Chapter 2

ARG Inference

2.1 Discrete ARG Methods

In this chapter I describe a novel method for explicitly reconstructing ARGs from population genotype data. There are broadly two approaches to ARG based inference from population genetic data: methods based on the coalescent-with-recombination, and methods based on a discrete representation of the ARG. Discrete methods define the ARG as the genealogical topology that relates a set of sequences, and do not necessarily define a statistical model for those topologies. The method described here falls into that category.

The majority of work from the discrete perspective relates to finding the minimum number of recombination events required to derive a sample of sequences (Hudson and Kaplan, 1985; Myers and Griffiths, 2003; Wiuf, 2004; Song and Hein, 2004, 2005; Lyngsø et al., 2005; Bafna and Bansal, 2006). It is impossible to find the true number of recombination events as many recombinations are silent and leave no trace in the sample, but the minimum can be provably found when sufficient, and generally prohibitive (Wang et al., 2001), computational effort is applied. It is useful to know the minimum number of obligate recombination events, and their positions, when, say, exploring whether there has been mitochondrial recombination (Zsurka et al., 2005). The motivation often given in papers on this topic is towards detecting recombination hotspots; in Fearnhead et al. (2004) the obligate recombination inference method of Myers and Griffiths (2003) is shown to give results consistent with the presence of hotspots, as inferred by a likelihood method and detected by sperm typing.

In Hudson and Kaplan (1985); Myers and Griffiths (2003); Wiuf (2004); Song and Hein (2004); Bafna and Bansal (2006) and Gusfield et al. (2007) lower bounds are developed for the number of recombination events required in the genealogy of a sample of sequences.

Hudson's method (Hudson and Kaplan, 1985) is based around what is known as the four gamete test. The method infers a recombination event between a pair of SNPs when all four possible gametic types are present (00, 01, 10 and 11) in the population. Under the infinite sites model, a recombination must have occurred; there is no tree that can describe such a configuration of haplotypes. The method tests all pair of SNPs in the region to construct a collection of intervals, within each of which there is an obligate recombination event. Under the conservative assumption that overlapping intervals correspond to the same recombination event, the algorithm finds the largest subset of non-overlapping intervals from the collection, and the lower bound is given as the number of intervals.

The four gamete test becomes less accurate as the recombination rate and sample size increase. The bound can be improved by considering haplotypes rather than pairs of SNPs, and by moving towards a genealogical framework, as in Myers and Griffiths (2003), where two methods are presented. The first counts the number of haplotypes within a local window and relates this to the number of recombination events, and then combines local estimates via dynamical programming in order to achieve a lower bound for the whole region. The second method derives local estimates by taking the sequences and performing coalescences and mutations, until no such further operations are possible. A sequence is then removed from the sample, and this corresponds to at least one recombination event. The algorithm proceeds in this way until only one sequence remains. The algorithm performs a search over these "histories". Again, local estimates are combined into a lower bound for the whole region by dynamical programming.

A more sophisticated and computationally intensive approach is to attempt explicit construction of minimal ARGs (Song and Hein, 2005; Lyngsø et al., 2005).

Song and Hein (2005) finds a sequence of trees, with a tree for each marker, and the recombination events required to shift the tree at one marker into the tree at the next.

To do this, all possible trees are constructed for each marker. Then, the recombination events required to move from every tree at one marker, to every tree at the next marker are computed. The minimum number of recombination events, and the minimal ARG(s) can then be constructed by following the path through the markers that minimises the number of recombinations. This method can handle at most nine sequences with current computing power.

Lyngsø et al. (2005) takes a branch and bound approach and offers a significant improvement in speed and memory requirements, although is still limited to tens of sequences. Rather than working left-to-right along the sequence, this method works backwards in time, applying mutation, coalescence and recombination events until a single grand common ancestor is reached. A search is performed over the possible sequences of events, attempting to find a history with a given number of recombination events. If this does not exist, the number of permitted recombination events is increased by one, and so on, until an ARG is found.

2.2 Motivation for the Method

In order to be applicable to the current cohort of case-control association studies, any method must be computationally feasible when applied to data involving thousands of individuals typed for hundreds of thousands of markers, and where the data is unphased and has missing genotypes. However, the minimal ARG methods described above are limited in that they can only be applied to the smallest of datasets; underlying this is the fact that finding the minimum number of recombination events is NP-Hard (Wang et al., 2001). Additionally, such methods often require the sequence data to be phased, which is potentially a problem because the way in which the data is phased will affect the minimum number of recombination events.

The method described here is able to explicitly construct ARGs for thousands of individuals, typed at hundreds of SNPs. This is achieved by removing the requirement that minimal ARGs are inferred. The algorithm can also handle unphased and missing data.

As discussed in Chapter 1, there are computationally intensive statistical methods based on approximations to the coalescent-with-recombination, which can often be applied to large data sets. However, these approximate methods do not explicitly construct ARG topologies. The algorithm described here is heuristic, and I do not claim to sample from the coalescent-with-recombination, but rather, I attempt to show that the algorithm infers plausible ARGs.

My contribution in this chapter is therefore a method that explicitly constructs ARGs while remaining applicable to large-scale population genotype data. While the method falls into the line of discrete ARG construction methods, its use is not to construct minimal ARGs. Rather, in subsequent chapters, it is used to tackle population genetics questions that are often the domain of statistical methods.

The algorithm described here has been implemented in a program called MARGARITA (Minichiello and Durbin, 2006), which is available for download from http://www.sanger.ac.uk/Software/analysis/margarita/

2.3 ARG Inference

In order to help develop an intuition for how the ARG inference algorithm works, I first give an informal description.

The goal of the algorithm is to coalesce all sequences into a grand common ancestor. Since a coalescence event corresponds to an ancestral sequence transmitting two copies of itself, a coalescence can only be made when two sequences are identical.

In order to make two sequences identical, mutations can be added, flipping any conflicting alleles into agreement. However, under the infinite sites model, a mutation can only occur once in the history of a SNP, meaning that there will be exactly one mutation in the ARG for every polymorphic site. Consequently, a mutation can only be applied at a position when there is only one copy of the mutant allele in the sequences. By applying coalescence events, the number of copies of an allele is gradually reduced to one, allowing this.

However, coalescence and mutation alone will not always be sufficient to arrive at a grand common ancestor. By applying a recombination event to a sequence, the sequence is split onto two separate lineages. On the left parent of the recombination event, the genetic material to the right of the recombination breakpoint is undefined because it was not successfully transmitted into the sampled sequences. This means that while a recombination event increases the number of lineages by one, the actual amount of known genetic material remains constant. Furthermore, since there is no information on what was contained in the untransmitted regions, we let those regions coalesce with anything. This means that it may then be possible to coalesce a recombination parent with another sequence, where there was in the child a mismatching allele to the other side of the recombination.

Figure 2.1 illustrates this logic:

- A. Four sequences sampled from the contemporary population.
- B. It is possible to remove singleton alleles with a mutation event back to the ancestral allele.
- C. Since none of the sequences are identical, no coalescences are possible; in addition, no further mutations are possible. However, the second and third sequence are identical at the first two positions. Putting a recombination on the third sequence between the second and third positions will subsequently allow a coalescence.
- D. Undefined genetic material can coalesce with anything, allowing 01**.** and 011 to coalesce. Note that **..**0 and 110 can also be coalesced, but the ordering is chosen at random.
- E. After the coalescence event, there is only one copy of the 1 allele at the second position, allowing this to be mutated to 0.
- F. Two sequences are identical, allowing a coalescence.
- G-I. Further mutations and coalescences are performed until a grand common ancestor is reached.

Compared to other discrete ARG methods, the methods works from the bottom up, that is, backward in time, as in Myers and Griffiths (2003) and Lyngsø et al. (2005), rather from left to right across the sequences, as in Song and Hein (2004). The independently developed approach of Lyngsø et al. (2005) is the most similar, the fundamental difference being that their method performs a branch and bound search over the ARG construction operations in order to minimise the number of recombinations.

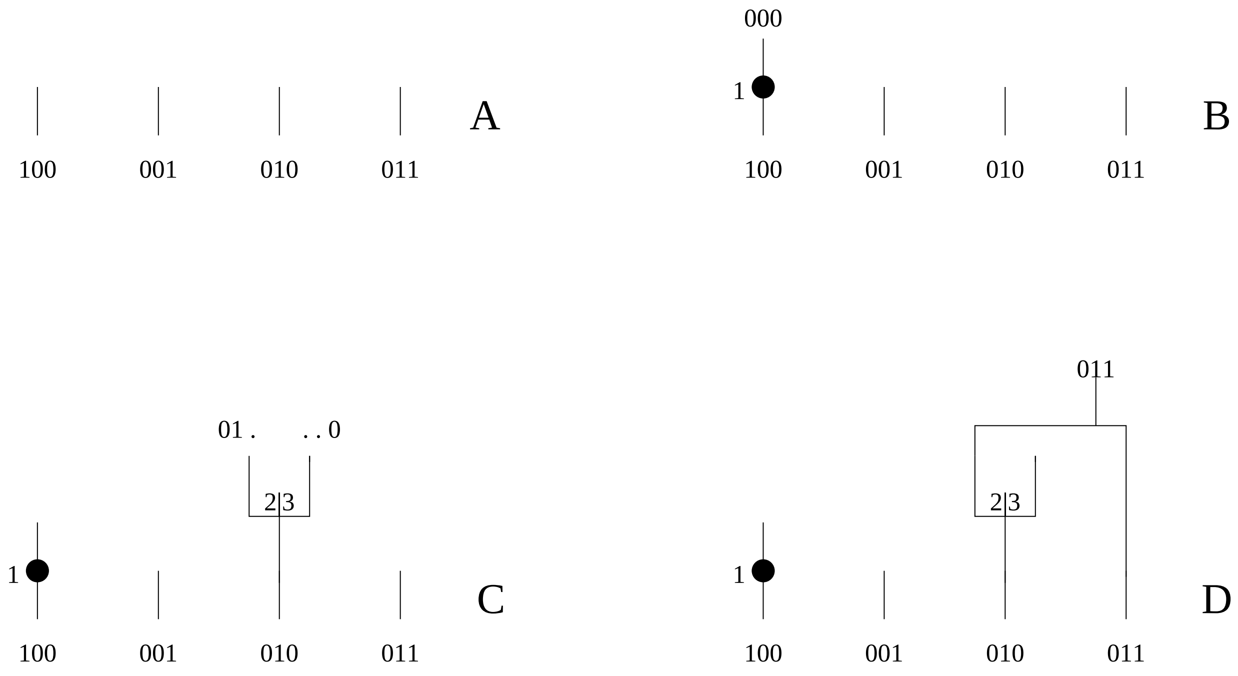
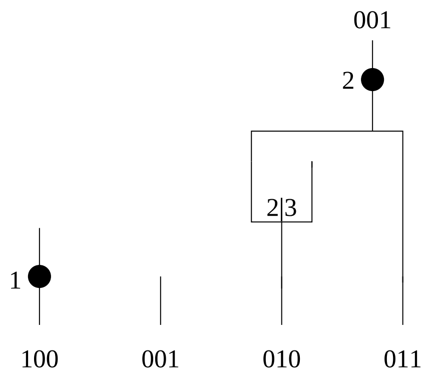
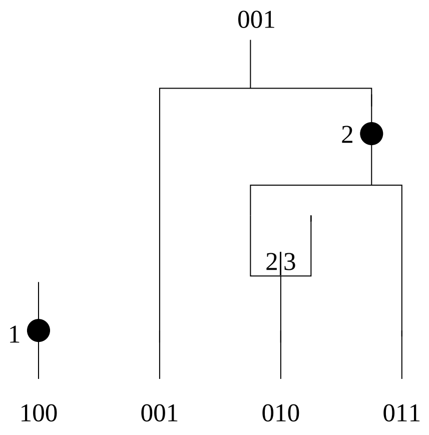


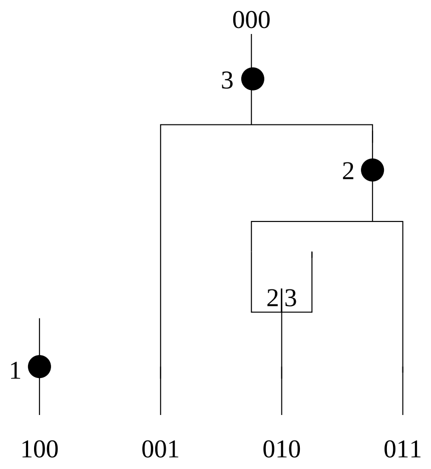
Figure 2.1: Constructing an ARG, see the text for a description. Figure continued on next page.


001
2
2|3
1
100
001
010
011

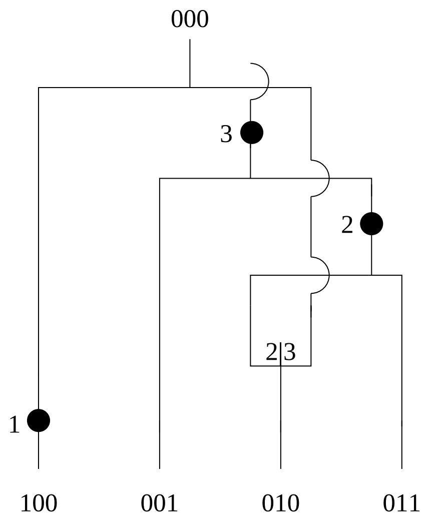
E


001
2
2|3
1
100
001
010
011

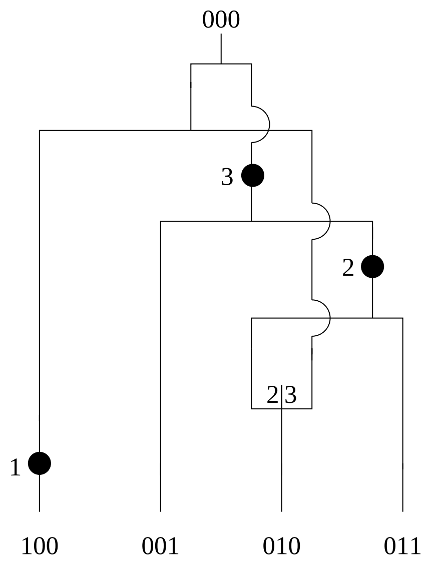
F


000
3
2
2|3
1
100
001
010
011

G


000
3
2
2|3
1
100
001
010
011

H


000
3
2
2|3
1
100
001
010
011

I

2.4 ARG Inference Algorithm

I now describe the algorithm precisely.

The algorithm works backwards in time from the contemporary, typed population of chromosome sequences to a single ancestor sequence. Each step back in time, accomplished with a recombination, mutation or coalescence, defines an ancestral population of sequences. We denote the set of sequences at time T as S_T, and the sequences are, in the phase-known case, strings of length m from the alphabet $\{0, 1, \centerdot\}$, where m is the number of markers, 0 is one of the SNP alleles, 1 is the other allele and $\centerdot$ denotes an undefined copy of the allele—undefined because it was not inherited by any sequences in the contemporary, typed population.

The allelic state of a SNP on sequence C is denoted $C[i]$, where i is the marker position, numbered from 1, so $1 \leq i \leq m$. We define $C_1[i] \sim C_2[i]$ if and only if $C_1[i] = C_2[i]$ or $C_1[i] = \centerdot$ or $C_2[i] = \centerdot$. We define a complement operator $\neg$ such that if $C[i] = 0$ then $\neg C[i] = 1$ and vice versa, and $\centerdot$ is its own complement.

There is a shared tract between sequences C_1 and C_2, over the contiguous set of markers $a, \ldots, b$, if :

1. $C_1[i] \sim C_2[i]$ for all $a \leq i \leq b$;
2. there is at least one i for which $C_1[i] = C_2[i] \neq \centerdot$;
3. if $a > 1$ then $C_1[a-1] \neq C_2[a-1]$ and neither are $\centerdot$;
4. if $b < m$ then $C_1[b+1] \neq C_2[b+1]$ and neither are $\centerdot$.

(1) requires that the two sequences have the same allelic state over the shared tract; (2) requires that for at least one position in the tract, both sequences are defined; and (3) and (4) require that the shared tract is maximal. We denote such a shared tract as $\{C_1, C_2\}[a, b]$.

The algorithm is initialised at time $T = 1$ (T is incremented as we move back in time) by setting S_1 to be the set of contemporary, typed sequences. The algorithm proceeds by finding which coalescences, mutations and recombinations can be performed, determining this according to the rules below. Applying one of these operations defines an ancestral population

S_{T+1}, which is constructed from S_T using the transitions also described below. The algorithm continues in this way until it arrives at a population with only one sequence.

- **Coalescence.**

  Rule. If there exist two sequences C_1 and C_2 in S_T such that for all i, $C_1[i] \sim C_2[i]$, then C_1 and C_2 can be coalesced into an ancestor.

  Transition. $S_{T+1} = (S_T \setminus \{C_1, C_2\}) \cup \{C'\}$ where $C'[i] = C_1[i]$ when $C_1[i] \neq \textbf{.}$ and $C'[i] = C_2[i]$ otherwise. (By $(S_T \setminus \{C_1, C_2\}) \cup \{C'\}$ we mean S_T with the sequences C_1 and C_2 removed and the sequence C' added in.)

- **Mutation.**

  Rule. If there exists a sequence C_1 in S_T and a marker i, where for all C_2 in $S_T \setminus \{C_1\}$ we have $C_2[i]$ is $\neg C_1[i]$ or $\textbf{.}$, then we can remove the derived allele ($C_1[i]$) from the population.

  Transition. $S_{T+1} = (S_T \setminus \{C_1\}) \cup \{C'\}$ where $C'[i] = \neg C_1[i]$ and $C'[j] = C_1[j]$ for all $j \neq i$.

- **Recombination.**

  Rule. When the rules for coalescence and mutation are not satisfied, we must perform a recombination (or a pair of recombinations) instead. We denote a recombination breakpoint as (α, β) meaning that it occurs between markers α and β. Picking a shared tract $\{C_1, C_2\}[a, b]$ from those available in S_T, the aim becomes putting recombinations on the lineages of C_1 and C_2 such that one recombination parent of C_1 and one recombination parent of C_2 satisfy the rule for coalescence. To do this, we must put a breakpoint at $(a-1, a)$ if $a \neq 1$, and a breakpoint at $(b, b+1)$ if $b \neq m$.

  Transition. From the tract $\{C_1, C_2\}[a, b]$, pick (1) a valid breakpoint (α, β), either $(\alpha, \beta) = (a-1, a)$ or $(\alpha, \beta) = (b, b+1)$; and (2) a recombinant sequence C_R, either $C_R = C_1$ or $C_R = C_2$. Then $S_{T+1} = (S_T \setminus \{C_R\}) \cup \{C'_1, C'_2\}$ where $C'_1[i] = C_R[i]$ for $i \leq \alpha$ and $C'_1[i] = \textbf{.}$ otherwise; and $C'_2[i] = C_R[i]$ for all $i \geq \beta$ and $C'_2[i] = \textbf{.}$ otherwise. If both $(a-1, a)$ and $(b, b+1)$ are valid breakpoints (that is, $a \neq 1$ and $b \neq m$), we must put the second recombination (taking us to state S_{T+2}) on an appropriate ancestor of

C_1 or C_2. See Figure 2.1 for an example.

These rules define the constraints on the algorithm that must be enforced if it is to produce legal ARGs. However, at any stage of the algorithm there may be a number of different coalescences, mutations or recombinations that satisfy the rules. We choose between these using the heuristics below, and the stochastic elements result in novel ARGs being generated each time the algorithm is run.

1. **Only perform a recombination if no mutations or coalescences are possible.**
2. **If it is possible to add multiple mutations and/or multiple coalescences at the same time, the order in which these are done is chosen arbitrarily.**
3. **Only coalesce sequences if they have an overlapping region of defined material,** i.e. the two sequences must match for at least one position that is not **.**. This restriction reflects ideas in the sequentially-Markovian coalescent-with-recombination model (McVean and Cardin, 2005).
4. **Recombinations are added at the ends of longer shared tracts first.** During the recombination step, I choose a shared tract $\{C_1, C_2\}[a, b]$ such that the base pair distance between markers a and b is maximised. I only use this heuristic a user-specified proportion of the time *plong*, and at other times $(1 - plong)$ a randomly selected shared tract is used.
5. **The first coalescence after a recombination is based on the shared tract that was used to decide the location of that recombination.**

Heuristic 1 was chosen in order to produce more parsimonious ARGs (fewer recombination events). Heuristic 2 was chosen in order to increase the stochastic element of the algorithm and thus the space of ARGs explored. Heuristic 3 was chosen in order to simplify the system. In McVean and Cardin (2005) it was shown that simulation using this restriction does not cause significant departure from simulations resulting from the standard coalescent-with-recombination. Heuristic 4 reflects the fact that longer shared tracts will tend to arise from more recent recombination and coalescence events. However, because this is only a tendency, not absolute, I only use this heuristic a certain proportion of the time *plong*, and break this heuristic with probability $1 - plong$, when I instead use a randomly selected shared tract to position the recombination breakpoint(s). Throughout this thesis $plong = 0.9$, except as discussed in the next section. Heuristic 5 follows on from Heuristic 4 and again ensures that longer shared tracts are coalesced earlier.

There are a number of other heuristics which could be used. For example, basing the shared tract selection on genetic distance rather than physical distance. Another option would be to select shared tracts which terminate at the positions where recombinations have already been inferred in the ARG. This could be done iteratively, thereby making a preference to place recombinations in hotspots. Additionally, the ancestral sequence at the root of the ARG could be fixed to that suggested by, for example, Chimp data. Additionally, the heuristics could be modified to allow errors in genotyping, or multiple mutations at the same site.

In order to help guide the selection of suitable heuristics and their parameters, I used the set of small case-control studies simulated in Zollner and Pritchard (2005). I evaluated the fine mapping performance for ARGs inferred using different heuristics and adopted those which are (1) genetically sensible and (2) result in discernibly better mapping performance (mapping using inferred ARGs is described in the next chapter). In the next section I show the effect of varying the *plong* heuristic parameter on ARG inference.

2.5 Comparison to the Coalescent-with-Recombination

In this section I compare the ARGs inferred using the above algorithm, which I implemented in a program called MARGARITA, to those generated under the neutral coalescent-with-recombination.

I simulated samples of haplotype sequences using Hudson's MS program (Hudson, 2002), and compared:

- The true marginal trees from MS, which describe the true genealogical history underlying the sample, with the marginal trees inferred by MARGARITA; and

- The number of non-recombining segments as reported by MS (giving a lower bound on the true number of recombinations), with the number of recombinations inferred by MARGARITA.

Note that I compared marginal trees because MS does not output the full ARG. Also, MS does not output the total number of recombination events in the sample history, it only outputs the number of non-recombining segments, giving a lower bound on the true number of recombinations (number of segments - 1).

Using MS I simulated 100 samples of 100 haplotypes, over a 10kb region with a recombination rate of 2.2×10^{-9} per generation per nucleotide and a mutation rate of 1.1×10^{-9} per generation per nucleotide, and an effective population size of 4×10^6, giving a population scaled recombination rate for region of $\rho = 88$, and a scaled mutation rate of $\theta = 44$, and on average 230 segregating sites.

In order to compare MARGARITA's inferred marginal trees with those from MS, I calculated a tree correlation score for each true tree with the inferred trees at that position. Let $\mathcal{B}(T)$ be the set of all non-unary, inequivalent bipartitions of the leaves for a binary tree T. Each bipartition is obtained by cutting an internal branch of the tree and seeing which leaves fall under the cut, and which do not. There are $n - 3$ such bipartitions. The tree correlation score between trees T_1 and T_2 is then:

$$\frac{|\mathcal{B}(T_1) \cap \mathcal{B}(T_2)|}{n-3},$$

which is the proportion of bipartitions that are shared between the two trees. I use this metric because it is directly related to disease mapping using inferred ARGs. In my approach to disease mapping, described in the next chapter, hypothetical disease causing mutations are placed on marginal trees, and these mutations partition the leaves into two non-overlapping sets: those chromosomes with or without the putative causative mutation. The accuracy in partitioning reflects the accuracy of causative mutation inference.

For one simulation, Figure 2.2 shows the tree correlation of the inferred marginal tree at each segregating site with the true marginal tree for that position, averaged over 100 ARG

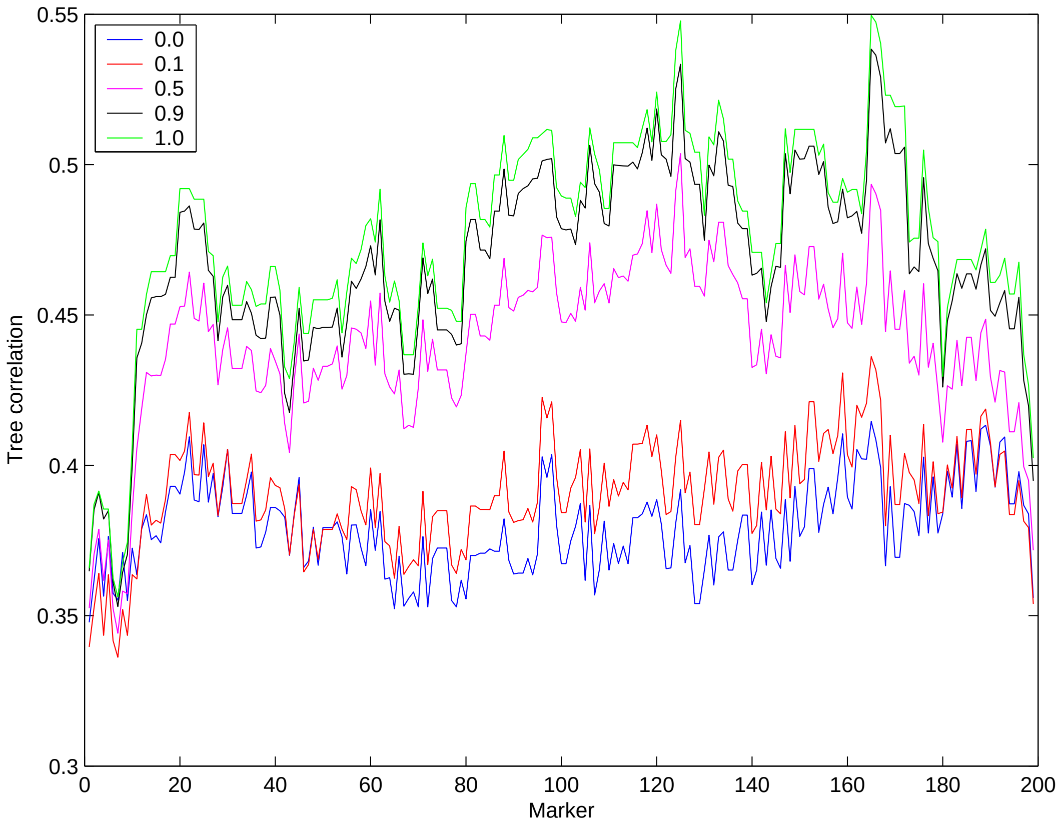


Figure 2.2: Tree correlation of the inferred marginal tree with the true tree simulated by MS at each segregating site, averaged over 100 inferred ARGs. This is for one simulated population, with 100 haplotypes over 10kb, with $\rho = 88$ and $\theta = 44$, with 199 segregating sites. Each line corresponds to a different value of the heuristic parameter *plong*.

inferences. Five lines are plotted, each one corresponding to a different value of *plong*, the heuristic parameter which specifies the frequency with which a longest shared tract is used to position recombination events, rather than a randomly selected shared tract.

Marginal tree reconstruction is more accurate (higher tree correlation) for greater values of *plong*, indicating that the longest shared tract heuristic appropriately captures features from the neutral coalescent-with-recombination. Also note that the accuracy of marginal tree reconstruction is better away from the edges of the "typed" region. This is expected because there is less information towards the edges about the long range sharing of haplotypes.

Figure 2.3 shows the quality of marginal tree reconstruction for samples of 100 sequences over a 10kb region, with a scaled mutation rate of $\theta = 44$ as above, but now with $\rho \in \{8, 88, 880\}$. The number of haplotypes in each sample was also varied, to be either 30 or 100,

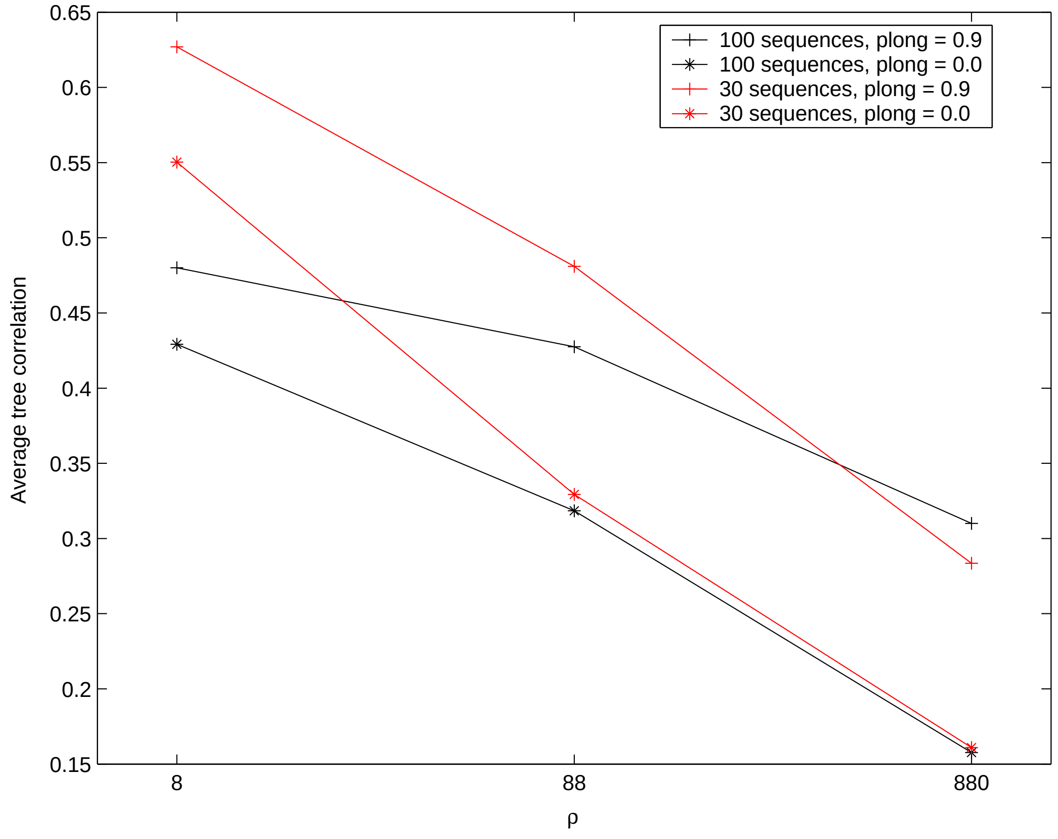


Figure 2.3: Tree correlation of inferred marginal trees with the true trees for a range of population scaled recombination rates ρ, heuristic parameters *plong* and number of sequences. Each point corresponds to the average over all segregating sites in 100 simulations, over 100 ARG inferences for each simulation.

and the heuristic parameter *plong* was set to either 0.9 or 0.0

As the recombination rate increases, the quality of marginal tree reconstruction decreases. This is because there become many more shifts in MS tree topology than segregating sites (θ remains set at 44), and it is harder to observe those recombination events (and their impact on the marginal tree topology) from the data.

Marginal tree reconstruction tends to be more accurate for samples with fewer sequences. This is expected because when more sequences are sampled, which may be very similar, the number of possible tree topologies within each non-recombining region increases, as does the uncertainty in the ordering of coalescence events.

As already observed, $plong = 0.9$ gives more accurate reconstruction than $plong = 0.0$. This helps justify the choice of setting $plong = 0.9$ for the disease mapping experiments in

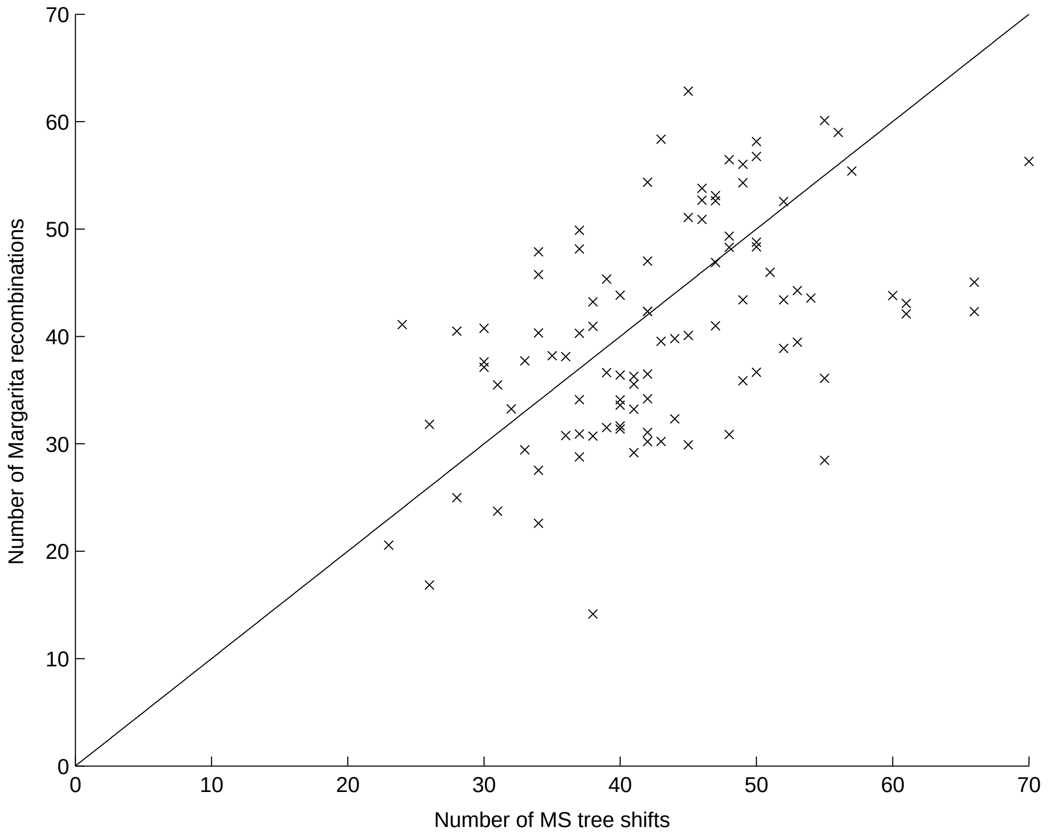


Figure 2.4: The number of recombination events in inferred ARGs from MARGARITA, compared with the number of tree shifts in the true history as simulated by MS. For 100 simulations, each with 100 haplotypes, with $\rho = 8$, $\theta = 44$ and *plong* $= 0.9$. The line $y = x$ is plotted.

latter chapters.

Figure 2.4 gives the number of recombination events in MARGARITA's inferred ARGs, compared with the number of shifts in tree topology in the true ARGs simulated by MS (the number of non-recombining segments - 1). The number of tree shifts gives a lower bound on the true number of recombination events. The number of inferred recombination events matches the number of tree shifts well (for these simulations with parameters $\rho = 8$, $\theta = 44$). However, the slope of the linear regression line for the data points in Figure 2.4 is 0.54. Part of this underestimation in the number of recombinations may be due to MARGARITA only inferring recombination events when no coalescences or mutations are possible. Also, there will be some recombination events for which there is no evidence in the data, and the proportion of these will increase as the recombination rate increases.

In the following chapters I discuss the application of inferred ARGs to disease mapping. The good mapping performance achieved when using inferred ARGs gives further evidence that they capture correctly many important properties of the true genealogy.

2.6 Existing Methods for Handling Unphased Data

So far I have only considered phase known haplotype sequences, but by extending the algorithm it is possible to resolve haplotype phase and impute missing data while constructing an ARG, and this is described in the next section. First I define more clearly the phase resolution problem and discuss other algorithms for handling unphased data.

In diploid organisms, there are two copies of each autosome. The sequence of a single chromosome is the haplotype sequence, and the genotype sequence is the conflation of those two haplotype sequences. Consider a SNP where there are two alleles, 0 and 1. If both copies of the haplotypes have the 0 (respectively 1) allele, then the genotype sequence will register homozygous 0 (respectively 1). However, if one of the haplotypes has the 0 allele, and the other haplotype the 1 allele, the genotype sequence will register a heterozygote. When genotype sequences are obtained, heterozygote calls do not indicate which chromosome has the 0 allele, and which has the 1 allele; and without further information it is impossible to determine the haplotype sequences.

Although haplotypes can be determined experimentally (Patil et al., 2001), current large-scale sequencing and genotyping technologies only provide genotype sequences, and it is more common to determine haplotype sequences computationally by using information from other individuals in the population (LD).

We write a genotype sequence as a string of 0,1,U characters, where 0 indicates that the individual is homozygous for the 0 allele; 1 indicates the individual is homozygous for the 1 allele; and U indicates that the individual is heterozygous. The haplotype inference problem is to correctly resolve the haplotype sequences from genotype sequences.

There are a number of existing phasing algorithms. An early one was Clark's algorithm (Clark, 1990). This algorithm starts by identifying any genotype sequences that are completely resolved, having no unphased U characters; this genotype sequence yields two identical haplotype sequences. The algorithm then looks for a match between an unphased genotype sequence and a phased haplotype sequence, that is, where the homozygous positions on the unphased sequence have the same alleles as the phased sequence. When such an instance is found, the unphased sequence is resolved into two haplotype sequences: one is the same as the phased sequence, and the other is set to the complement in the U positions. For example, if the genotyped sequence is 0UU0, and there is a phased haplotype sequence, 0100, then 0UU0 becomes the two haplotype sequences: 0100 and 0010. When a genotype sequence is resolved it is removed from the set of genotype sequences, and the resolved haplotype sequences are added to the set of sequences available for matching to the remaining genotype sequences. The algorithm stops when all genotype sequences have been resolved or when no further phase resolution is possible. With this approach, the order in which sequences are resolved affects the end phased result.

The underlying population genetic model for Clark's algorithm is that sequences in a population are likely to be similar to each other. This approach can also be viewed as an attempt to minimise the total number of haplotypes observed in the sample and, hence, is a parsimony approach.

A more common approach is to use Expectation-Maximisation (EM) to find maximum likelihood estimates for the haplotype phases (Excoffier and Slakin, 1995; Long et al., 1995; Chiano and Clayton, 1998). This overcomes the problems associated with Clark's algorithm, that the haplotype resolution depends on the order in which the algorithm is run, and that the algorithm will not be able to start if all the individuals have ambiguous haplotypes.

The likelihood of a possible phase resolution, assuming Hardy-Weinberg equilibrium, is

$$L = \prod_{h=1}^{n_d} \frac{\pi\left(h_1\right)\pi\left(h_2\right)}{\left(2n_d\right)^2},$$

where n_d is the number of diploid individuals, and the two haplotypes for individual h are h_1 and h_2, and the frequencies of those haplotypes in the population are $\pi\left(h_1\right)$ and $\pi\left(h_2\right)$.

The EM algorithm works iteratively to compute the unobserved haplotype frequencies

$\pi(h_i)$ and phase resolutions, starting from arbitrary initial values $\pi(h_i)^{(0)}$, for example, where all haplotypes have equal frequency. The initial values are taken as though they are the true values and used to estimate the probability of each possible phase resolution for each individual, and any missing genotype data are jointly estimated. This is called the expectation step.

These estimated phase resolutions are then used to estimate the haplotype frequencies in the next iteration $\pi(h_i)^{(1)}$. By summing the probabilities for each distinct haplotype and scaling the sum to 1, we obtain the updated haplotype frequencies (the maximisation step). This continues until convergence: when the changes in haplotype frequencies in consecutive iterations are small.

Another approach to finding maximum likelihood estimates is to generate possible phase resolutions using a tree model, and then to select the resolution that has the maximum likelihood (Halperin and Eskin, 2004).

The methods of Gusfield (2002) and Halperin and Eskin (2004) utilises the observation that haplotypic variation is organised into blocks of limited diversity (Daly et al., 2001), which result from population substructure, bottlenecks and recombination hotspots. Daly et al. (2001) and Rioux et al. (2001) genotyped 103 common SNPs in an 500 kb region for 129 parents-offspring trios. They found that the region can be divided into blocks (referred as haplotype blocks) spanning from tens up to about one hundred kb.

In these blocks, although there may be m SNPs, there are far fewer than 2^m haplotypes in the population: typically up to 5 different haplotype sequences (Daly et al., 2001; Rioux et al., 2001). Within haplotype blocks, the haplotypic variation may be organised on a tree, explaining how the haplotypes are related to each other by mutation. In Halperin and Eskin (2004), haplotype blocks are defined using an approach similar to the dynamic programming method of Zhang et al. (2002), which jointly maximises the lengths of blocks while minimising the number of SNPs within those blocks required to unambiguously tag the haplotypes. The algorithms of Gusfield (2002) and Halperin and Eskin (2004) differ in how strictly sequences within a block must fit a tree under the infinite sites model. Trees are then constructed for the sequences within a block, and these yield haplotype sequences at the leaves. From amongst the candidate solutions that fit the block and tree model, the phase resolution is chosen that maximises the likelihood of observing the genotypes given the haplotype frequencies.

There are methods, such as PHASE (Stephens et al., 2001; Stephens and Donnelly, 2003; Stephens and Scheet, 2005) and FASTPHASE (Scheet and Stephens, 2006) which do not require haplotypes to be organised into blocks.

PHASE (Stephens et al., 2001; Stephens and Donnelly, 2003; Stephens and Scheet, 2005) is partly based on the Li and Stephens (2003) model, described in Chapter 1. PHASE uses Markov chain-Monte Carlo to approximately sample form the distribution of haplotypes given the genotypes. It starts with an initial guess of the haplotype phases, and then repeatedly selects an individual at random, and assuming that all other individuals are correctly phased, estimates that individual's pair of haplotype sequences. The probability of a particular haplotype pair given the haplotypes of the other individuals is approximated using a model similar to Li and Stephens (2003). A large number of haplotype reconstructions are sampled in this way, discarding the first, say, 100,000 and keeping 200,000, sampled every 100 iterations (Stephens et al., 2001). These can then be used to estimate a single best haplotype phase resolution for the data.

FASTPHASE (Scheet and Stephens, 2006) is again based on the observation that over short distances, haplotypes tend to cluster into groups of similar haplotypes, however, it does not force cluster membership to change at haplotype block boundaries. Rather, membership is allowed to change continuously along the chromosome, and this is modelled by a hidden Markov model. This ensures that at adjacent loci, a haplotype is more likely to remain in the same cluster. This results in haplotype sequences which are made up of mosaics of similarity with other haplotypes. This is a more realistic model because while haplotype blocks are believed to arise in part from recombination hotspots, not all recombination occurs in hotspots, hence cluster membership does change continually. The probability that part of a haplotype belongs to a particular cluster is dependent on the relative frequency of that cluster in the population and the allele frequencies in that cluster. The probability that a haplotype sequence changes cluster membership between loci is dependent on the recombination frequency between them and the frequencies of the clusters.

The parameters of the fastPHASE model are estimated using expectation-maximisation. This typically finds parameter estimates corresponding to local likelihood maxima; however the goal of fastPHASE is not to explicitly estimate these parameters, rather it is to find phase resolutions. Therefore, the expectation-maximisation algorithm is run multiple times, and results averaged across the different runs.

Phased haplotype sequences can then be sampled from the fitted model. In order to estimate the phase resolution for an individual, pairs of haplotypes which are consistent with the genotype sequence are sampled, and the most frequent pair can be taken as the best estimate. FastPHASE can also be used for missing genotype imputation, which is described in Chapter 6.

Of the current haplotype phase resolution methods, comparative studies appear to show that PHASE provides the most accurate performance (Marchini et al., 2006). However, computationally, fastPHASE is significantly more efficient than PHASE.

One solution to handling missing and unphased data is to apply one of the algorithms described above as a preprocessing step, and then construct ARGs for the resolved data. However, doing so with association studies can result in a loss of statistical power, because the uncertainty in the phase resolution is not taken into account (Morris et al., 2004). Hence, I have modified the algorithm to operate directly on unphased and missing data.

2.7 Unphased and Missing Data with Inferred ARGs

Handling missing data is the simpler of the two cases. A missing character is allowed to coalesce with any other character (0,1,**.**, or another missing character), and when it coalesces with a known allele (0 or 1), the missing character becomes fixed to that allele and this assignment is propagated down the ARG to the leaves.

Phasing data is similar, except a record of the diploid pairings of chromosomes is kept. A phase-unknown character may not coalesce with the corresponding phase-unknown character on its sister chromosome (because the individual is heterozygous at that position). When a phase-unknown character coalesces with a known allele, its phase becomes fixed, as does the character on its sister chromosome, although to the complement allele. When phase-unknown characters from two chromosomes coalesce, these chromosomes and their sisters become dependent on each other. Neither of those chromosomes may coalesce with the sister of the other one. And when one of the chromosomes has a position phase-resolved, that position is also resolved on the other chromosome, and the two sister chromosomes are set to the complement allele. Of course, many more than four chromosomes can become involved in such interdependencies.

Figure 2.5 gives an example of this logic:

- A. Four sequences. The first two are from the same individual and are heterozygous for the third position; the fourth sequence has a missing character in the second position.
- B. An unphased character can match any other character as long as it is not from the sister chromosome (or a dependent), hence the coalescence shown is possible.
- C. The unphased character becomes fixed by inheritance from the coalescence parent.
- D. The dependency that the sister chromosome has the complement allele is propagated through the ARG.
- E. A missing character can coalesce with anything.
- F. The missing character is imputed by inheritance from the coalescence parent.

This approach is similar to Clark (1990), which performs the equivalent of coalescences between phase unknown genotype sequences and haplotype sequences. A key difference being that the method described here also allows mutation events and coalescences over subregions (via recombination). Nor does my method require any of the input sequences to be completely phased to begin with, and hence always finds a phase resolution. Like Gusfield (2002) and Halperin and Eskin (2004), phasing happens on a tree structure, however, my approach does not assume haplotype blocks and instead uses the inferred recombination events to define the span over which haplotypes are compared.

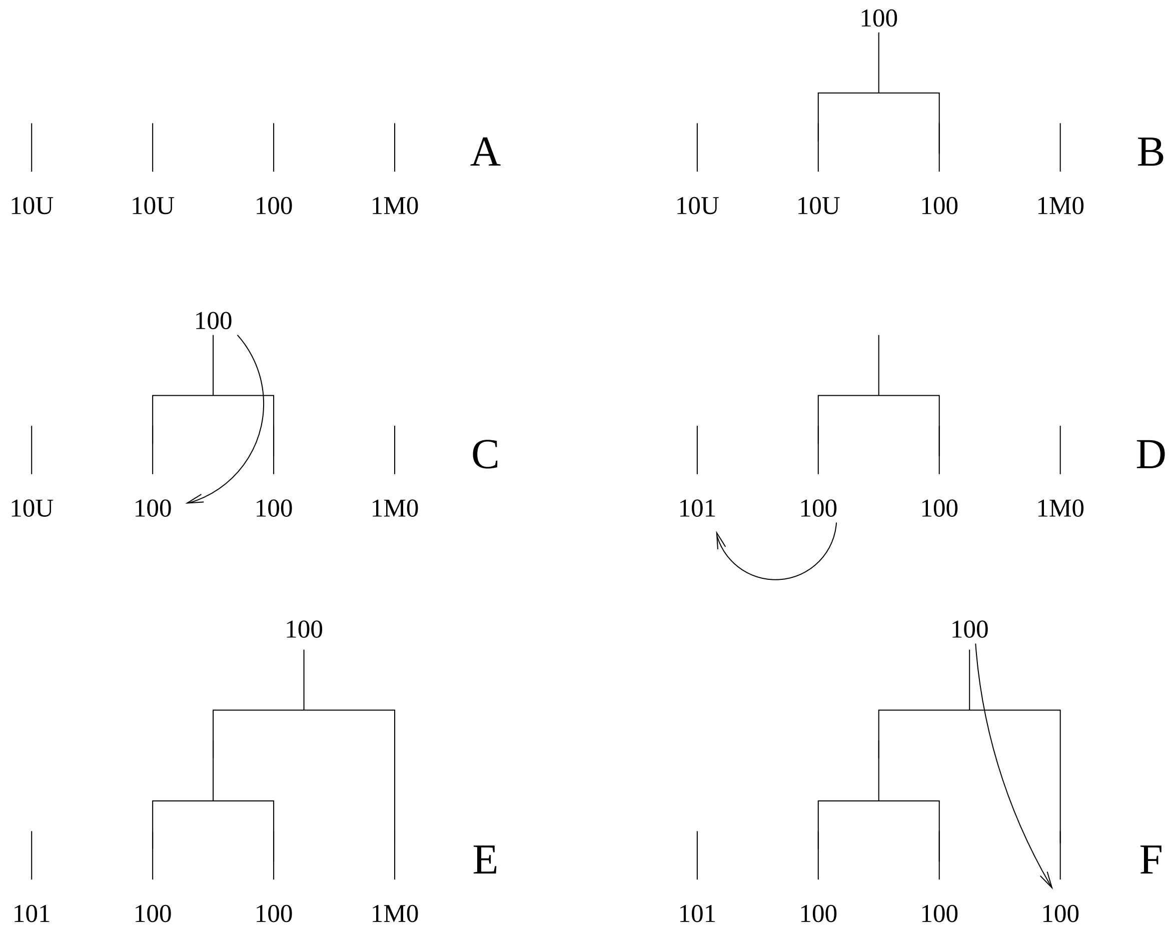


Figure 2.5: Inferring haplotype phase and missing data. See the text for a description.