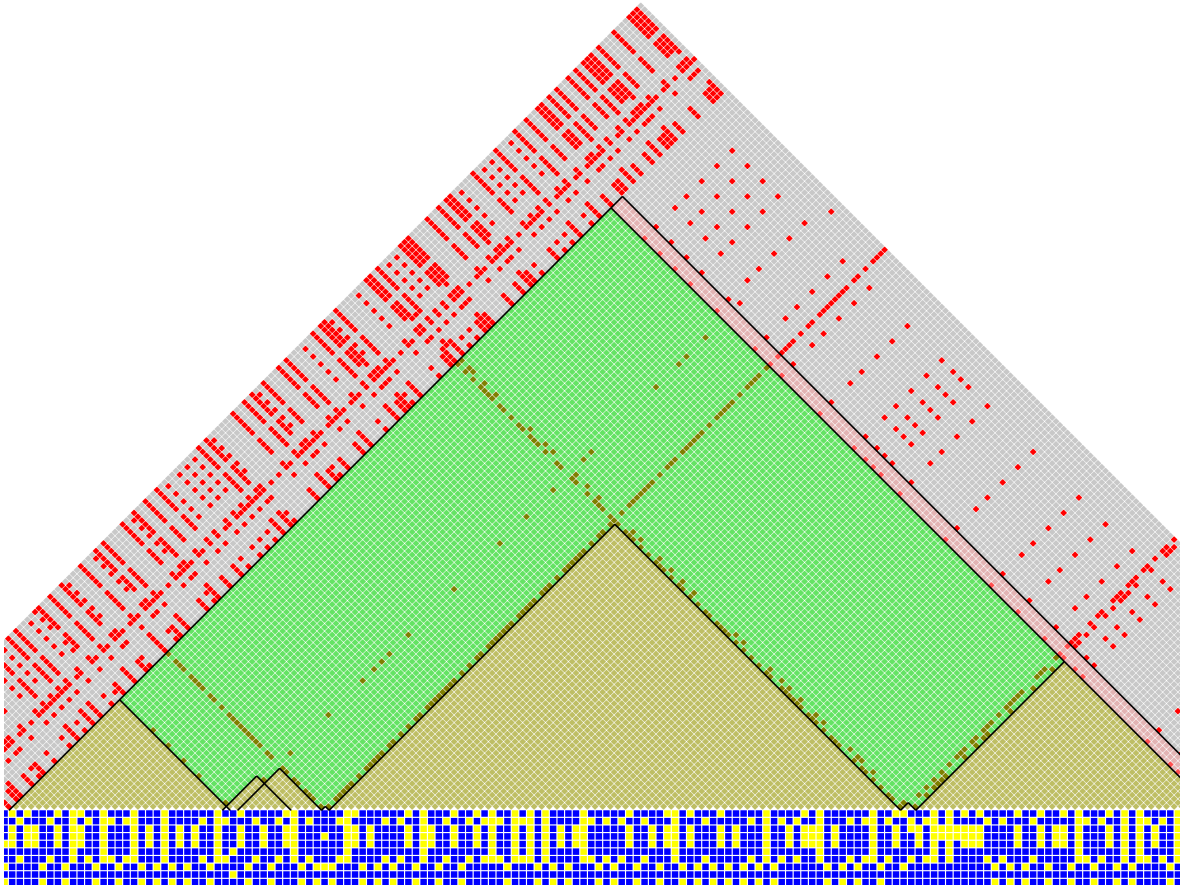


Figure 5.5: A subset of a simulated set of haplotypes with the corresponding compatibility matrix and interval sets. The blue and yellow matrix at the bottom shows a set of haplotypes where each row represents the haplotype for a single sample. Each column represents a SNP, where blue is the majority allele (0) and yellow is the minority allele(1). The compatibility matrix show the pairwise four-gamete compatibility between SNPs where gray is compatible and red is incompatible. There are three intervals sets shown, shaded with the colors pink, green, and yellow. The yellow intervals represent the Max-k cover for exactly the given haplotypes. The green represents the Max-k cover for the same set of haplotypes before 0.1% error was introduced. Pink represents the Max-k cover using an R_5 relaxation as described in this chapter.

5.4.3 Complexity

The extra work required to determine which SNPs to remove/ignore affects the complexity of the LR, RL, and Uber scanning algorithms. Computing the unique cores and Max-k cover is not different from what was described in Chapter 2 since these operate on the previous interval sets. During a scan, the extra Minimum Vertex Cover problem must be solved whenever an incompatibility is introduced. The Minimum Vertex Cover Problem has been thoroughly studied [42, 95] and, while it is a classic example of an NP-complete problem [42], there

exist many efficient approximation algorithms. The branch-and-bound algorithm I describe is $O(2^n)$ where n is the number of nodes in the graph - in this case, SDPs. As I discussed in Chapter 2, the number of mutually compatible SDPs is bounded by the number of samples, $n \leq 2m - 3$. So, my solution achieves $O(2^m)$. Optimizations of the Minimum Vertex Cover Problem [95] or the character removal problem generally [35] could be trivially incorporated to improve the performance of this method.

5.5 Results

I tested my R_r compatible intervals using both simulated genotype data as well as real data from laboratory mice. Using simulated population data to test my model allows us to create genotype data with known phylogenetic structure, introduce various errors into the data, then find out if we can recover the "true" underlying phylogeny structure. In the likely cases that our simulation model is oversimplified and does not represent the range of biological and experimental variation found in real genotypes, I also validate my R_r intervals using the imputation method described in Chapter 3. By using the discovered phylogenetic structure to make predictions about imputed genotypes, I can obtain a measure of how well my local phylogenies reflect the true population structure.

5.5.1 Simulation

I simulated a population of genotypes consisting of 100 haplotypes undergoing mutation at a rate of 0.0001 per locus per generation and recombination at five per generation, for 50 generations. I used a coalescent model with non-overlapping generations, so that at each generation, 100 progeny are created, each with a random, possibly duplicate, "parent" from the previous generation. I chose a random sample of 10 individuals from the resulting population as my experimental group. I computed the normal Max-k cover of these genotypes as

described in Chapter 2 and used this as my baseline "truth". Figure 5.5 shows a subset of the genome containing one of these intervals, shaded green.

To test the ability of my extended model to overlook errors, I introduce random allelic errors into the simulated data set. I simulate genotyping error or, equivalently, homoplasy events occurring in a single sample by randomly permuting a single allele. The error rates are given as a probability of error per locus, per sample. Note that an error rate of 0.1 over a dataset with ten samples would result in an average of one error in each SNP.

In order to determine the accuracy of an R_r cover, we must determine the concordance of interval sets. For example, we would like to find out how well our ground truth Max-k cover "agrees" with our R_r cover. I find the maximum pairwise overlap of two interval sets, such that each interval in one set may be "paired" with at most one interval in the other and vice versa. Because two covers may have different numbers of intervals, this is a nontrivial problem. However, we can achieve a close approximation using a greedy pairing of intervals for maximum overlap. I quantify the "agreement" between two interval sets by the total pairwise overlap. The pairing of interval sets to determine agreement is shown in Figure 5.6. This value must be less than or equal to the minimum total interval size of the two covers. I represent the agreement value as a fraction of the total interval size of the base truth, as shown in Figure 5.8.

In simulation, we can clearly see the effect of the SNP removal. Figure 5.5 shows an example of a region of my simulated dataset with 0.1% error. This region contains one true interval, computed over the same region without error (shaded green), normal Max-k intervals (yellow), and an R_5 interval, closely approximating the true interval regardless of the erroneous SNPs. Four SNPs with errors introduce abundant incompatibilities. These necessarily partition the normal Max-k cover, but are ignored as likely errors when we allow a number of incompatible SNPs in the R_r scan. Figure 5.7 shows visually the concordance of R_r C_{Max-k} and the ground truth C_{Max-k} for several values of r. We can see qualitatively how

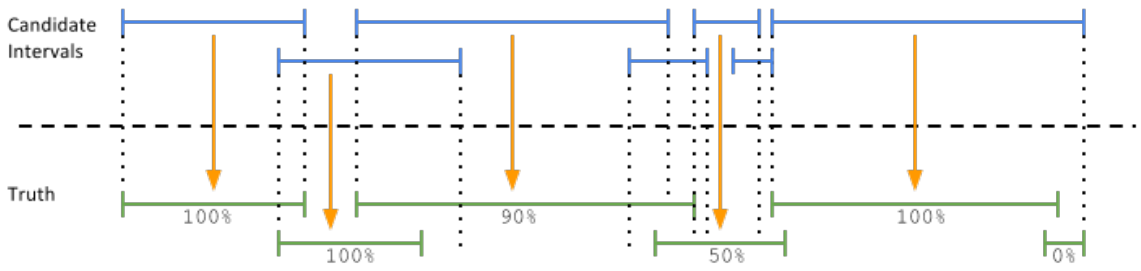


Figure 5.6: The maximum pairwise overlap between two possible sets of intervals. This is a hard problem, but can be approximated using a greedy approach. Determining the pairwise overlap relative to a ground truth lets us assess the accuracy of candidate compatible interval covers. Two interval sets are shown, a set of candidate intervals in blue and a "ground truth" set in green. Each green intervals is paired with the blue interval that overlaps the most. Orange arrows indicate the assignment of candidate to true intervals to maximize the total overlap, the percetages indicating the fraction of the true interval which is overlapped.

the interval cover predicted by the R_r Max-k scans approach the true intervals. Figure 5.8 shows the percent concordance between R_r intervals and the ground truth as r changes. As we can see, the accuracy increases asymptotically as r increases. Shown in Figure 5.8, $r = 0$ corresponds to the original Max-k cover described in Chapter 2. In simulation, the accuracy of the predicted interval cover doubles between $r = 0$ and $r = 7$, increasing the concordance of these intervals to the true local phylogenetic structure and regions of shared inheritance.

5.5.2 Imputation of real data

I also validate my R_r compatible intervals using real-world genotype data. In previous studies, the local phylogenetic structure was computed based on microarray-based SNP data. This has a relatively low SNP density compared to other methods, but also a relatively low error rate after it has been properly curated. In order to improve the overall accuracy and precision of my model of local phylogeny, a method is required which can be applied to other, denser sources of genotypes while remaining resistant to possible errors. I computed an R_r cover over a set of SNPs derived from high-throughput sequence data produced by the Sanger Institute [43], with a density over 20 times that of our MDA data in classical laboratory strains. The sequencing technology used in the study by Keane et al. has an error rate

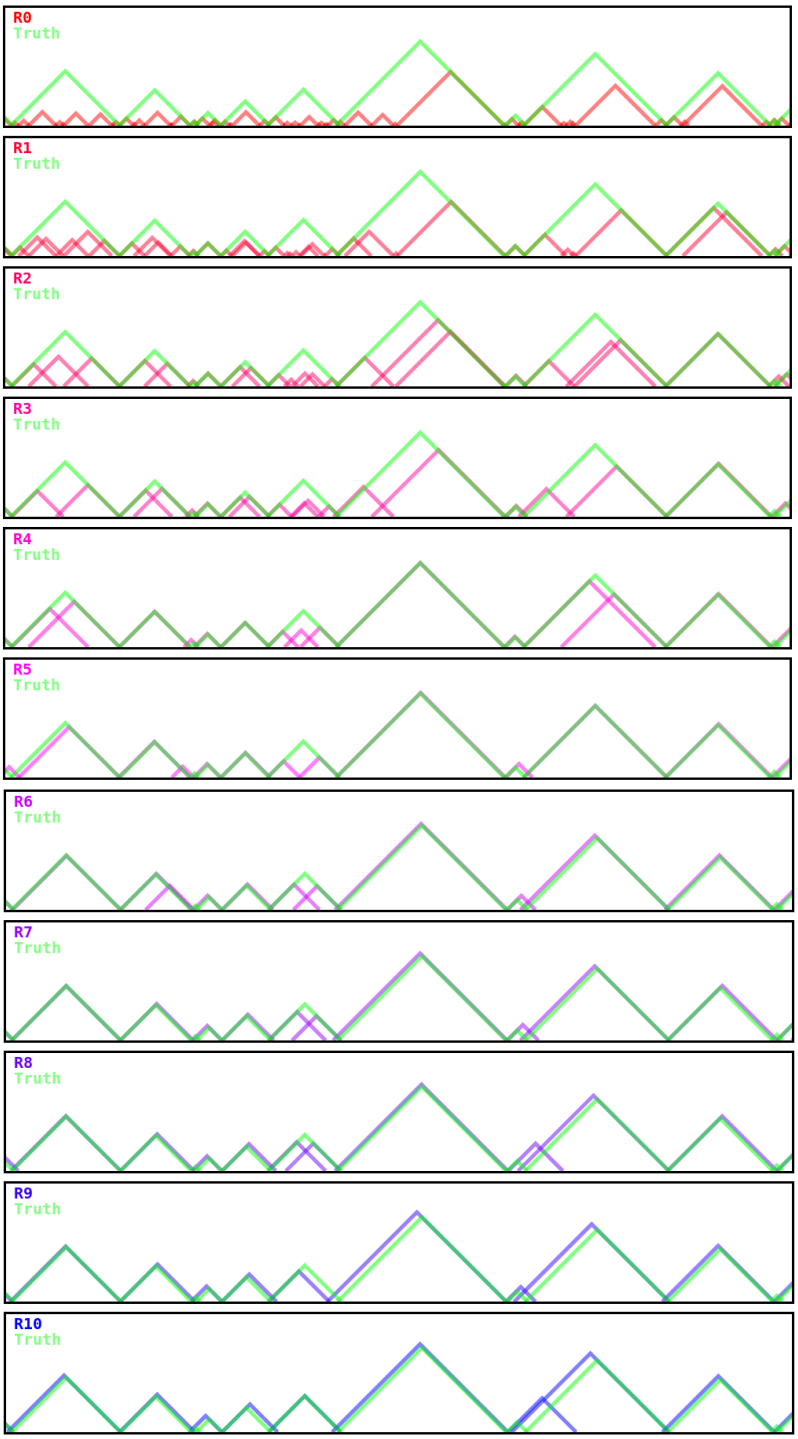


Figure 5.7: The concordance of R_r C_{Max-k} covers with the ground truth for several values of r. The green peaks represent Max-k intervals over a simulated data set with no error. The other sets of peaks represent R_r Max-k intervals over the same simulated data with 0.001 error introduced. As r increases, the R_r interval cover approaches the true interval cover, up to a point where the R_r intervals become overly general.

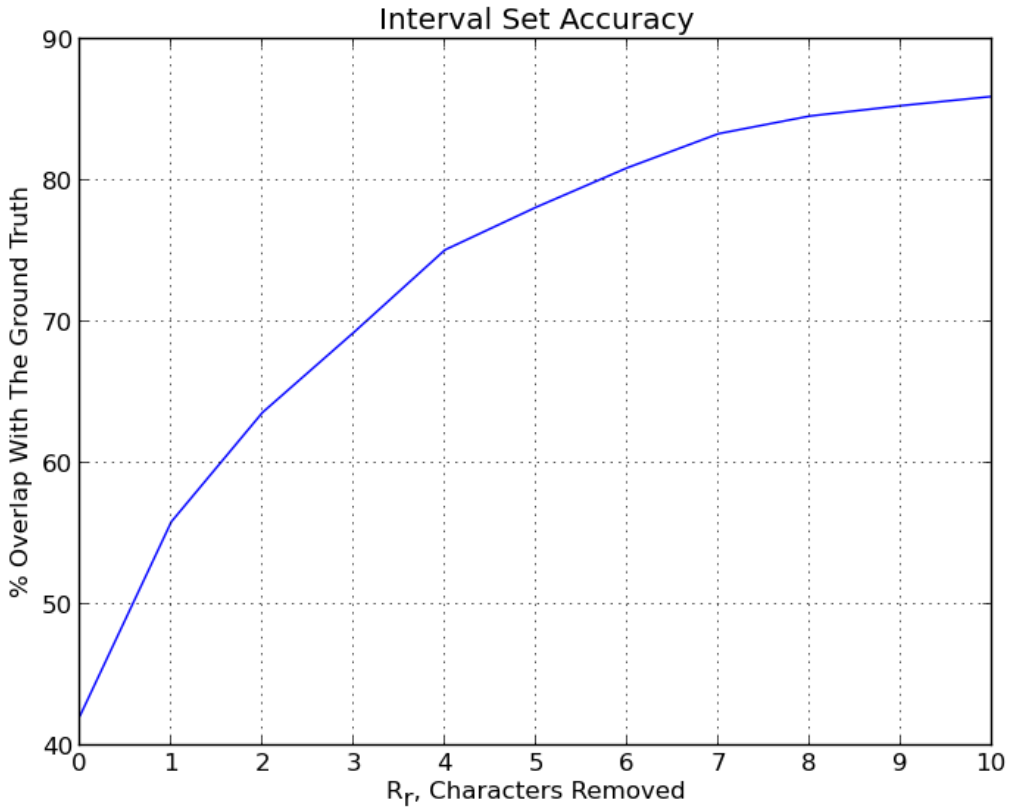


Figure 5.8: Here I show the concordance of R_r interval covers with the ground truth as r increases. The Y-axis shows concordance as a percentage of interval set overlap between the R_r C_{Max-k} and the truth. The X-axis corresponds to increasing values of r. Note that as r increases, the accuracy increases asymptotically.

approximately consistent with my simulation trials.

I computed an R_5 Max-k cover over 11 classical laboratory mouse strains, 129S1SvlmJ, A/J, AKR/J, BALB/cJ, C3H/HeJ, C57BL/6NCI, CBA/J, DBA/2J, LP/J, NOD/ShiLtJ, and NZO/HILtJ . While a normal Max-k cover produces 6,868 intervals, when I allowed $r = 5$ SNPs to be ignored, the minimum cover size is reduced to 1,898 in R_5C_{Max-k}. In comparison, the Max-k cover using the MDA data over the same region consists of 1,342 intervals. This cover is illustrated in Figure 5.9. To assess the accuracy of the imputation using these intervals, I use a leave-one-out analysis. Each of the 11 sequenced strains is imputed independently using the remaining 10. The error rate is then computed from the difference between the imputed genotype and the reported genotype based on sequencing. Table 5.1 shows the

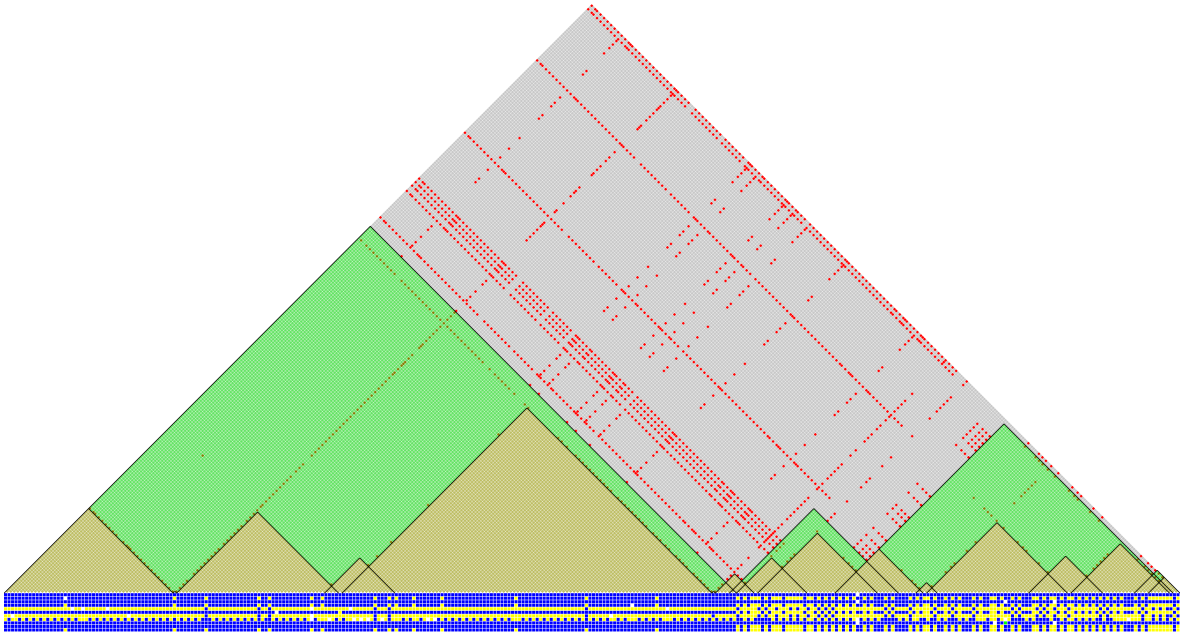


Figure 5.9: A region of the genome showing the C_{Max-k} (yellow) and R_5 C_{Max-k} (green) covers over the set of Sanger SNP data for 11 sequenced strains. The yellow intervals represent the Max-k cover using the standard algorithm described in Chapter 2. The green represents the R_5 Max-k cover for the same set of haplotypes. We see the same signature of likely erroneous SNPs as we saw in the simulated data, with occasional SNPs being incompatible with proximal SNPs on both sides. These are ignored to construct more accurate and informative compatible intervals.

genome-wide imputation accuracy in high confidence regions using C_{Max-k} derived from MDA genotypes and R_r C_{Max-k} over the Sanger SNP set. As discussed in Chapter 3, we assign high confidence to those regions within which a strains share a haplotype from which we can impute genotypes. I exclude C57BL/6J that was imputed using MDA because this strain was not explicitly sequenced as a part of the Sanger project. This increases our error across the board because we have fewer source genotypes from which to impute, and significantly impairs our ability to impute the closely related strain C57BL/6NCrl. Regardless, I imputed nine of the eleven strains more accurately than we could with MDA. Three strains, C57BL/6NCrl, NOD/ShiLtJ, and NZO/HlLtJ have slightly higher error rates, likely because they are more closely related to the excluded C57BL/6J. The average error rate using an R_5 Max-k cover is 0.07%, an improvement from the 0.08% error using MDA.

Strain	C_{Max-k} MDA error %	R_5C_{Max-k} Sanger error %
129S1SvlmJ	0.07	0.05
A/J	0.09	0.05
AKR/J	0.11	0.09
BALB/cJ	0.08	0.04
C3H/HeJ	0.09	0.03
C57BL/6J	0.08	
C57BL/6NCrl	0.04	0.10
CBA/J	0.08	0.04
DBA/2J	0.10	0.07
LP/J	0.07	0.04
NOD/ShiLtJ	0.10	0.11
NZO/HlLtJ	0.10	0.15
Average	0.08	0.07

Table 5.1: Comparison of the leave-one-out imputation error rate for C_{Max-k} over the MDA SNPs, as described in Chapter 2, and R_5C_{Max-k} over the Sanger SNPs. This table shows error rates in genotypes imputed with high confidence, as described in Chapter 3.

As discussed, the rate of error in high confidence imputed genotypes using MDA approaches the error rate we expect in high-throughput sequencing, so we should not expect a dramatic improvement using additional sources of data. This approach does, however, allow accurate imputation using other sources of data, imputing additional samples, or improving our resolution in regions with unusual variation (for example, structural variants). We can see that, by allowing character removal, we can improve genome-wide imputation accuracy, by eliminating presumably erroneous variants from consideration when we construct the local phylogenetic structure. Using a leave-one-out method, we can assess the accuracy of the imputation method, but these genotypes do not tell us anything new. However, this validates my modified method for constructing compatible intervals using erroneous data in a real-world context.

5.5.3 Implications about Genotyping Error and Homoplasy

Using my model of the effect of genotyping error and homoplasy on compatible interval detection, we can gain a better understanding of the relative and, perhaps, absolute frequency of these events. I suggest two interpretations of this model, one to infer the absolute rate of erroneous intervals (including both genotyping error and homoplasy) from the number of detected intervals, and another to infer the relative abundance of true genotyping error and homoplasy contributing to the increase in intervals.

Figure 5.3 illustrates the linear relationship between errors and detected intervals. In my simulation model, this corresponds to approximately one extra interval detected for each error. Note that the maximum number of new intervals an error may produce is two if it is immediately incompatible with both adjacent SNPs. If this holds true for the Sanger SNP data, we can infer a per-base error rate from the slope of the linear fit component. As illustrated in Figure 5.4, there are approximately 0.006 excess intervals introduced per SNP. So, if there is a 1:1 correspondence between errors and intervals, we can estimate a 0.006 error rate per SNP, or 0.0005 per sample per SNP. This is consistent with my claim that the error rate in my high confidence imputation (Chapter 3), 0.08%, or 0.0008 per sample-SNP, is on the order of the true error rate in the data. Recall that this "error" consists of both experimental (genotyping) error and homoplasy which violates our infinite-sites assumption. We can attempt to disentangle these by using some basic biological insights.

There are four nucleotides which make up DNA: adenine (A), cytosine (C), guanine (G), and thymine (T). A and G are in a class of molecules called "purines" and C and T are "pyrimidines". Mutations from purine to purine or pyrimidine to pyrimidine are called "transitions", and mutations between purines and pyrimidines are called "transversions". Figure 5.10 illustrates the possible transitions between nucleotides, where blue indicate transitions and orange indicate transversions. It is known that transitions naturally occur twice as often as transversions [49]. If we assume there is no genotyping error and our data is a result of only mutation

(including homoplasy), we expect to see a 2:1 ratio of transitions:transversions. I use E_{Ts} and E_{Tv}, representing the fraction of "error" in transitions and transversions, respectively. It holds that $E_{Ts} + E_{Tv} = 1$. Overall, the Sanger SNP data support this model, with $E_{Ts} = 0.67$ and $E_{Tv} = 0.33$.

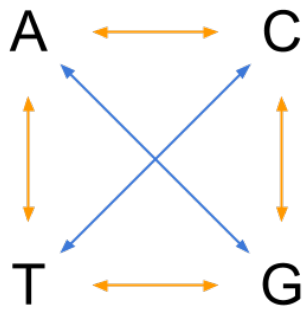


Figure 5.10: The four DNA nucleotides, adenine (A), cytosine (C), guanine (G), and thymine (T), are divided into purines (A and G) and pyrimidines (C and T). Mutations from purine to purine or pyrimidine to pyrimidine are known as transitions (blue). Mutations between purines and pyrimidines are known as transversions (orange).

To categorize the SNPs causing erroneous intervals, I separated the SNPs representing transitions from those representing transversions and independently computed compatible intervals (Figure 5.11). The observed interval detection curves for transitions and transversions are independently decomposed into asymptotic and linear components. As expected, the asymptotic components closely correspond with one another, indicating that these disjoint SNP sets reflect the same set of true compatible intervals. However, the linear "error" components are not the same. The slopes of these lines indicate the total error attributable to transitions and transversions, respectively. I observe $E_{Ts} = 0.581$ and $E_{Tv} = 0.419$. This ratio (˜7:5) does not match the expectation if these extra intervals are due only to homoplasy (2:1).

Truly random genotyping errors, unlike homoplasy, should have no such bias. An A may be just as likely to be mis-identified as a C, G, or T. Truly random errors are, in fact, twice as likely to cause a transversion as a transition. The observed transition:transversion ratio (˜7:5)

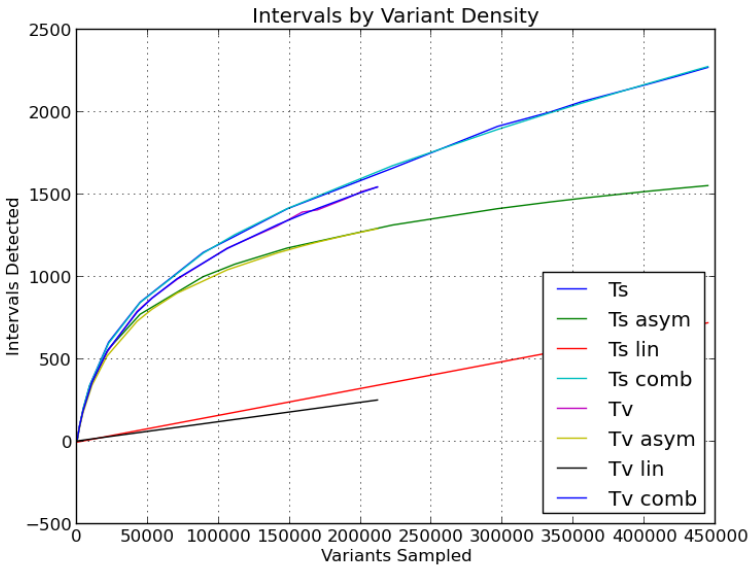


Figure 5.11: The growth rate of detectable compatible intervals at increasing sample sizes over the Sanger SNPs, separated into transitions (Ts) and transversions (Tv). Each observed curve is decomposed into asymptotic and linear components. As expected, the asymptotic fits align with one another, indicating that these disjoint SNP sets reflect the same set of true compatible intervals. The difference between the two error lines reflects the variance between the "error" rate in transitions and transversions. There appear to be more errors in the transition SNPs due to the increased chance of homoplasy with respect to transversions.

also does not match this expectation based on random genotyping error (1:2). However, we can infer a weighted combination of homoplasy and genotyping error contributing to the observed ratio.

I represent the relationships between error probabilities as $E_{Ts} = \frac{E_G}{3} + \frac{2E_H}{3}$, $E_{Tv} = \frac{2E_G}{3} + \frac{E_H}{3}$, and $E_G + E_H = 1$ where E_G and E_H represent the fraction of error attributable to genotyping error and homoplasy, respectively. Solving this system of equations when $E_{Ts} = 0.581$ and $E_{Tv} = 0.419$ yields $E_G = 0.257$ and $E_H = 0.743$. This indicates that a large majority of "erroneous" intervals are, in fact, caused by homoplasy, violating the infinite-sites model, rather than actual genotyping error.

5.6 Discussion and Conclusion

I have described a method for partitioning a set of genotypes into accurate local phylogenies in the face of errors. In Chapter 2, I describe compatible intervals computed using SNPs from the Mouse Diversity Array. While these intervals have proven very informative (Chapter 3), to improve the precision and accuracy of my model, we would like to include additional sources of genotypes. In this Chapter, I consider computing compatible intervals using SNPs derived from high-throughput sequence data. These data provide much denser genotypes, but may have significant error that would otherwise confound my model of local phylogenetics. I address this by relaxing the model of compatible intervals used in Chapter 2 to include a number of, presumably erroneous, ignored SNPs. This modification improves our ability to accurately predict blocks of local phylogeny in both simulation and real genotype data.

I further analyzed confounding factors in computing compatible intervals. I describe how my model can be used to predict the absolute rate of error affecting my method's ability to accurately describe local phylogenetic structure. I also considered the attribution of these "errors" to either genotyping error or homoplasy, a violation of the infinite-sites assumption.

This method may be applied to other data sets with significant error. Notably, we could apply the same method to the structural variation data reported by the Sanger Institute as a part of the same sequencing effort [96]. This includes small insertions and deletions (indels), indicated by small nonlinearities in the read alignments, as well as large-scale structural variations discovered by a combination of post-alignment heuristics. Including these variants while compensating for reporting error or confounding biological factors may further improve our categorization of local phylogenetic structure in the population of classical laboratory mice.

The inclusion of additional datasets and different types of input data to my compatible intervals model may also improve the characterization of local phylogeny in regions with unusual variation, for example structural variants. There exist regions of structural variation

and regions with otherwise few detectable SNPs and recombinations, which - for this very reason - may include biologically significant findings that we wish to identify. My extension of the compatible intervals model to better handle errors enables the analysis of these types of features.

Chapter 6

Discussion and Conclusion

Local phylogenetic structure within species can provide unique insights into an organism's history and inheritance. Local phylogenetic structure can be used to increase the density of genotype data through imputation, improve the specificity or narrow target regions in GWAS, and to act as a scaffold for the discovery of secondary genotypic information including subspecific origin and gene expression. In this thesis, I introduce highly effective and generalizable methods for characterizing phylogeny structure as a mosaic of genomic regions showing no evidence of recombination.

I described in Chapter 2 a generic method for computing an unambiguous and maximally informative set of compatible intervals over the genome. A compatible interval is defined as a contiguous region of SNPs for which no pair violates the four-gamete rule, that is, they show no evidence of historical recombination under the infinite-sites model. I describe an interval cover with the minimum number of intervals required to cover the genome, while each interval is maximal in size. I show that this cover can be computed efficiently. Additionally, I describe extensions of this model which can be applied to unphased genotype data (Chapter 2) and a relaxation of four-gamete compatibility that provides more useful intervals over erroneous genotype data (Chapter 5).

In addition to these methods, I have contributed to the study of intra-species phylogeny

through the design and development of web-based visualization and analysis tools. These tools use the model of local phylogenetic structure I describe to portray the relationships between individuals in a holistic manner. These tools are being actively used by researchers and geneticists interested in comparative genomics between common laboratory strains, particularly aiding the identification of shared inherited haplotype structure, a crucial indicator of likely gene expression and phenotype. These tools have been and continue to be widely used (see Table 6.1). Thus far, my approaches have primarily applied to mouse populations, however, the tool is open source (see `https://github.com/txje/phylogeny-viewer`) and easily extensible to other species and populations.

6.1 Applications of Local Phylogeny

The model of local phylogenetic structure and compatible intervals I have developed has been and continues to be used in studies examining the variation among individuals within a species. The understanding of the relationships between local inherited haplotypes can be used to infer, among other things, regions sharing ancestral haplotypes, subspecific origin, and likely gene expression profiles.

The compatible intervals described in this thesis have been used to label the genomic regions of laboratory mice according to their subspecific origin [98]. This was accomplished by identifying a collection of informative SNPs ascertained from wild mice of known subspecies. The subspecific origin of laboratory mice has been a hotly debated subject [28, 99, 98], as it is an important factor in understanding the origins and sources of the genetic diversity in an important model organism. This work uncovered at least three important results relating to the use of the laboratory mouse as a platform for studying genetics. First, the vast majority of the laboratory mouse's genome is derived from a single subspecies. Second, the classical laboratory strains available today appear to be derived from a relatively

small population of ancestral mice, as evidenced by the limited number of distinct haplotypes seen at any place in the genome. This suggests that relatively few chromosomes (likely less than ten) can explain the overwhelming majority of the genetic diversity of the common laboratory mouse. Third, our studies suggest that some of the more recently developed "wild-derived" inbred strains are not representatives from a single subspecies. In fact, most exhibit evidence of introgression that may have occurred either in nature or in the lab. All of these theories and conclusions rely heavily on the understanding genomic structure as it relates to local phylogeny.

Genome-wide association studies have used the local phylogenies I derived to identify or narrow regions possibly containing causative mutations. Because compatible intervals have the property that they show no evidence or historical recombination, we can partition individuals into groups with shared ancestral haplotypes. These groups may be used to better differentiate genotypic variation across the genome and relate it to phenotypic variation, like that which is done in GWAS. In a recent study [9], my collaborators and I used the local phylogenetic structure of laboratory mice to better refine the region of the X chromosome likely containing *XCE*, the X-inactivation controlling element. Analysis of MDA genotypes and sequence data showed an enrichment of consistent SDPs (eight MDA SNPs, 120 Sanger SNPs and indels) in a 194 kb compatible interval spanning from ˜99.09 - 100.46 Mb. Within this candidate interval, all phenotyped strains cluster into haplotypes based on their *XCE* allele.

The compatible intervals and associated haplotype groups were used by Szatkiewicz et al. [81] to inform the identification of copy-number variations between laboratory mice. Shared haplotypes derived from the local phylogenetic trees where used to confirm the predicted copy-number variants and to predict these variants in other strains which are identical-by-descent (IBD) according to the phylogenies.

As a part of the Collaborative Cross (CC) Consortium [12], my analysis of local phylogenetic structure has been used to help categorize the structure of the emerging CC population [90, 2, 14, 27]. In addition to the visualization tools described in Section 6.2, we have extended our mosaic of subspecific origin and ancestral haplotype structure in the "founder" (progenitor) inbred laboratory mice to the CC population. These analyses and tools have helped determine the emerging population structure, including the relative representation of founders, haplotypes, and subspecies, and, thereby, the predictive power of phenotypic studies using this population.

Similar to GWAS, studies of relative gene expression in the laboratory mouse have benefited from the understanding of local phylogenetic structure. Haplotypes within compatible intervals are related to the relative gene expression between individuals, and can therefore be used to both predict and validate gene expression when it is derived from ancestral genotypic variance [79, 17]. We have developed a separate visualization tool allowing comparison of gene expression profiles based on local phylogenetic structure, particularly differential gene expression based on subspecific origin [GECCO, Gene Expression in the Collaborative Cross (`csbio.unc.edu/gecco`), unpublished].

6.1.1 Improving Imputation

As discussed in Chapter 3, compatible intervals can be used as a scaffold to very accurately impute missing genotypes where we can identify blocks of shared ancestral haplotypes between strains. While I show that this imputation can be done with unprecedented accuracy using relatively sparse genotypes from microarrays, our ability to improve the overall accuracy of these methods is limited by the quantity and quality of both dense source genotypes from which we can impute and our ability to accurately characterize the local phylogenetic structure. As discussed, we may trivially increase the number of genotypes we can impute

with high confidence by acquiring additional high-density source genotypes. The increasing pace and decreasing cost of high-throughput sequencing will provide an abundance of additional source genotypes we can use in our imputation efforts as time goes on.

In addition, we can better characterize the local phylogenetic structure - to determine those individuals with shared haplotypes - by incorporating additional data. The effectiveness of incorporating these data into our compatible intervals depends on the density and accuracy of these genotypes. While high-throughput sequence is much denser than microarray data, both technologies have non-ignorable error rates [67]. My extension of the compatible interval cover algorithm I describe in Chapter 5 helps allow us to incorporate these data into the model while minimizing the effect of error. So far, we ignore genomic variation that is not represented by SNPs. However, other sources of variation may be incorporated to help better describe inheritance relationships between individuals not well represented by SNPs. Small insertions and deletions (indels) and large-scale structural variants including deletions, duplications, and reorganizations may be incorporated into my model of local phylogenetic structure in a way similar to the way I used SNPs. Like SNPs, these variants represent specific strain distribution patterns (SDPs) which can be accounted for in my assessment of four-gamete compatibility. Including these extra sources of data may improve the accuracy of our imputation, particularly by refining breakpoints and identifying small-scale haplotype blocks which are not detected using our MDA genotypes.

A notable limitation of my imputation method is in the detection of de-novo mutations, having occurred since the divergence of the inbred strains or in the particular individual which was sequenced. Mutation is very rare and has had very little time, relatively, to occur between closely related individuals. However, de-novo mutations do occur. Since there is no representation of these alleles in the remainder of the population, they cannot be reasonably imputed, but must be discovered experimentally.

6.2 Extensibility of Visualization Tools

In Chapter 4 I describe a web-based tool built to allow comparative analysis and visualization of individuals within a population. This application is, at its core, an extensible framework for comparative visualization of any genomic data. While the instance discussed supports comparison of common strains of laboratory mice, the framework can be trivially extended to other data set or populations.

I have implemented another version of this framework to support preliminary analysis of the emerging CC lines [2]. This browser supports analysis of 458 mouse lines in the CC in various stages of inbreeding as well as the 8 inbred founder strains [14]. Visualized data sets include the assigned founder mosaic for each CC line, subspecific origin, haplotype diversity, and local phylogenetic trees. This resource is available at `http://csbio.unc.edu/ccv` and is being used extensively by the CC Consortium and others to analyze these data. Figure 6.1 shows a snapshot of this tool.

While this tool and the Mouse Phylogeny Viewer discussed in Chapter 4 have seen widespread use (Figure 6.1), additional implementations of the local phylogeny model and these visualization tools have the potential to contribute to the study of other genetic models and populations.

Resource	Unique Users	Accesses/Downloads
Mouse Phylogeny Viewer	8680	199553
Collaborative Cross Viewer	1847	55623
Imputed genotypes	144	197

Table 6.1: Usage of online tools and resources since their inception (1-3 years)

Understanding the phylogenetic structure of a population allows us to do things we could not before and leads to improvements in other commonly performed analyses. My set of maximal compatible intervals and my visualization and analysis tools based upon them have

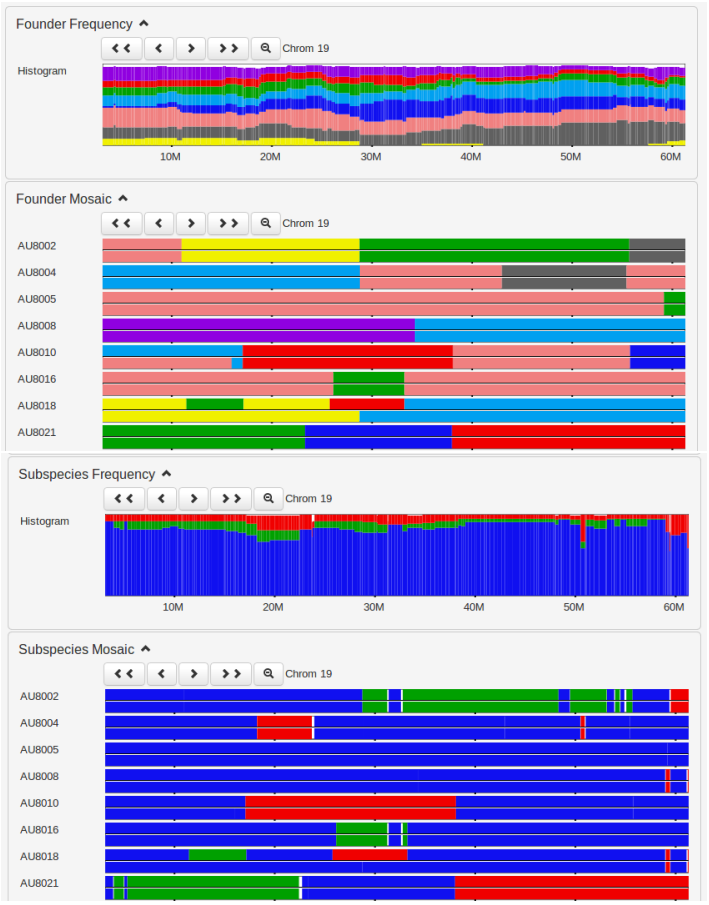


Figure 6.1: A sample of the data tracks available in the Collaborative Cross Viewer (`http://csbio.unc.edu/ccv`). The viewer supports analysis and visualization of 458 emerging CC lines and the 8 founder strains [10]. From top to bottom, the CC founder origin histogram and mosaic for a subset of lines is shown. The founder origins indicate the likely founder haplotypes inherited across the genome for the CC lines since the CC lines are a mosaic of the 8 founders. Note that many of these lines are not yet fully inbred, so there exist regions of heterozygosity as well as homozygosity. The subspecific origin histogram and mosaic is derived from the subspecific origin of the founders, as described for the Mouse Phylogeny Viewer.

and continue to go to great use in a wide variety of research, primarily those based on comparative analyses among the population of classical laboratory mice. These include analysis of subspecific origin in the lab mouse, a large collection of genome-wide association studies (GWAS), breeding and experimental design, and genome-wide imputation. There has been a variety of work others have done using, or informed by, the compatible interval method and results.

6.3 Conclusion

In my research, I have introduced methods to accurately and efficiently compute compatible intervals representing the local phylogenetic structure between individuals within a population. I provide code to compute compatible intervals in Appendix A. I demonstrate how these methods may be applied to assess multiple haplotypes and genotypes, and in the presence of erroneous data. I have shown how these intervals can be used to perform accurate genome-wide imputation. I also discuss how my model of local phylogeny has been used to inform a wide variety of intra-specific comparative studies including GWAS, QTL analysis, and identification of other genomic features like subspecific origin and copy-number variation. I have designed and implemented a web-based genome browser to visualize local phylogenetic structure and facilitate analyses on the basis of compatible intervals.

Appendix A

Compatible Intervals Code

Code to compute compatible intervals, in Python (version 2.6 - 2.7)

```
import sys

# --------------------------------------------------------------------
# Data structures
# --------------------------------------------------------------------

class SNP:
   def __init__(self, position, sdp_index, alleles):
      self.position = position
      self.sdp_index = sdp_index
      self.alleles = alleles

   def __repr__(self):
      return str(self.position)

   def __cmp__(self, other):
      return cmp(self.position, other.position)

class SDP:
   def __init__(self, bin, n, h, v, d, length):
      # each of bin, n, h, v, and d are bit vectors (integers) whose bits represent
          presence of the particular call
      # each vector stores the call for each strain on this SDP in the order they appear in
          the Chromosome.strains array from most to least significant bit
      # i.e. strains[0] call is vector >> length-1 (highest bit) and strains[length] call
          is vector & 1 or vector % 2 (lowest bit)
      self.bin = bin # 1 indicates minority allele (1)
      self.n = n # 1 indicates no call (N)
      self.h = h # 1 indicates heterozygous (H)
      self.v = v # 1 indicates VINO (V)
      self.d = d # 1 indicates deletion (D)
      # from process of elimination, a positions with 0 in all bit vectors is the majority
          allele (0)
      self.length = length

   def __repr__(self):
      return self.toarray()

   def __str__(self):
      return ''.join(self.toarray())

   def toarray(self):
      sdparray = []
      for i in xrange(self.length):
         if self.n >> self.length-1-i & 1 == 1:
            sdparray.append('N')
         elif self.h >> self.length-1-i & 1 == 1:
            sdparray.append('H')
         elif self.v >> self.length-1-i & 1 == 1:
            sdparray.append('V')
         elif self.d >> self.length-1-i & 1 == 1:
            sdparray.append('D')
```

```
            elif self.bin >> self.length-1-i & 1 == 1:
                sdparray.append('1')
            else:
                sdparray.append('0')
        return sdparray

    def __eq__(self, other):
        if self.toarray() == other.toarray():
            return True
        return False

    def singleton(self):
        j = 0
        while self.bin > 2**j:
            j += 1
        if self.bin == 2**j:
            return True
        return False

    def identical(self, other):
        if self.bin == other.bin and self.n == other.n and self.h == other.h and self.v ==
             other.v and self.d == other.d and self.length == other.length:
            return True
        return False

class Interval:
    def __init__(self, start_index, end_index):
        #the snps at both start and end indices are part of the interval
        self.start_index = start_index
        self.end_index = end_index

    def __repr__(self):
        return "(" + str(self.start_index) + "," + str(self.end_index) + ")"

    def __cmp__(self, other):
        return self.start_index - other.start_index;

    def size(self):
        return self.end_index - self.start_index + 1

class Chromosome:
    def __init__(self):
        self.chrName = ''
        self.strains = []
        self.SNPs = []
        self.SDPs = []

        self.left = []
        self.right = []
        self.cores = []
        self.uber = []
        self.maxk = []

    def sortSNPs(self):
        self.SNPs.sort()

    def lrScan(self):
        self.left = allScan(self, 0, len(self.SNPs)-1, uber=False)

    def rlScan(self):
        scan = allScan(self, len(self.SNPs)-1, 0, uber=False)
        reversed = []
        for i in xrange(len(scan) - 1, -1, -1):
            temp = scan[i].start_index
            scan[i].start_index = scan[i].end_index
            scan[i].end_index = temp
```

```
            reversed.append(scan[i])
        self.right = reversed

    def uberScan(self):
        self.uber = allScan(self, 0, len(self.SNPs)-1, uber=True)

    def coreScan(self):
        if len(self.left) == 0 or not len(self.left) == len(self.right):
            print "Can't core scan because left != right or left = []"
            print "left: " + str(len(self.left)) + ", right: " + str(len(self.right))
            return
        invs = []
        for i in xrange(len(self.left)):
            invs.append(Interval(self.left[i].start_index, self.right[i].end_index))
        self.cores = invs

    def maxkScan(self):
        if len(self.uber) == 0:
            self.uberScan()
        if len(self.cores) == 0:
            if len(self.left) == 0:
                self.lrScan()
            if len(self.right) == 0:
                self.rlScan()
            self.coreScan()
        self.maxk = maxkScan(self)

    def invFromType(self, type):
        if type == "left":
            return self.left
        if type == "right":
            return self.right
        if type == "cores":
            return self.cores
        if type == "uber":
            return self.uber
        if type == "maxk":
            return self.maxk

class Node:
    def __init__(self, strains, label="", description=""):
        self.strains = strains
        self.label = label
        self.description = description
        self.edges = []

class Edge:
    def __init__(self, direction, weight, end_node):
        self.direction = direction
        self.weight = weight
        self.sink = end_node

# Convert to binary tree with no labelled internal nodes
def printTree(t, strains):
    if len(t.edges) > 0:
        out = [printTree(e.sink, strains) + [e.weight] for e in t.edges]
        if len(t.strains) > 0:
            out.append(['|'.join(strains[s] for s in t.strains), 0]) # zero-weight edge
        # make bifurcating
        while(len(out) > 2): # sub1, sub2
            out = [[out[0], out[1], 0]] + out[2:]
        return out
    else:
        return ['|'.join(strains[s] for s in t.strains)]

# ------------------------------------------------------------------
```

```
# LR, RL, and Uber Scan computation
# --------------------------------------------------------------------

# allScan performs any of left->right, right->left (if start > end), and uber scan (if uber
    flag is set)
# 8/2010 - added sdp_hash to prevent checking of duplicate SDPs a la the time proof
def allScan(chrom, start, end, uber=False):
   SNPs = chrom.SNPs
   SDPs = chrom.SDPs
   inc = 1
   if start > end:
      inc = -1
   invs = []
   i = start
   invStart = start
   scan_back = False
   sdp_hash = {}
   while not i == end + inc:
      if scan_back:
      #scan backward
         while not invStart == start - inc:
            checkPos = i
            string = str(SDPs[SNPs[invStart].sdp_index])
            if not sdp_hash.has_key(string): # do not re-check duplicate SDPs, this gives
                us linear time scan
               while (not checkPos == invStart) and
                   compatible(SDPs[SNPs[invStart].sdp_index],
                   SDPs[SNPs[checkPos].sdp_index]):
                  checkPos -= inc
               if not checkPos == invStart:
                  break
               sdp_hash[string] = 1
            invStart -= inc
         invStart += inc
      if uber:
         scan_back = True
      #scan forward
      i += inc
      while not i == end + inc:
         checkPos = invStart
         string = str(SDPs[SNPs[i].sdp_index])
         if not sdp_hash.has_key(string): # do not re-check duplicate SDPs, this gives us
             linear time scan
            while (not checkPos == i) and compatible(SDPs[SNPs[i].sdp_index],
                SDPs[SNPs[checkPos].sdp_index]):
               checkPos += inc
            if not checkPos == i:
               invs.append(Interval(invStart, i - inc))
               if uber:
                  invStart = i - inc
               else:
                  invStart = i
               sdp_hash = {}
               break
            sdp_hash[string] = 1
         i += inc
   invs.append(Interval(invStart, end))
   return invs

# compatible tests 4-gamete compatibility of sdp1 and sdp2 with 2 alleles, n, h, d, and v
def compatible(sdp1, sdp2):
   l = sdp1.length
   mask = (sdp1.n^(2**l-1)) & (sdp2.n^(2**l-1)) & (sdp1.h^(2**l-1)) & (sdp2.h^(2**l-1)) &
       (sdp1.d^(2**l-1)) & (sdp2.d^(2**l-1)) & (sdp1.v^(2**l-1)) & (sdp2.v^(2**l-1))
   gametes = 0
   if sdp1.bin & sdp2.bin & mask > 0:
```

```
            gametes += 1
        if (sdp1.bin^(2**l-1)) & sdp2.bin & mask > 0:
            gametes += 1
        if sdp1.bin & (sdp2.bin^(2**l-1)) & mask > 0:
            gametes += 1
        if (sdp1.bin^(2**l-1)) & (sdp2.bin^(2**l-1)) & mask > 0:
            gametes += 1
        if gametes == 4:
            return False
        return True

# ------------------------------------------------------------------
# MaxK data structures and computation
# ------------------------------------------------------------------

class DPNode:
    def __init__(self, a, b):
        self.next = []
        self.a = a
        self.b = b
        self.back = None
        self.total = 0
        self.back_count = 0

    def AddNext(self, node):
        overlap = self.b + 1 - node.a
        if overlap >= 0:
            self.next.append(node)
            newTotal = self.total + overlap
            if newTotal == node.total:
                node.back = self
                self.back_count += 1
            if newTotal > node.total:
                node.total = newTotal
                node.back = self
                self.back_count = 1

class DPArray:
    def __init__(self):
        self.current = []
        self.last = None
        self.start = self.current

    def NextLevel(self):
        if len(self.current) > 0:
            self.last = self.current
            self.current = []

    def AddNode(self, a, b):
        newNode = DPNode(a, b)
        self.current.append(newNode)
        if self.last != None:
            for node in self.last:
                node.AddNext(newNode)

    def LastNode(self):
        max = -1
        lastNode = None
        for node in self.current:
            if node.total >= max:
                max = node.total
                lastNode = node

        return lastNode
```

```
def maxkScan(chrom):
   cores = chrom.cores
   uber = chrom.uber
   minDPArray = DPArray()
   last = 0
   for i in xrange(len(cores)):
      minDPArray.NextLevel()
      for j in xrange(last, len(uber)):
         if uber[j].start_index > cores[i].start_index:
            break
         elif (uber[j].end_index >= cores[i].end_index):
            minDPArray.AddNode(uber[j].start_index, uber[j].end_index)
      last = j
   invs = []
   node = minDPArray.LastNode()
   while node != None:
      invs.append(Interval(node.a, node.b))
      node = node.back
   invs.reverse()
   return invs

def parse(filename):
   data = [line.strip().split(',') for line in open(filename,
       'r').read().strip().split('\n')]
   strains = data[0][1:]

   sdps = {}
   sdp_count = 0
   snps = []
   length = len(strains)
   for d in data[1:]: # skip header
      position = int(d[0])
      snp = d[1:]
      #SNPList = subset(SNPList, indices)
      alleles = []
      bin = 0
      n = 0
      h = 0
      v = 0
      d = 0
      ns = 0
      for i in xrange(length):
         letter = snp[i].upper()
         if letter == '1':
            bin += 2**(length-1-i)
         elif letter == 'N' or letter == 'H' or letter == 'V' or letter == 'D':
            if letter == 'N':# or letter == 'H' or letter == 'D':
               n += 2**(length-1-i)
            if letter == 'H':
               h += 2**(length-1-i)
            if letter == 'V':
               v += 2**(length-1-i)
            if letter == 'D':
               d += 2**(length-1-i)
      sdpkey = bin*(2**(length*4)) + n*(2**(length*3)) + h*(2**(length*2)) +
          v*(2**(length)) + d
      sdp_index = -1
      if not sdps.has_key(sdpkey):
         sdps[sdpkey] = [sdp_count, 1]
         sdp_index = sdp_count
         sdp_count += 1
      else:
         sdp_index = sdps[sdpkey][0]
         sdps[sdpkey][1] += 1
      snps.append(SNP(position, sdp_index, alleles))
```

```
    sdps = [SDP(k>>length*4, (k>>length*3)%(2**length), (k>>length*2)%(2**length),
        (k>>length)%(2**length), k%(2**length), length) for k,v in sorted(sdps.iteritems(),
        key=lambda a:a[1][0])]
    chrom = Chromosome()
    chrom.SNPs = snps
    chrom.SDPs = sdps
    chrom.strains = strains
    print "SNPs: " + str(len(data)-1)
    return chrom

if __name__ == "__main__":
    print "Usage: python compatinv.py <input> <output> <interval type>"

    infile = sys.argv[1]
    output = sys.argv[2]
    inv_type = sys.argv[3]

    print "Reading file..."
    chrom = parse(infile)

    print "Processing..."
    chrom.maxkScan()
    invs = chrom.invFromType(inv_type)
```

Bibliography

[1] S. Altschul, W. Gish, W. Miller, E. Myers, and D. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215:403–410, 1990.

[2] D. Aylor, W. Valdar, W. Foulds-Mathes, R. Buus, R. Verdugo, R. Baric, M. Ferris, J. Frelinger, M. Heise, M. Frieman, L. Gralinski, T. Bell, J. Didion, K. Hua, D. Nehrenberg, C. Powell, J. Steigerwalt, Y. Xie, S. Kelada, F. Collins, I. Yang, D. Schwartz, L. Branstetter, E. Chesler, D. Miller, J. Spence, E. Liu, L. McMillan, A. Sarkar, J. Wang, W. Wang, Q. Zhang, K. Broman, R. Korstanje, C. Durrant, R. Mott, F. Iraqi, D. Pomp, D. Threadgill, F. Villena, and G. Churchill. Genetic analysis of complex traits in the emerging collaborative cross. *Genome Research*, 21:1213–1222, 2011.

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

[4] L. E. Baum and T. Petrie. Statistical inference for probabilistic functions of finite state markov chains. *Annals of Mathematical Statistics*, 37(6):1554–1563, 1966.

[5] J. A. Beck, S. Lloyd, M. L.-P. M. Hafezparast, J. T. Eppig, M. F. Festing, and E. M. Fisher. Genealogies of mouse inbred strains. *Nature Genetics*, 24:23–25, 2000.

[6] S. Besenbacher, T. Mailund, and M. Schierup. Local phylogeny mapping of quantitative traits: Higher accuracy and better ranking than single-marker association in genomewide scans. *Genetics*, 181:747–753, 2009.

[7] R. Brodie, R. Roper, and C. Upton. Jdotter: A java interface to multiple dot plots generated by dotter. *Bioinformatics*, 20(2):279–281, 2004.

[8] E. Buckler and M. Gore. An arabidopsis haplotype map takes root. *Nature Genetics*, 39(9):1056–1057, 2007.

[9] J. D. Calaway, A. B. Lenarcic, J. P. Didion, J. R. Wang, J. B. Searle, L. McMillan, W. Valdar, and F. P.-M. de Villena. High incidence of skewed x inactivation in laboratory mouse is a byproduct of the speciation process and domestication. *PLoS Genetics, under review*, 2013.

[10] R. H. Chung and D. Gusfield. Perfect phylogeny haplotyper: haplotype inferral using a tree model. *Bioinformatics Application Note*, 19(6), 2003.

[11] D. M. Church, L. Goodstadt, L. W. Hillier, M. C. Zody, S. Goldstein, X. She, C. J. Bult, R. Agarwala, J. L. Cherry, M. DiCuccio, W. Hlavina, Y. Kapustin, P. Meric, D. Maglott, Z. Birtle, A. C. Marques, T. Graves, S. Zhou, B. Teague, K. Potamousis, C. Churas, M. Place, J. Herschleb, R. Runnheim, D. Forrest, J. Amos-Landgraf, D. C.

Schwartz, Z. Cheng, K. Lindblad-Toh, E. E. Eichler, C. P. Ponting, and The Mouse Genome Sequencing Consortium. Lineage-specific biology revealed by a finished genome assembly of the mouse. *PLoS Biology*, 7, 2009.

[12] G. A. Churchill, D. C. Airey, H. Allayee, J. M. Angel, A. D. Attie, J. Beatty, W. D. Beavis, J. K. Belknap, B. Bennett, W. Berrettini, A. Bleich, M. Bogue, K. W. Broman, K. J. Buck, E. Buckler, M. Burmeister, E. J. Chesler, J. M. Cheverud, S. Clapcote, M. N. Cook, R. D. Cox, J. C. Crabbe, W. E. Crusio, A. Darvasi, C. F. Deschepper, R. W. Doerge, C. R. Farber, J. Forejt, D. Gaile, S. J. Garlow, H. Geiger0, H. Gershenfeld, T. Gordon, J. Gu, W. Gu, G. de Haan, N. L. Hayes, C. Heller, H. Himmelbauer, R. Hitzemann, K. Hunter, H.-C. Hsu, F. A. Iraqi, B. Ivandic, H. J. Jacob, R. C. Jansen, K. J. Jepsen, D. K. Johnson, T. E. Johnson, G. Kempermann, C. Kendziorski, M. Kotb, R. F. Kooy, B. Llamas, F. Lammert0, J.-M. Lassalle, P. R. Lowenstein, L. Lu, A. Lusis, K. F. Manly, R. Marcucio, D. Matthews, J. F. Medrano, D. R. Miller, G. Mittleman, B. A. Mock, J. S. Mogil, X. Montagutelli, G. Morahan, D. G. Morris, R. Mott, J. H. Nadeau, H. Nagase, R. S. Nowakowski, B. F. O'Hara, A. V. Osadchuk, G. P. Page, B. Paigen, K. Paigen, A. A. Palmer, H.-J. Pan, L. Peltonen-Palotie, J. Peirce, D. Pomp, M. Pravenec, D. R. Prows, Z. Qi, R. H. Reeves, J. Roder, G. D. Rosen, E. E. Schadt, L. C. Schalkwyk, Z. Seltzer, K. Shimomura, S. Shou, M. J. Sillanp, L. D. Siracusa, H.-W. Snoeck, J. L. Spearow, K. Svenson, L. M. Tarantino, D. Threadgill, L. A. Toth, W. Valdar, F. P.-M. de Villena, C. Warden, S. Whatley, R. W. Williams, T. Wiltshire, N. Yi, D. Zhang, M. Zhang, and F. Zou. The collaborative cross, a community resource for the genetic analysis of complex traits. *Nature Genetics*, 36:1133–1137, 2004.

[13] R. Clark, G. Schweikert, C. Toomajian, S. Ossowski, G. Zeller, P. Shinn, N. Warthmann, T. Hu, G. Fu, D. Hinds, H. Chen, K. Frazer, D. Huson, B. Scholkopf, M. Nordborg, G. Ratsch, J. Ecker, and D. Weigel. Common sequence polymorphisms shaping genetic diversity in arabidopsis thaliana. *Science*, 317:338–342, 2007.

[14] Collaborative Cross Consortium. The genome architecture of the collaborative cross mouse genetic reference population. *Genetics*, 190(2):389–401, 2012.

[15] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms.* MIT Press, 2004.

[16] O. Couronne, A. Poliakov, N. Bray, T. Ishkhanov, D. Ryaboy, E. Rubin, L. Pachter, and I. Dubchak. Strategies and tools for whole-genome alignments. *Genome Research*, 13:73–80.

[17] J. J. Crowley, V. Zhabotynsky, W. Sun, S. Huang, Y. Kim, J. D. Calaway, J. R. Wang, I. K. Pakatci, D. R. Miller, P. F. Sullivan, F. Zou, L. McMillan, and F. P.-M. de Villena. Pervasive allelic imbalance revealed by allele-specific gene expression in highly divergent mouse crosses. *Nature Genetics, under review*, 2013.

[18] E. Cuppen. Haplotype-based genetics in mice and rats. *TRENDS in Genetics*, 21(6), 2005.

[19] M. Daly, J. Rioux, S. Schaffner, T. Hudson, and E. Lander. High-resolution haplotype structure in the human genome. *Nature Genetics*, 29:229–232, 2001.

[20] J. Didion, H. Yang, K. Sheppard, C. Fu, L. McMillan, F. Villena, and G. Churchill. Discovery of novel variants in genotyping arrays significantly improves genotype retention and corrects ascertainment bias. *BMC Genomics*, 13(34), 2012.

[21] L. Ding, M. J. Ellis, S. Li, D. E. Larson, K. Chen, J. W. Wallis, C. C. Harris, M. D. McLellan, R. S. Fulton, L. L. Fulton, R. M. Abbott, J. Hoog, D. J. Dooling, D. C. Koboldt, H. Schmidt, J. Kalicki, Q. Zhang, L. Chen, L. Lin, M. C. Wendl, J. F. McMichael, V. J. Magrini, L. Cook, S. D. McGrath, T. L. Vickery, E. Appelbaum, K. DeSchryver, S. Davies, T. Guintoli, L. Lin, R. Crowder, Y. Tao, J. E. Snider, S. M. Smith, A. F. Dukes, G. E. Sanderson, C. S. Pohl, K. D. Delehaunty, C. C. Fronick, K. A. Pape, J. S. Reed, J. S. Robinson, J. S. Hodges, W. Schierding, N. D. Dees, D. Shen, D. P. Locke, M. E. Wiechert, J. M. Eldred, J. B. Peck, B. J. Oberkfell, J. T. Lolofie, F. Du, A. E. Hawkins, M. D. O'Laughlin, K. E. Bernard, M. Cunningham, G. Elliott, M. D. Mason, D. M. Thompson, Jr., J. L. Ivanovich, P. J. Goodfellow, C. M. Perou, G. M. Weinstock, R. Aft, M. Watson, T. J. Ley, R. K. Wilson, , and E. R. Mardis. Genome remodelling in a basal-like breast cancer metastasis and xenograft. *Nature*, 464:999–1005, 2010.

[22] Z. Ding, V. Filkov, and D. Gusfield. A linear-time algorithm for the perfect phylogeny haplotyping (PPH) problem. *RECOMB*, 2005.

[23] Z. Ding, T. Mailund, and Y. Song. Efficient whole-genome association mapping using local phylogenies for unphased genotype data. *Bioinformatics*, 24(19):2215–2221, 2008.

[24] S. Dombrowski and D. Maglott. Using the map viewer to explore genomes. *The NCBI Handbook*, 2002.

[25] E. Eskin, E. Halperin, and R. Karp. Large scale reconstruction of haplotypes from genotype data. *RECOMB*, 2003.

[26] E. Eskin, E. Halperin, and R. M. Karp. Efficient reconstruction of haplotype structure via perfect phylogeny. *Journal of bioinformatics, computational biology.*, 2003.

[27] M. T. Ferris, D. L. Aylor, D. Bottomly, A. C. Whitmore, L. D. Aicher, T. A. Bell, B. Bradel-Tretheway, J. T. Bryan, R. J. Buus, L. E. Gralinski, B. L. Haagmans, L. McMillan, D. R. Miller, E. Rosenzweig, W. Valdar, J. Wang, G. A. Churchill, D. W. Threadgill, S. K. McWeeney, M. G. Katze, F. P.-M. de Villena, R. S. Baric1, and M. T. Heise. Modeling host genetic regulation of influenza pathogenesis in the collaborative cross. *PLoS Pathogens*, 9(2), 2013.

[28] K. Frazer, E. Eskin, H. Kang, M. Bogue, D. Hinds, E. Beilharz, R. Gupta, J. Montgomery, M. Morenzoni, G. Nilsen, C. Pethiyagoda, L. Stuve, F. Johnson, M. Daly,

C. Wade, and D. Cox. A sequence-based variation map of 8.27 million SNPs in inbred mouse strains. *Nature*, 2007.

[29] K. Frazer, L. Pachter, A. Poliakov, E. Rubin, and I. Dubchak. Vista: computational tools for comparative genomics. *Nucleic Acids Research*, 32:273–9, 2004.

[30] S. Gabriel, S. Schaffner, H. Nguyen, J. Moore, J. Roy, B. Blumenstiel, J. Higgins, M. DeFelice, A. Lochner, M. Faggart, S. Liu-Cordero, C. Rotimi, A. Adeyemo, R. Cooper, R. Ward, E. Lander, M. Daly, and D. Altshuler. The structure of haplotype blocks in the human genome. *Science*, 296:2225–2229, 2002.

[31] J. Gramm, T. Hartman, T. Nierhoff, R. Sharan, and T. Tantau. On the complexity of SNP block partitioning under the perfect phylogeny model. *Discrete Mathematics*, 4175:92–102, 2008.

[32] V. Guryev, B. Smits, J. van de Belt, M. Verheul, N. Hubner, and E. Cuppen. Haplotype block structure is conserved across mammals. *PLoS Genetics*, 2(7):e121, July 2006.

[33] D. Gusfield. Efficient algorithms for inferring evolutionary trees. *Networks*, 21:19–28, 1991.

[34] D. Gusfield. Haplotyping as perfect phylogeny: Conceptual framework and efficient solutions. *RECOMB*, 2002.

[35] D. Gusfield. The multi-state perfect phylogeny problem with missing and removable data: solutions via integer-programming and chordal graph theory. *Journal of Computational Biology*, 17(3):383–399, 2010.

[36] R. Gysel and D. Gusfield. Extensions and improvements to the chordal graph approach to the multi-state perfect phylogeny problem. *Bioinformatics Research, Applications*, 6053:52–60, 2010.

[37] E. Halperin and R. Karp. Perfect phylogeny and haplotype assignment. *Proceedings of RECOMB*, pages 10–19, 2004.

[38] F. Hsu, W. Kent, H. Clawson, R. Kuhn, M. Diekhans, and D. Haussler. The UCSC known genes. *Bioinformatics*, 22(9):1036–1046, 2006.

[39] T. Hubbard, D. Barker, E. Birney, G. Cameron, Y. Chen, L. Clark, T. Cox, J. Cuff, V. Curwen, T. Down, R. Durbin, E. Eyras, J. Gilbert, M. Hammond, L. Huminiecki, A. Kasprzyk, H. Lehvaslaiho, P. Lijnzaad, C. Melsopp, E. Mongin, R. Pettett, M. Pocock, S. Potter, A. Rust, E. Schmidt, S. Searle, G. Slater, J. Smith, W. Spooner, A. Stabenau, J. Stalker, E. Stupka, A. Ureta-Vidal, I. Vastrik, and M. Clamp. The ensembl genome database project. *Nucleic Acids Research*, 30(1):38–41, 2002.

[40] R. Hudson and N. Kaplan. Statistical properties of the number of recombination events in the history of a sample of dna sequences. *Genetics*, 111:147–164, 1985.

[41] H. M. Kang, N. A. Zaitlen, and E. Eskin. Eminim: An adaptive and memory-efficient algorithm for genotype imputation. *Journal of Computational Biology*, 17:547–560, 2010.

[42] R. M. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computation*, pages 85–103, 1972.

[43] T. M. Keane, L. Goodstadt, P. Danecek, M. A. White, K. Wong, B. Yalcin, A. Heger, A. Agam, G. Slater, M. Goodson, N. A. Furlotte, E. Eskin, C. Nellker, H. Whitley, J. Cleak, D. Janowitz, P. Hernandez-Pliego, A. Edwards, T. G. Belgard, P. L. Oliver, R. E. McIntyre, A. Bhomra, J. Nicod, X. Gan, W. Yuan, L. van der Weyden, C. A. Steward, S. Balasubramaniam, J. Stalker, R. Mott, R. Durbin, I. J. Jackson, A. Czechanski, J. A. G. Assuno, L. R. Donahue, L. G. Reinholdt, B. A. Payseur, C. P. Ponting, E. Birney, J. Flint, and D. J. Adams. Mouse genomic variation and its effect on phenotypes and gene regulation. *Nature*, 477:289–294, 2011.

[44] J. Kececioglu and D. Gusfield. Reconstructing a history of recombinations from a set of sequences. *Proceedings of SODA*, pages 471–480, 1998.

[45] W. Kent, C. Sugnet, T. Furey, K. Roskin, T. Pringle, A. Zahler, and D. Haussler. The human genome browser at UCSC. *Genome Research*, 12:966–1006, 2002.

[46] S. Kim, V. Plagnol, T. Hu, C. Toomajian, R. Clark, S. Ossowski, J. Ecker, D. Weigel, and M. Nordborg. Recombination and linkage disequilibrium in arabidopsis thaliana. *Nature Genetics*, 39(9):1151–1155, 2007.

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

[48] M. Kimura. The number of heterozygous nucleotide sites maintained in a finite population due to steady flux of mutations. *Genetics*, 61:893903, 1969.

[49] M. Kimura. A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. *Journal of Molecular Evolution*, 16(2):111–120, 1980.

[50] A. Kirby, H. M. Kang, C. M. Wade, C. Cotsapas, E. Kostem, B. Han, N. Furlotte, E. Y. Kang, M. Rivas, M. A. Bogue, K. A. Frazer, F. M. Johnson, E. J. Beilharz, D. R. Cox, E. Eskin, and M. J. Daly. Fine mapping in 94 inbred mouse strains using a high-density haplotype resource. *Genetics*, 185:1081–1095, 2010.

[51] A. T. Kraja, H. A. Lawson, D. K. Arnett, I. B. Borecki, U. Broeckel, L. de las Fuentes, S. C. Hunt, M. A. Province, J. Cheverud, and D. Rao. Obesity-insulin targeted genes in the 3p26-25 region in human studies and LG/J and SM/J mice. *Metabolism*, 61(8):1129–1141, 2012.

[52] M. Krzywinski, J. Schein, nan Birol, J. Connors, R. Gascoyne, D. Horsman, S. J. Jones, and M. A. Marra. Circos: an information aesthetic for comparative genomics. *Genome Research*, 19:1639–1645, 2009.

[53] S. Kumar and S. Subramanian. Mutation rates in mammalian genomes. *PNAS*, 99(2):803808.

[54] H. Li, B. Handsaker, A. Wysoker, T. Fennell, J. Ruan, N. Homer, G. Marth, G. Abecasis, and R. Durbin. The sequence alignment/map format and SAMtools. *Bioinformatics*, 25:2078–2079, 2009.

[55] H. Li, J. Ruan, and R. Durbin. Mapping short dna sequencing reads and calling variants using mapping quality scores. *Genome Research*, 18:1851–1858, 2008.

[56] B. L. Lundrigan, S. A. Jansa, and P. K. Tucker. Phylogenetic relationships in the genus mus, based on paternally, maternally, and biparentally inherited characters. *Systems Biology*, 51(3):410431, 2002.

[57] T. Mailund, S. Besenbacher, and M. Schierup. Whole genome association mapping by incompatibilities and local perfect phylogenies. *BMC Bioinformatics*, 7:254, 2006.

[58] E. R. Mardis, L. Ding, D. J. Dooling, D. E. Larson, M. D. McLellan, K. Chen, D. C. Koboldt, R. S. Fulton, K. D. Delehaunty, S. D. McGrath, L. A. Fulton, D. P. Locke, V. J. Magrini, R. M. Abbott, T. L. Vickery, J. S. Reed, J. S. Robinson, T. Wylie, S. M. Smith, L. Carmichael, J. M. Eldred, C. C. Harris, J. Walker, J. B. Peck, F. Du, A. F. Dukes, G. E. Sanderson, A. M. Brummett, E. Clark, J. F. McMichael, R. J. Meyer, J. K. Schindler, C. S. Pohl, J. W. Wallis, X. Shi, L. Lin, H. Schmidt, Y. Tang, C. Haipek, M. E. Wiechert, J. V. Ivy, J. Kalicki, G. Elliott, R. E. Ries, J. E. Payton, P. Westervelt, M. H. Tomasson, M. A. Watson, J. Baty, S. Heath, W. D. Shannon, R. Nagarajan, D. C. Link, M. J. Walter, T. A. Graubert, J. F. DiPersio, R. K. Wilson, and T. J. Ley. Recurring mutations found by sequencing an acute myeloid leukemia genome. *New England Journal of Medicine*, 361:1058–1066, 2009.

[59] P. McClurg, M. Pletcher, T. Wiltshire, and A. Su. Comparative analysis of haplotype association mapping algorithms. *BMC Bioinformatics*, 7(1):61–76, 2006.

[60] M. D. McMullen, S. Kresovich, H. S. Villeda, P. Bradbury, H. Li, Q. Sun, S. Flint-Garcia, J. Thornsberry, C. Acharya, C. Bottoms, P. Brown, C. Browne, M. Eller, K. Guill, C. Harjes, D. Kroon, N. Lepak, S. E. Mitchell, B. Peterson, G. Pressoir, S. Romero, M. O. Rosas, S. Salvo, H. Yates, M. Hanson, E. Jones, S. Smith, J. C. Glaubitz, M. Goodman, D. Ware, J. B. Holland, and E. S. Buckler. Genetic properties of the maize nested association mapping population. *Science*, 325(5941):737–740, 2009.

[61] R. Mott and J. Flint. Simultaneous detection and fine mapping of quantitative trait loci in mice using heterogeneous stocks. *Genetics*, 160:1609–1618, 2002.

[62] Mouse Genome Sequencing Consortium, R. Waterston, K. Lindblad-Toh, E. Birney, J. Rogers, J. Abril, P. Agarwal, R. Agarwala, R. Ainscough, M. Alexandersson, P. An, S. Antonarakis, J. Attwood, R. Baertsch, J. Bailey, K. Barlow, S. Beck, E. Berry, B. Birren, T. Bloom, P. Bork, M. Botcherby, N. Bray, M. Brent, D. Brown, S. Brown, C. Bult, J. Burton, J. Butler, R. Campbell, P. Carninci, S. Cawley, F. Chiaromonte, A. Chinwalla, D. Church, M. Clamp, C. Clee, F. Collins, L. Cook, R. Copley, A. Coulson, O. Couronne, J. Cuff, V. Curwen, T. Cutts, M. Daly, R. David, J. Davies, K. Delehaunty, J. Deri, E. Dermitzakis, C. Dewey, N. Dickens, M. Diekhans, S. Dodge, I. Dubchak, D. Dunn, S. Eddy, L. Elnitski, R. Emes, P. Eswara, E. Eyras, A. Felsenfeld, G. Fewell, P. Flicek, K. Foley, W. Frankel, L. Fulton, R. Fulton, T. Furey, D. Gage, R. Gibbs, G. Glusman, S. Gnerre, N. Goldman, L. Goodstadt, D. Grafham, T. Graves, E. Green, S. Gregory, R. Guig, M. Guyer, R. Hardison, D. Haussler, Y. Hayashizaki, L. Hillier, A. Hinrichs, W. Hlavina, T. Holzer, F. Hsu, A. Hua, T. Hubbard, A. Hunt, I. Jackson, D. Jaffe, L. Johnson, M. Jones, T. Jones, A. Joy, M. Kamal, E. Karlsson, D. Karolchik, A. Kasprzyk, J. Kawai, E. Keibler, C. Kells, W. Kent, A. Kirby, D. Kolbe, I. Korf, R. Kucherlapati, E. Kulbokas, D. Kulp, T. Landers, J. Leger, S. Leonard, I. Letunic, R. Levine, J. Li, M. Li, C. Lloyd, S. Lucas, B. Ma, D. Maglott, E. Mardis, L. Matthews, E. Mauceli, J. Mayer, M. McCarthy, W. McCombie, S. McLaren, K. McLay, J. McPherson, J. Meldrim, B. Meredith, J. Mesirov, W. Miller, T. Miner, E. Mongin, K. Montgomery, M. Morgan, R. Mott, J. Mullikin, D. Muzny, W. Nash, J. Nelson, M. Nhan, R. Nicol, Z. Ning, C. Nusbaum, M. O'Connor, Y. Okazaki, K. Oliver, E. Overton-Larty, L. Pachter, G. Parra, K. Pepin, J. Peterson, P. Pevzner, R. Plumb, C. Pohl, A. Poliakov, T. Ponce, C. Ponting, S. Potter, M. Quail, A. Reymond, B. Roe, K. Roskin, E. Rubin, A. Rust, R. Santos, V. Sapojnikov, B. Schultz, J. Schultz, M. Schwartz, S. Schwartz, C. Scott, S. Seaman, S. Searle, T. Sharpe, A. Sheridan, R. Shownkeen, S. Sims, J. Singer, G. Slater, A. Smit, D. Smith, B. Spencer, A. Stabenau, N. Stange-Thomann, C. Sugnet, M. Suyama, G. Tesler, J. Thompson, D. Torrents, E. Trevaskis, J. Tromp, C. Ucla, A. Ureta-Vidal, J. Vinson, A. Von Niederhausern, C. Wade, M. Wall, R. Weber, R. Weiss, M. Wendl, A. West, K. Wetterstrand, R. Wheeler, S. Whelan, J. Wierzbowski, D. Willey, S. Williams, R. Wilson, E. Winter, K. Worley, D. Wyman, S. Yang, S. Yang, E. Zdobnov, M. Zody, and E. Lander. Initial sequencing and comparative analysis of the mouse genome. *Nature*, 420:520–562, 2002.

[63] F. Pan, L. McMillan, F. Pardo-Manuel de Villena, D. Threadgill, and W. Wang. Treeqa: Quantitative genome wide association mapping using local perfect phylogeny trees. *PSB*, January 2009.

[64] P. Paschou, M. Mahoney, A. Javed, J. Kidd, A. Pakstis, S. Gu, K. Kidd, and P. Drineas. Intra- and interpopulation genotype reconstruction from tagging SNPs. *Genome Research*, 17(1):96–107, 2006.

[65] N. Patil, A. Berno, D. Hinds, W. Barrett, J. Doshi, C. Hacker, C. Kautzer, D. Lee, C. Marjoribanks, D. McDonough, B. Nguyen, M. Norris, J. Sheehan, N. Shen,

D. Stern, R. Stokowski, D. Thomas, M. Trulson, K. Vyas, K. Frazer, S. Fodor, and D. Cox. Blocks of limited haplotype diversity revealed by high-resolution scanning of human chromosome 21. *Science*, 294(23):1719–1723, 2001.

[66] P. M. Petkov, Y. Ding, M. A. Cassell, W. Zhang, G. Wagner, E. E. Sargent, S. Asquith, V. Crew, K. A. Johnson, P. Robinson, V. E. Scott, and M. V. Wiles. An efficient SNP system for mouse genome scanning and elucidating strain relationships. *Genome Research*, 14:1806–1811, 2004.

[67] F. Pompanon, A. Bonin, E. Bellemain, and P. Taberlet. Genotyping errors: Causes, consequences and solutions. *Nature Reviews*, 6, 2005.

[68] D. Reich, M. Cargill, S. Bolk, J. Ireland, P. Sabeti, D. Richter, T. Lavery, R. Kouyoumjian, S. Farhadlan, R. Ward, and E. Lander. Linkage disequilibrium in the human genome. *Nature*, 411:199–204, 2001.

[69] N. Saitou and M. Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, 4(4):406–425, 1987.

[70] R. Schwartz, B. Halldorsson, V. Bafna, A. Clark, and S. Istrail. Robustness of inference of haplotype block structure. *Journal of Computational Biology*, 10:13–19, 2003.

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

[72] S. T. Sherry, M. H. Ward, M. Kholodov, J. Baker, L. Phan, E. M. Smigielski, and K. Sirotkin. dbSNP: the NCBI database of genetic variation. *Nucleic Acids Research*, 29:308–311, 2001.

[73] S. Shifman, J. Bell, R. Copley, M. Taylor, R. Williams, R. Mott, and J. Flint. A high-resolution single nucleotide polymorphism genetic map of the mouse genome. *PLoS Biology*, 4(12):e395, 2006.

[74] M. Skinner, A. Uzilov, L. Stein, C. Mungall, and I. Holmes. JBrowse: A next-generation genome browser. *Genome Research*, 19:1630–1638, 2009.

[75] Y. Song, Z. Ding, D. Gusfield, C. Langley, and Y. Wu. Algorithms to distinguish the role of gene-conversion from single-crossover recombination in the derivation of SNP sequences in populations. *Journal of Computational Biology*, 14(10):1273–1286, 2007.

[76] Y. Song, Y. Wu, and D. Gusfield. Efficient computation of close lower and upper bounds on the minimum number of recombinations in biological sequence evolution. *Bioinformatics*, 21:413–422, 2005.

[77] S. Sridhar, F. Lam, G. Blelloch, R. Ravi, and R. Schwartz. Efficiently finding the most parsimonious phylogenetic tree via linear programming. *LNCS:Proceedings of ISBRA*, 4463:37–48, 2007.

[78] L. Stein, C. Mungall, S. Shu, M. Caudy, M. Mangone, A. Day, E. Nickerson, J. Stajich, T. Harris, A. Arva, and S. Lewis. The generic genome browser: A building block for a model organism system database. *Genome Research*, 12:1599–1610, 2002.

[79] W. Sun, S. Lee, V. Zhabotynsky, F. Zou, F. A. Wright, J. J. Crowley, Z. Yun, R. J. Buus, D. R. Miller, J. Wang, L. McMillan, F. P.-M. de Villena, and P. F. Sullivan. Transcriptome atlases of mouse brain reveals differential expression across brain regions and genetic backgrounds. *G3*, 2:203–211.

[80] J. P. Szatkiewicz, G. L. Beane, Y. Ding, L. Hutchins, F. P.-M. de Villena, and G. A. Churchill. An imputed genotype resource for the laboratory mouse. *Mammalian Genome*, 19:199–208, 2008.

[81] J. P. Szatkiewicz, W. Wang, P. F. Sullivan, W. Wang, and W. Sun. Improving detection of copy-number variation by simultaneous bias correction and read-depth segmentation. *Nucleic Acis Research*, 41(3):1519–1532, 2013.

[82] A. Templeton, T. Maxwell, D. Posada, J. Stengard, E. Boerwinkle, and C. Sing. Tree scanning: A method for using haplotype trees in phenotype/genotype association studies. *Genetics*, 169:441–453, 2005.

[83] The International Hapmap Consortium. A haplotype map of the human genome. *Nature*, 437:1299–1320, 2005.

[84] W3C. XMLHttpRequest - W3C working draft 6 December 2012. `http://www.w3.org/TR/XMLHttpRequest/`, 2012. [Online; accessed 5-April-2013].

[85] J. Wang, K. Moore, Q. Zhang, F. Villena, W. Wang, and L. McMillan. Genome-wide compatible SNP intervals and their properties. *Proceedings of ACM International Conference on Bioinformatics and Computational Biology*, 2010.

[86] J. Wang, F. Villena, H. Lawson, J. Cheverud, G. Churchill, and L. McMillan. Imputation of SNPs in inbred mice using local phylogeny. *Genetics*, 190(2):449–458, 2012.

[87] J. R. Wang, F. P.-M. de Villena, and L. McMillan. Dynamic visualization and comparative analysis of multiple collinear genomic data. *Proceedings of ACM International Conference on Bioinformatics and Computational Biology*, 2011.

[88] J. R. Wang, F. P.-M. de Villena, and L. McMillan. Comparative analysis and visualization of multiple collinear genomes. *BMC Bioinformatics*, 13(Supplement 3), 2012.

[89] N. Wang, J. Akey, K. Zhang, R. Chakraborty, and L. Jin. Distribution of recombination of crossovers and the origin of haplotype blocks: The interplay of population history, recombination, and mutation. *American Journal of Human Genetics*, 71:1227–1237, 2002.

[90] C. E. Welsh, D. R. Miller, K. F. Manly, J. Wang, L. McMillan, G. Morahan, R. Mott, F. A. Iraqi, D. W. Threadgill, and F. P.-M. de Villena. Status and access to the collaborative cross population. *Genetics*, 23:706–712, 2012.

[91] C. Wiuf. Inference on recombination and block structure using unphased data. *Genetics*, 166:537–545, 2004.

[92] A. Woerner, M. Cox, and M. Hammer. Recombination-filtered genomic datasets by information maximization. *Bioinformatics*, 23(14):1851–1853, 2007.

[93] Y. I. Wolf, I. B. Rogozin, N. V. Grishin, and E. V. Koonin. Genome trees and the tree of life. *TRENDS in Genetics*, 18(9), September 2002.

[94] Y. Wu and D. Gusfield. Improved algorithms for inferring the minimum mosaic of a set of recombinations. *LNCS:Proceedings of Combinatorial Pattern Matching*, 4580:150–161, 2007.

[95] X. Xu and J. Ma. An efficient simulated annealing algorithm for the minimum vertex cover problem. *Neurocomputing*, 69(7-9):913 – 916, 2006.

[96] B. Yalcin, K. Wong, A. Agam, M. Goodson, T. M. Keane, X. Gan, C. Nellker, L. Goodstadt, J. Nicod, A. Bhomra, P. Hernandez-Pliego, H. Whitley, J. Cleak, R. Dutton, D. Janowitz, R. Mott, D. J. Adams, and J. Flint. Sequence-based characterization of structural variation in the mouse genome. *Nature*, 477:326–329, 2011.

[97] H. Yang, Y. Ding, L. Hutchins, J. Szatkiewicz, T. Bell, B. Paigen, J. Graber, F. Villena, and G. Churchill. A customized and versatile high-density genotyping array for the mouse. *Nature Methods*, 6:663–666, 2009.

[98] H. Yang, J. Wang, J. Didion, R. Buus, T. Bell, C. Welsh, F. Bonhomme, A. Yu, M. Nachman, J. Pialek, P. Tucker, P. Boursot, L. McMillan, G. Churchill, and F. Villena. Subspecific origin and haplotype diversity in the laboratory mouse. *Nature Genetics*, 43:648–655, 2011.

[99] Y. Yang, T. Bell, G. Churchill, and F. Pardo-Manuel de Villena. On the subspecific origin of the laboratory mouse. *Nature Genetics*, 39(9):1100–1107, 2007.

[100] K. Zhang, M. Deng, P. Calabrese, M. Nordborg, and F. Sun. Haplotype block structure and its applications to association studies: Power and study designs. *American Journal of Human Genetics*, 71:1386–1394, 2002.

[101] K. Zhang, M. Deng, T. Chen, M. Waterman, and F. Sun. A dynamic programming algorithm for haplotype block partitioning. *PNAS*, 99(11):7335–7339, 2002.

[102] K. Zhang, Z. S. Qin, J. S. Liu, T. Chen, M. S. Waterman, and F. Sun. Haplotype block partitioning and tag SNP selection using genotype data and their applications to association studies. *Genome Research*, 14(5):908–916, 2004.

[103] Q. Zhang, W. Wang, L. McMillan, F. Pardo-Manuel de Villena, and D. Threadgill. Inferring genome-wide mosaic structure. *PSB*, January 2009.

[104] Q. Zhang, W. Wang, L. McMillan, J. Prins, F. Pardo-Manuel de Villena, and D. Threadgill. Genotype sequence segmentation: Handling constraints and noise. *Proceedings of 8th Workshop on Algorithms in Bioinformatics*, 2008.